DISENTANGLING THE EXECUTION GAP FROM THE INFORMATION GAP WITH CONDITIONAL COMPUTATION IN VISION-AND-LANGUAGE NAVIGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-and-language navigation turns a spoken instruction into a route through a real building, so a robot can act on directions in a space it has not been given as a map. In continuous environments the agent sees only first-person frames and moves through low-level steps and turns. Recent video-language navigators process image histories and emit actions. Yet these tasks still differ in their information requirements: some require finding or grounding a target, whereas others directly specify distances, angles, or a return to the starting point. A single aggregate success rate cannot tell whether a failed navigator lacks the ability to execute or lacks the information needed to determine the correct action. We route each episode by how completely its instruction already states a self-motion program. When language specifies that program, we introduce **P**roprioceptive **Nav**igation (PNav), a conditional computation architecture that holds an origin register outside the frozen policy and passes the program to an external executor. Without training, PNav reaches a new SOTA success rate on NavSpace and leaves trajectories outside its takeover scope unchanged. Further closed-loop and perceptual-aid tests indicate that the tested readout, control and bypass-perception modules recover only part of the gap to a privileged-information reference. These results suggest that VLN failures have different causes: when language already specifies an action-relevant motion program, scaling the visual model alone may target the wrong bottleneck, while other instructions still need a usable goal condition.

1 INTRODUCTION

Vision-and-language navigation asks an agent to follow a natural-language instruction through an unseen building and to choose each step from what it sees. The ability lets a robot carry out spoken directions indoors without a prebuilt route. Early approaches implemented this capability as navigation between panoramic viewpoints on a fixed graph (Anderson et al., 2018). Continuous control removed that graph, so the agent must step and turn from a first-person view (Krantz et al., 2020). NavSpace keeps this setting and asks whether the agent can follow spatial instructions, from precise movement to viewpoint shifts and scene structure (Yang et al., 2026). The systems that now carry this ability are video-language models that emit one action at a time, among them NaVid (Zhang et al., 2024), NaVILA (Cheng et al., 2025), Uni-NaVid (Zhang et al., 2025a), StreamVLN (Wei et al., 2026) and SNav (Yang et al., 2026).

Recent systems extend visual histories or change training and action generation (Zhang et al., 2024; 2025a; Wei et al., 2026). Even so, the instructions themselves do not ask the same thing of the agent. "Move forward 1.5 meters, turn right 60 degrees, then go forward 2.75 meters" already supplies a self-motion program, and vision is left to handle collisions. "Find the farthest room" requires exploring the scene and comparing candidates. "Imagine you are a television and move to the left of the television" requires binding that instance and estimating its frame. These examples reflect the distinct spatial instruction categories in NavSpace (Yang et al., 2026). A single aggregate success rate cannot tell those cases apart. We treat the difference as an operational split between an

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

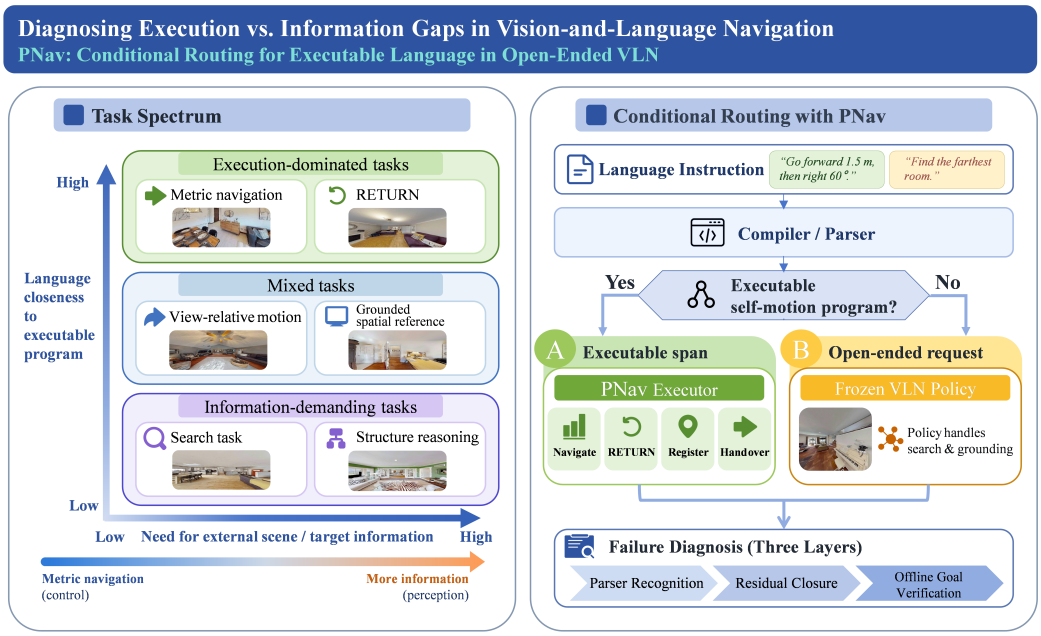


Figure 1: PNav executes motion the instruction already specifies and leaves the rest to the frozen navigator. Left: tasks by language closure and scene need. Right: conditional routing and diagnosis.

execution gap, where language fixes the motion, and an information gap, where the action needs a target or scene that the instruction itself lacks.

We therefore study both gaps. On the execution side, the frozen model is highly insensitive to distance and direction. Because prompted counts and dead reckoning do not become state updated by its own actions, we externalize this control in **P**roprioceptive **Nav**igation (PNav). When language specifies a METRIC or RETURN program, an external executor carries it out using an origin register updated from emitted actions. Otherwise, the frozen policy retains control.

Where language supplies a complete motion program, PNav turns a small minority of successes into a majority across multiple environments and task scenarios. For episodes that never trigger takeover, the mechanism also guarantees stability: under deterministic execution, it preserves the original trajectory and therefore does not degrade the outcome of those episodes. We also obtain informative negative results. Where language does not supply the motion program, experiments show substantial room for further improvement, but the tested identity, bearing and distance readouts recover only part of this potential. We therefore conclude that execution and information gaps call for different remedies. When language already specifies an action-relevant motion program, scaling the visual model alone may target the wrong bottleneck. The information gap still calls for a changed interface or better training and memory.

Our main contributions are as follows:

1. To separate failures of a stated motion program from failures that require scene or target information, we use mutually exclusive offline labels and a hidden-goal verifier. This makes the execution–information split measurable at the benchmark level.
2. We introduce PNav, which executes metric or return programs from an external origin register and otherwise leaves control with the frozen policy. In paired RGB NavSpace runs, it reaches a new SOTA, preserves base trajectories on episodes without takeover and shows measured gains across task interfaces.
3. We find that explicit execution improves navigation on stated motion programs, whereas the tested readouts do not yield a reliable goal condition for grounding-dependent instructions. These findings suggest that the two gaps call for different remedies and that scaling the visual model alone may miss the bottleneck on executable instructions.

2 Related Work

Learned VLN policies. NaVid, Uni-NaVid and StreamVLN study video conditioned navigation with different history and action interfaces (Zhang et al., 2024; 2025a; Wei et al., 2026). NaVILA couples intermediate language commands to locomotion (Cheng et al., 2025), and NavGPT-2 aligns visual input with a frozen language model and navigation policy (Zhou et al., 2024). ETPNav combines online topological mapping and continuous control (An et al., 2025). NavSpace supplies the benchmark with six axes and the SNav baseline used here (Yang et al., 2026). The novelty is conditional execution: when the instruction already states a metric or return program, an external origin register carries it out, and the frozen policy keeps control otherwise.

Modular control. SayCan grounds proposals from a language model with affordances, and Code as Policies produces executable robot policy programs (Ichter et al., 2022; Liang et al., 2023). MapGPT uses an online linguistic map for navigation (Chen et al., 2024b). ObjectNav makes target categories explicit (Batra et al., 2020). AgentVLN combines high level reasoning, skills and geometric execution, while Embodied Navigator links pointing and trajectory memory to control (Xin et al., 2026; Feng et al., 2026). DART-VLN changes memory and action selection at test time without updating its backbone (Zhang et al., 2026b). The novel mechanism is narrower: a parsed METRIC or RETURN command is run from an integrated origin by a deterministic executor.

Information limits. VSI Bench probes spatial recall from video (Yang et al., 2025). SPACENUM and SpatialVLM examine numerical and metric spatial reasoning (Zhang et al., 2026a; Chen et al., 2024a). NavNuances decomposes navigation instructions (Wang et al., 2024), and Embodied4C studies embodied reasoning in a closed loop (Sohn et al., 2025). Privileged experts can depend on information unavailable to a learner (Walsman et al., 2023). Causal confusion in imitation learning shows that predictive success can arise from non-causal correlations and need not identify the mechanism that remains valid under intervention (de Haan et al., 2019). The contribution is to separate parser recognition, residual closure and offline goal compatibility. These labels depend on a fixed grammar, action grid and success criterion. The hidden goal enters only the offline verifier, while our negative result concerns the tested readouts and controllers.

3 Method

3.1 Task, Observations and Information Requirements

An episode is $e = (L, s_0, g, \rho)$, with instruction L, initial pose $s_0 = (p_0, \psi_0)$, hidden scoring goal g and success radius ρ. The harness and executor use the same four tokens $\mathcal{A} = \{\mathsf{F}, \mathsf{L}, \mathsf{R}, \mathsf{S}\}$. The step is $\Delta = 0.25$ m, the turn is $\delta = 30^\circ$ and the budget is $T_{\max} = 70$. The benchmark checks $\|p_T - g\| < \rho$ when S ends the episode at true position p_T (Appendix C.2). A policy without privileged inputs sees $h_t = (L, o_{\le t}, a_{<t})$, where o_t is the RGB frame and a_t is the emitted action. PNav also integrates committed actions and collision feedback b_t, received after a_t, but reads neither g, the true pose nor a navmesh path to g. Metric programs need instruction parameters and proprioception. Grounded tasks additionally need a referent z, its estimated position $q_t(z)$, an object-defined frame T_z, or local scene memory $\mathcal{M}_t$. We summarize these possible dependencies as $a_t^\star = f(L, h_t^{\text{prop}}, z, q_t(z), T_z, \mathcal{M}_t)$, with $h_t^{\text{prop}} = (L, a_{<t}, b_{<t})$. PNav targets cases where f does not depend on scene and target variables, while Section 4.3 tests the remainder.

3.2 A Compiler Relative Split with C Closure

A fixed compiler C emits a program, a residual clause and a closure bit,

$$C(L) = \big(P, L_{\text{res}}, c\big), \qquad P \in \{\bot\} \cup \mathcal{P}_{\text{metric}} \cup \mathcal{P}_{\text{return}}, \qquad c \in \{0, 1\}. \tag{1}$$

Here $P = \bot$ denotes rejection, and L_{res} is the language left after removing the recognized span. The closure bit is set when that residual is empty or contains only STOP. The set $\mathcal{P}_{\text{metric}}$ contains programs $P = ((\theta_i, d_i))_{i=1}^n$ of n ordered segments, each a signed relative turn θ_i followed by a nonnegative advance d_i. The set $\mathcal{P}_{\text{return}}$ contains RETURN programs for an origin from the trajectory up to that point, as stored in the register.

Appendix B.2 defines three diagnostic layers for parser recognition, residual closure and offline goal compatibility. Table 2 groups outcomes by regenerated offline information labels. The hidden goal never enters runtime control on either local arm.

3.3 Conditional Computation and Action Routing

Let the frozen policy be π_θ, the program executor π_{exec} and the arbitration bit $\alpha_t \in \{0, 1\}$. The executor reads the external register R_t: its action-integrated pose, established origins, active program and progress within that program. The wrapped action is

$$a_t = \begin{cases} \pi_{\text{exec}}(R_t, P), & \alpha_t = 1, \\ \pi_\theta(h_t), & \alpha_t = 0. \end{cases} \tag{2}$$

Here $\alpha_t = 1$ when the parser provides an active program and the executor can act within the remaining budget (Appendix A). This runtime eligibility check uses the instruction, parse, budget and register. Neither g nor the offline labels enter arbitration. The policy may be queried for cache continuity, but its action is not sent while the executor acts. Both local arms use the same RGB stream and action interface. The wrapper integrates emitted actions and collision feedback in the origin register for the executor to read next.

3.4 The Executor

3.4.1 The Origin Register with ESTABLISH and RELABEL

The register is a set of local reference frames, not a bare action counter. Let $\hat{s}_t = (\hat{p}_t, \hat{\psi}_t)$ be the action-integrated pose in a frame initialized at $\hat{p}_0 = (0, 0)$ and $\hat{\psi}_0 = 0$. It is updated by the emitted actions and the collision bit as

$$\hat{\psi}_{t+1} = \hat{\psi}_t + \delta \mathbf{1}[a_t = \mathsf{L}] - \delta \mathbf{1}[a_t = \mathsf{R}], \quad \hat{p}_{t+1} = \hat{p}_t + \Delta\,(1 - b_t)\,u(\hat{\psi}_t)\mathbf{1}[a_t = \mathsf{F}], \tag{3}$$

with $u(\psi) = (\cos\psi, \sin\psi)$ and $b_t = 1$ when the forward action is blocked. `establish` stores the current register pose as an origin, and `relabel` copies that pose to a new origin (Appendix A.6). The active RGB configuration uses the starting origin for RETURN. Clause-level relabeling is disabled in the 54.2% run. Neither operation supplies a hidden goal.

3.4.2 Metric and RETURN Execution

Each command is snapped to the action grid at parse time. The executor then uses whole ticks.

$$\widetilde{d}_i = \Delta\,\text{round}(d_i/\Delta), \qquad \widetilde{\theta}_i = \delta\,\text{round}(\theta_i/\delta). \tag{4}$$

On the action grid the executed and ideal endpoints coincide under ideal kinematics without obstacles. Quantities off the grid incur at most half a tick per quantity. In the reported RGB configuration, METRIC emits S at the end of the parsed program.

RETURN is armed after the base policy leaves the origin. With local displacement $\hat{p}_t = (\hat{p}_{t,x}, \hat{p}_{t,y})$, let $\beta_t = \text{wrap}(\text{atan2}(-\hat{p}_{t,y}, -\hat{p}_{t,x}) - \hat{\psi}_t)$. The controller turns while $|\beta_t| > \delta/2$, advances when aligned and $\|\hat{p}_t\| > \rho_{\text{stop}}$, then emits S under the reported configuration, where $\rho_{\text{stop}} = \max(0.75\rho, \Delta)$ (Appendix A.7). On an episode where the wrapper never takes over, the base action is sent verbatim, so deterministic policies have trace identity by induction (Appendix A.5). This tests execution of distances, angles and origin relations specified by language. It does not test whether the base policy parses every numeric instruction correctly. Alignment is checked before the advance. A bearing outside $\pm\delta/2$ produces a turn even when the register is already inside ρ_{stop}, and stop is emitted only when the bearing lies inside that band and $\|\hat{p}_t\| \le \rho_{\text{stop}}$. This margin sits inside the success radius (Appendix A.7). Each command is snapped once, so later ticks do not round the heading again once it is on the grid (Appendix A.1).

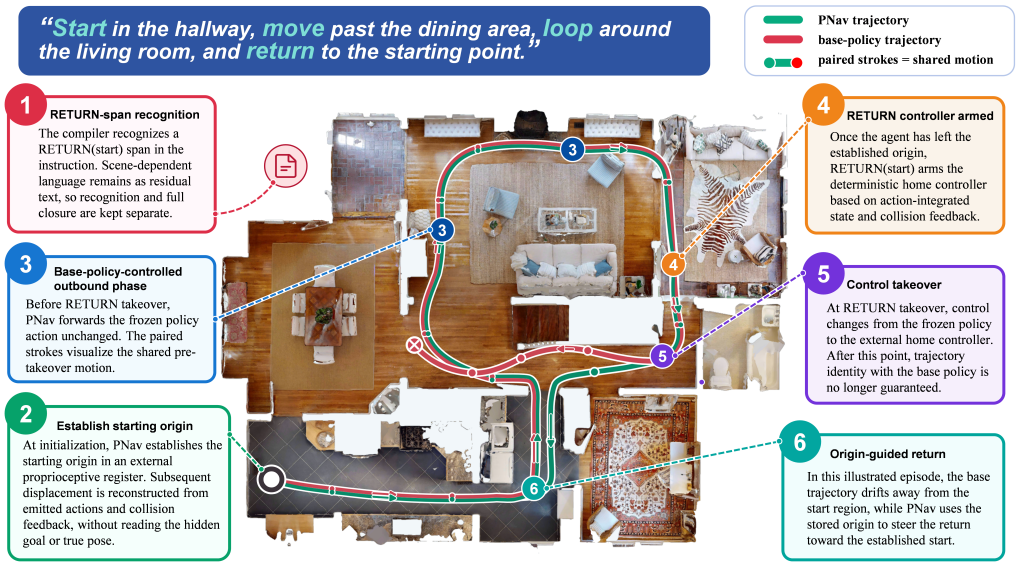


Figure 2: Origin-guided RETURN. PNav forwards frozen-policy actions until takeover, then an external controller steers to the established origin using action-integrated state.

4 Experiments and Analysis

4.1 Evaluation Protocol and the Boundaries of Our Claims

The main result estimates the paired effect of the complete wrapper. Semantic attribution is a narrower secondary analysis. The experiments cover 1,228 episodes on six NavSpace axes. Both arms use the official `TidalYang/SNav-7B` (Yang et al., 2026), a history of 16 frames, at most four actions per step, a 70 step budget and the same success criterion. PNav attaches the compiler and origin register to this RGB policy. Privileged, geometry and corpus diagnostics are reported separately.

Statistics use paired outcomes for each episode, including success rate (SR), the difference between PNav and control, exact McNemar tests (McNemar, 1947) and Newcombe paired score 95% intervals (Newcombe, 1998) (Appendix C.4). Episodes that start inside the goal are reported separately, not counted as method successes. Quantitative claims use completed runs only. Unmeasured extensions are stated as limits in Appendix D. The paired comparisons measure the complete parser–executor wrapper. Table 2 groups outcomes by offline labels regenerated with the metric parser.

4.2 Positive Evidence for Repairing the "Semantics as Program" Slice

4.2.1 Main Results

Table 1 places the method beside published NavSpace results and reports our paired runs in its lower block. The sensor column lists RGB for both local arms, while s marks source-reported inputs for published systems. Issued actions and collision events are available through the shared execution harness, and PNav uses them to update its origin register. Only our lower rows are paired, and ΔM averages the six axis differences. Precise remains difficult for the published systems despite its explicit distances and turns. In our paired runs, PNav improves the six-axis macro rate by 8.1 points. Across 1,228 episodes the micro difference is 8.0 points, with 114 PNav-only and 16 control-only successes. The exact McNemar test gives $p = 2.07 \times 10^{-19}$, and the Newcombe paired score 95% interval for the difference is $[6.3, 9.8]$ points. Precise reaches 54.2% in the RGB run.

4.2.2 Conditional Effects Locate the Mechanism Better than a Macro Score

Table 2 separates metric endpoint-compatible gains from in-goal starts. In the 146-episode slice, PNav succeeds on 104 episodes versus 17 for control. The exact McNemar test gives $p = 3.38 \times 10^{-24}$, and the Newcombe paired score 95% interval for the difference is $[50.8, 67.2]$ points.

Table 1: NavSpace results (success rate, SR, %). Published leaderboard figures (Yang et al., 2026) are included for context.

Method	Sensors	Vert	Prec	View	Rel	Env	Struct	Macro	ΔM
Chance (frequency)	–	11.5	3.5	11.6	7.5	6.5	9.0	8.3	–
Qwen2.5-VL 72B	s	9.1	3.0	5.3	6.1	8.5	10.0	7.0	–
GPT-4o	s	10.1	4.0	3.9	9.9	8.5	9.5	7.7	–
Gemini 2.5 Pro	s	23.6	4.0	9.2	7.1	13.0	10.0	11.2	–
GPT-5	s	18.3	3.0	12.6	17.5	17.5	16.5	14.2	–
BEVBert	s	4.3	2.0	7.2	3.8	6.5	4.0	4.6	–
ETPNav	s	3.4	2.5	4.8	3.3	9.0	2.5	4.3	–
NaVILA	s	3.4	2.5	5.3	3.8	8.0	13.0	6.0	–
NaVid	s	23.1	7.0	22.7	24.1	33.0	10.0	20.0	–
StreamVLN	s	23.1	8.0	21.3	24.5	28.0	10.0	19.2	–
SNav-7B, published	s	28.8	12.4	23.7	32.5	41.5	17.0	26.0	–
Our runs, identical harness and success criterion									
SNav-7B, local control	RGB	26.4	11.4	28.5	38.7	47.5	13.0	27.6	–
+ PNav	RGB	26.4	**54.2**	28.5	38.7	47.5	**19.0**	**35.7**	**+8.1**

Table 2: Paired outcomes by independently regenerated offline information labels under the 54.2% Precise configuration (SR %).

Slice	n	Control	PNav	Diff.	Reading
In-goal or metric-match total	252	16.3	**56.0**	**+39.7**	Descriptive total
metric endpoint-compatible	146	11.6	**71.2**	**+59.6**	Goal match offline
RETURN, start in goal	16	6.2	81.2	+75.0	Return unverified
other trivial start	90	25.6	26.7	+1.1	Start in goal
other grounded	769	31.2	30.9	−0.3	Not trace identity
frame-relative	207	28.5	28.5	0.0	No object origin is established

The in-goal or metric-match total adds two groups that start inside the goal and must not be interpreted as a program execution effect. The other-grounded row contains outcome changes in both directions, so its aggregate difference near zero is not a trace identity result. These offline labels cannot separate parsing from execution and are never passed to the wrapper at runtime.

Table 3: NavSpace component and auxiliary closure tests (SR %)

Configuration	Struct.	Prec.
40-episode diagnostics		
Frozen SNav	7.5	17.5
+ prompt dead reckoning	7.5	–
+ RETURN only	25.0	–
+ METRIC only, no LM	–	37.5
RETURN subset (n=24)		
Strict semantic gate	4.2	–
+ PNav homing	**75.0**	–
+ same-parser retrace	66.7	–
Auxiliary $c = 1$ cohorts		
SNav ($n = 21/36$)	19.0	16.7
+ PNav, $c = 1$ gate	**76.2**	**75.0**

4.2.3 Failure Attribution, Trace Identity

Within the metric endpoint-compatible slice, remaining failures may reflect numeric parsing, action-grid quantization, blocked-motion drift, register alignment or stopping time. The current result export does not distinguish their shares. Appendix A.2.2–A.2.3 gives the geometric bounds on metric endpoint error and blocked-motion drift. For a deterministic base policy, episodes without takeover keep the base trajectory because the wrapper sends $\pi_\theta(h_t)$ at each step (Appendix A.5). This conditional property is distinct from the aggregate rates. The positive rows measure outcomes where the complete wrapper can act.

4.2.4 Component and Contract Ablations

Prompt dead reckoning does not become persistent program state. The executor makes a parsed program actionable on the action grid. On parser-selected RETURN instructions, the strict semantic gate never takes over, so its frozen rate cannot evaluate homing. Table 3 separates arbitration from execution across its RETURN and closure tests.

In Table 3, RETURN uses 24 parser-selected episodes. The auxiliary cohorts use 21 Structure and 36 Precise episodes from this same experiment. The 40-episode diagnostics test prompt dead reckoning, RETURN execution and metric execution without the language model on small slices. Their samples and interventions differ, so their rates are not successive contributions to the full-axis result. The RETURN rows compare homing with a same-parser retrace after the parser exposes an origin relation. Together, these tests locate the effect of execution on the reported slices while leaving parser accuracy and the shares of individual failure causes unresolved.

Table 4: Backbone and parser–executor tests (SR %)

Configuration	Struct.	Prec.
SNav control	13.0	11.4
+ PNav	**19.0**	**54.2**
StreamVLN control	12.0	3.0
+ PNav	**16.0**	**54.2**
LLaVA-Video control	17.5	1.5
+ PNav	**17.5**	**53.2**
OctoNav PointNav (SR %)		
Frozen StreamVLN		5.7
+ coordinate parse only		5.7
+ parse and executor		**19.7**

Across the full NavSpace axes, Table 4 shows the pattern with StreamVLN (Wei et al., 2026) as a second backbone tuned for VLN. The untuned LLaVA-Video row (Zhang et al., 2025b) checks the same executor on an additional backbone. The NavSpace rows use 200 Structure and 201 Precise episodes. The OctoNav rows (Gao et al., 2026) use 300 paired episodes and hold the episode set and StreamVLN observations fixed while coordinate parsing and execution are enabled in stages. Parsing alone changes no outcome, but executing the parse does. This isolates that execution contribution.

4.2.5 Generality Across Environments and Tasks

The same compiler and executor apply beyond NavSpace templates. Table 5 keeps slot frequencies, offline labels and measured outcomes separate. The comparisons use parser-triggered subsets from R2R-CE (Anderson et al., 2018; Krantz et al., 2020) and English RxR-CE (Ku et al., 2020; Krantz et al., 2020), target-instance episodes from REVERIE-CE (Qi et al., 2020; Li et al., 2022), and coordinate episodes from OctoNav (Gao et al., 2026). Observation contracts differ across these environments. Where the instruction states a program, conditional execution raises success on each measured subset. The NavSpace pair remains the primary RGB evidence.

To probe reliance on NavSpace wording, we froze 26 hand-authored paraphrases outside the benchmark templates before running the parser: 18 closed metric programs and eight metric prefixes followed by a grounded clause. The unmodified parser recovered the exact ordered action-grid program on 72.2% of closed paraphrases (13 of 18) and rejected closure on 100% of mixed instructions. These parser diagnostics show that the same grammar covers most reworded closed programs and withholds takeover on mixed instructions (Appendix D).

A separate held-out set uses human semantic labels rather than NavSpace templates. Recognition takeover also fires on mixed and grounding-required instructions, whereas the semantic gate rejects most of them and preserves the frozen outcomes on those classes. On fully executable instructions, semantic routing improves over the frozen policy. On mixed and grounded instructions, the ungated recognizer loses successes. This pattern on new wording supports conditional routing rather than a template-specific macro gain. Per-class counts are in Appendix D.

4.3 Negative Evidence from Interfaces That Do Not Close the Gap

The positive results ask whether execution can be repaired once a program is given. Here we test whether permitted side modules can recover a goal condition when it is not. A privileged-information reference first measures success when the hidden goal is supplied. Training, geometry, readout, control and relational tests then ask whether any tested interface supplies it without reading the hidden goal. A negative outcome therefore locates a missing signal in the tested interface rather than an absence of target information in the scene (Appendix E). The conclusion is therefore limited to the interfaces, probes and decision thresholds that the four subsections below test in turn.

4.3.1 Privileged Reference and Training Diagnostic

The privileged policy is a performance reference, not a deployable method. It reads the hidden goal g, which the RGB arms never receive, so the comparison shows what that extra information yields

Table 5: Compiler signals and closed-loop outcomes across corpora. NavSpace signals are recorded slot labels. Other corpora use parser screening. Outcomes are reported only where runs are available.

Corpus or setting	Total	Compiler signal	Offline condition	Closed-loop outcome
NavSpace Precise	201	200 METRIC slots (99.5%)	146 metric endpoint-compatible	11.6 to **71.2** on that label and 11.4 to **54.2** axis-wide
NavSpace Structure	200	117 STRUCT slots (58.5%)	Slot is not a verified RETURN subset	13.0 to **19.0** at axis level
Four remaining NavSpace axes	827	–	Mostly grounded or frame-relative	Axis-level SR unchanged, without an identity test
REVERIE-CE val-unseen	3,189	1,625 (51.0%)	Reference to target instances	Frozen 9.2% vs PNav **14.2%**
OctoNav PointNav	300	300 (100.0%)	Close relative of coordinate programs	Frozen 5.7% vs PNav **19.7%**
R2R-CE val-unseen	1,839	96 parser triggers (5.22%)	Closure not independently verified on this subset	Frozen 75.0% vs PNav **81.3%**
RxR-CE val-unseen	3,669	123 (3.35%)	74 metric, 49 homing	Frozen 10.6% vs PNav **13.0%**
Held-out human labels	47	E 15, M 18, G 14	Human class, not a parser flag	EGO 40.4, recognition 46.8, semantic routing **53.2**

rather than an upper bound on a legal goal condition. The student variants in Table 7 remain far below this reference. Their teacher and capacity choices do not reproduce access to the hidden goal. Proposition A.2 (Appendix A.3) explains why: when student histories coincide but hidden goals demand different actions, behaviour cloning converges to a posterior mixture of expert actions, and added student capacity only moves it closer to that mixture rather than to the goal-specific action.

Table 6: Privileged-information reference on the four non-taken-over axes

Axis	n	Priv.	Ctrl.	Gap
Viewpoint shifting	207	84.5	28.5	+56.0
Environment state	200	95.0	47.5	+47.5
Spatial relationship	212	90.1	38.7	+51.4
Vertical perception	208	77.9	26.4	+51.4
All four axes	**827**	**86.8**	**35.2**	**+51.6**

Table 7: Training diagnostics on the Structure subset (SR %)

Student	Teacher	Struct.	Reading
Tiny CNN	Priv. actions	5.0 to 18.0	Low
Frozen encoder	Geodesic	18.0	Low
7B LoRA	Waypoint	5.0 to 28.0	Unstable
GoalStream	Priv. labels	**25.0**	Best student
Priv. teacher	Hidden goal	**80.0**	Reference

4.3.2 Readout Probes

Depth, odometry and local occupancy can improve geometric state, but target identity and arrival still need a usable readout. Table 8 tests bearing, instance identity, distance, visibility and gating. None reliably supports target-conditioned control. Bearing stays near the three-way chance rate, and the distance error stays close to a constant predictor. The partial signals motivate stronger instance binding, memory and viewpoint selection without isolating a single missing module.

4.3.3 Closed-Loop Control

Yaw and turn-dead-band fixes improved the oracle servo, but frozen readouts reached only 16.7% success versus 15.0% for control. Bearing, range, visibility and stopping errors limited this result. A hidden-goal controller succeeded separately. Those servo fixes change a controller that already sees the goal, while the frozen readout still lacks that target signal. The measured stopping bound remained far below the gap to the privileged reference across four axes.

4.3.4 Relational Language and Failure Shapes

Space Structure has no phrase that combines a number with a unit (0 of 200), while relational quantifiers occur in 178 of 200. These terms describe relations among scene candidates rather than fixed quantities of self motion. Appendix Table 12 tests four reductions to event programs. An event count, region entry or distance ranking does not provide a stable arrival program: an ordinal ranks scene candidates, while a crossing count supplies neither their identity nor a path budget. With the geometry and readout results, this leaves target binding and control selection unresolved.

Table 8: Grounded readouts and gates. Chance is 33.3% for bearing and 50% for instance. FC denotes false commit.

Quantity	Probe	Reading	Interpretation
Bearing	Frozen VLM, 3-way choice	33.7%	Near chance
Instance	Two-view forced choice	53.0%	Near chance
Distance	Learned regression, MAE	2.36 m	Constant predictor: 2.45 m
Visibility	Learned readouts, AUC	0.50–0.56	Weak discrimination
Identity gate	Entropy/cov., named sofa	0% SR, FC 100%	Confidently wrong
Second view	Same-track agreement	FC 64.6%	Distractor not rejected
Active gate	Unseen referent, chosen view	72.9% SR, FC 0	Hands over after rejection
Weak attribute	Attribute accuracy 0.55	1.3% SR, 98.3% abstain	Gate adds no missing signal

5 Discussion

5.1 Choosing the Interface Before the Model

Evaluation should begin with information requirements. When an instruction gives a self-motion program, first check action integration, quantization, collision consistency and program completeness before adding visual capacity. When the target must be discovered in the scene, the evaluation should state how target identity, bearing, distance or an arrival event enters the interface. Mixing them hides executable repair and can mistake missing goal information for tuning.

The distinction also guides placement: the frozen policy handles episodes outside takeover, while an external executor handles recognized metric and origin programs. This supports conditional placement, not universal external control. Weaker in-model placements (Appendix Table 13) show that writing state or program text does not create persistent execution state. The privileged reference and the training, geometry and readout probes distinguish capacity from interface gaps: larger policies alone do not expose the missing target condition. Details appear in Appendix G.

5.2 Generalization and the Boundary of Delegation

The same executor applies where a motion program is available, the parser covers the wording, takeover is eligible and a base policy remains in place. StreamVLN tests a second backbone. Corpus screening records how often a metric prefix appears, and the paired subsets measure execution on those instructions. Held-out paraphrases recover the exact program on 72.2% of closed rewordings, while human labels separate executable, mixed and grounding-required instructions. Across these settings, takeover follows a stated motion program, and a grounded remainder stays with the frozen policy. A broader grammar would extend the same executor to more instructions.

For grounding-dependent instructions, geometric execution becomes useful only after the interface exposes a sufficiently reliable target condition. Table 8 and Appendix Table 14 test one such condition in a controlled sensor model: low entropy and low position covariance do not earn the handover, and neither does a second view of the track already in hand. The handover holds when an unseen referent is still in the belief and the next view is chosen to test it. NavSpace readouts remain near chance, so this probe supplies no usable target signal there.

6 Conclusion

VLN failures need not share a single cause. We first show that when an instruction already specifies an executable self-motion program, conditional execution can repair the execution gap: across 1,228 paired NavSpace episodes, PNav raises six-axis macro success from 27.6% to 35.7% and Precise from 11.4% to 54.2%, while preserving frozen SNav-7B trajectories on episodes outside takeover. We then examine what remains when motion is not fully specified. Training, geometry, readout, and control probes stay far below a privileged-information reference and do not recover a reliable goal condition. Together, these results suggest matching the interface to the information available: externalize stateful execution for stated motion programs, while grounded tasks require dependable target identity and arrival signals before delegation.

AI USE STATEMENT

Generative AI tools were used to aid and polish the writing of this paper, including edits to grammar, clarity, and phrasing. The authors reviewed the suggested edits, verified the final text, citations, and reported results, and take responsibility for the final content.

REPRODUCIBILITY STATEMENT

The method is specified in Section 3 and the appendices, the frozen grammar and offline verifier are described in Appendix B, and the evaluation protocol and statistics are described in Appendix C. The runtime signals and constants read by the executor and verifier are tabulated in Appendix C.3. Appendix D distinguishes completed local diagnostics from measurements outside the scope of the reported claims.

ETHICS STATEMENT

This work studies navigation policies in simulated indoor environments and uses publicly released benchmarks and checkpoints. The proposed intervention reads only observations that a deployed RGB navigation system could obtain, and it does not collect human data or introduce a new dataset. The negative analysis documents cases in which a plausible side module cannot recover goal information, which we report so that later work does not repeat the same measurement errors.

REFERENCES

Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. Etpnav: Evolving topological planning for vision-language navigation in continuous environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(7):5130–5145, 2025. doi: 10.1109/TPAMI.2024.3386695.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. 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*, pp. 3674–3683, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Anderson_Vision-and-Language_Navigation_Interpreting_CVPR_2018_paper.html.

Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*, 2020. URL https://arxiv.org/abs/2006.13171.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Dorsa Sadigh, Leonidas J. Guibas, and Fei Xia. SpatialVLM: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14455–14465, 2024a.

Jiaqi Chen, Bingqian Lin, Ran Xu, Zhenhua Chai, Xiaodan Liang, and Kwan-Yee K. Wong. Mapgpt: Map-guided prompting with adaptive path planning for vision-and-language navigation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 9796–9810, 2024b. URL https://aclanthology.org/2024.acl-long.529/.

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*, 2025. URL https://navila-bot.github.io/.

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

Hongyan Feng, Sunlai Chen, Xuanyu Liu, Miao Pan, Yangfan Xie, Yuxiang Cui, Zhongxiang Zhou, Rong Xiong, Wenqi Zhang, Jianwei Yin, Yueting Zhuang, and Xuhong Zhang. Embodied-navigator: Point, think, memorize, and align for efficient navigation. *arXiv preprint arXiv:2608.17512*, 2026. URL https://arxiv.org/abs/2608.17512.

Chen Gao, Liankai Jin, Xingyu Peng, Jiazhao Zhang, Yue Deng, Annan Li, He Wang, and Si Liu. OctoNav: Towards generalist embodied navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 40074–40084, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Gao_OctoNav_Towards_Generalist_Embodied_Navigation_CVPR_2026_paper.html.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alex Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Fu. Do As I Can, Not As I Say: Grounding language in robotic affordances. In *CoRL*, 2022. URL https://openreview.net/forum?id=bdHkMjBJG_w.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the navgraph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision*, pp. 104–120, 2020.

Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 4392–4412, 2020. doi: 10.18653/v1/2020.emnlp-main.356. URL https://aclanthology.org/2020.emnlp-main.356/.

Xinghang Li, Di Guo, Huaping Liu, and Fuchun Sun. REVE-CE: Remote embodied visual referring expression in continuous environment. *IEEE Robotics and Automation Letters*, 7(2):1494–1501, 2022. doi: 10.1109/LRA.2022.3141150.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation*, pp. 9493–9500, 2023. doi: 10.1109/ICRA48891.2023.10160591. URL https://arxiv.org/abs/2209.07753.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. doi: 10.1007/BF02295996.

Robert G. Newcombe. Improved confidence intervals for the difference between binomial proportions based on paired data. *Statistics in Medicine*, 17(22):2635–2650, 1998. doi: 10.1002/(SICI)1097-0258(19981130)17:22⟨2635::AID-SIM954⟩3.0.CO;2-C.

Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. REVERIE: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9982–9991, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Qi_REVERIE_Remote_Embodied_Visual_Referring_Expression_in_Real_Indoor_Environments_CVPR_2020_paper.html.

Tin Stribor Sohn, Maximilian Dillitzer, Jason J. Corso, and Eric Sax. Embodied4c: Measuring what matters for embodied vision-language navigation. *arXiv preprint arXiv:2512.18028*, 2025. URL https://arxiv.org/abs/2512.18028.

Aaron Walsman, Muru Zhang, Sanjiban Choudhury, Dieter Fox, and Ali Farhadi. Impossibly good experts and how to follow them. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sciA_xgYofB.

Zehao Wang, Minye Wu, Yixin Cao, Yubo Ma, Meiqi Chen, and Tinne Tuytelaars. Navigating the nuances: A fine-grained evaluation of vision-language navigation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 4681–4704. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.269. URL https://aclanthology.org/2024.findings-emnlp.269/.

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. In *IEEE International Conference on Robotics and Automation*, 2026. URL https://arxiv.org/abs/2507.05240.

Zihao Xin, Wentong Li, Yixuan Jiang, Ziyuan Huang, Bin Wang, Piji Li, Jianke Zhu, Jie Qin, and Shengjun Huang. Agentvln: Towards agentic vision-and-language navigation. *arXiv preprint arXiv:2603.17670*, 2026. URL https://arxiv.org/abs/2603.17670.

Haolin Yang, Yuxing Long, Zhuoyuan Yu, Zihan Yang, Minghan Wang, Jiapeng Xu, Yihan Wang, Ziyan Yu, Wenzhe Cai, Lei Kang, and Hao Dong. Navspace: How navigation agents follow spatial intelligence instructions. In *IEEE International Conference on Robotics and Automation*, 2026. URL https://arxiv.org/abs/2510.08173.

Jihan Yang, Shusheng Yang, Anjali W. Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10632–10643, 2025.

Jianshu Zhang, Yijiang Li, Huifeixin Chen, Haoran Lu, Letian Xue, Bingyang Wang, and Han Liu. Spacenum: Revisiting spatial numerical understanding in vlms. *arXiv preprint arXiv:2605.23898*, 2026a. URL https://arxiv.org/abs/2605.23898.

Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and He Wang. Navid: Video-based vlm plans the next step for vision-and-language navigation. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.079.

Jiazhao Zhang, Kunyu Wang, Shaoan Wang, Minghan Li, Haoran Liu, Songlin Wei, Zhongyuan Wang, Zhizheng Zhang, and He Wang. Uni-navid: A video-based vision-language-action model for unifying embodied navigation tasks. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025a. doi: 10.15607/RSS.2025.XXI.013.

Shaoheng Zhang, Zhichen Li, Guangfu Ma, and Jie Mei. Dart-vln: Test-time memory decay and anti-loop regularization for discrete vision-language navigation. *arXiv preprint arXiv:2607.01043*, 2026b. URL https://arxiv.org/abs/2607.01043v4.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. LLaVA-Video: Video instruction tuning with synthetic data. *Transactions on Machine Learning Research*, 2025b. URL https://openreview.net/forum?id=EElFGvt39K.

Gengze Zhou, Yicong Hong, Zun Wang, Xin Eric Wang, and Qi Wu. Navgpt-2: Unleashing navigational reasoning capability for large vision-language models. In *European Conference on Computer Vision*, 2024. URL https://arxiv.org/abs/2407.12366.

A Mathematical Definitions and Proofs

Section 3.3 hands control to the executor only when a program is active and still fits in the remaining budget,

$$\alpha_t = \mathbf{1}[P \neq \perp, \text{ program fits in } T_{\max} - t]\,. \tag{5}$$

$P = \perp$ denotes rejection.

A.1 How the Executor Places a Command on the Grid

Each planned quantity is snapped to the action grid once at parse time, and the executor then works in whole ticks.

$$\widetilde{d}_i = \Delta\,\mathrm{round}(d_i/\Delta), \qquad \widetilde{\theta}_i = \delta\,\mathrm{round}(\theta_i/\delta).$$

For a single quantity this costs at most half a tick per segment, $|\widetilde{d}_i - d_i| \le \Delta/2$ and $|\widetilde{\theta}_i - \theta_i| \le \delta/2$. Each command is snapped once, and the executor then applies those snapped increments, so the heading at segment i is $\sum_{j\le i}\widetilde{\theta}_j$ and the travel is $\sum_{j\le i}\widetilde{d}_j$. No second rounding is applied to the cumulative heading. On the action grid, where $d_i \in \Delta\mathbb{N}$ and $\theta_i \in \delta\mathbb{Z}$, the snap is the identity and the executed program equals the ideal one. Appendix A.2.2 bounds the accumulated heading error by $|e_i| \le i\delta/2$. Section 4.2.3 lists quantization among the possible causes of the remaining endpoint error and states that the export does not separate their shares, so this half-tick snap is not a measured residue.

A.2 Proposition 1 on Executability and Conservative Error Facts

Proposition A.1. Under ideal kinematics the following three conservative facts hold.

1. If the goal is an origin o established from the runtime trajectory, the RETURN controller uses the origin register to define its return direction. No finite-step arrival or safe-budget guarantee is asserted here.
2. If a METRIC program lies on the action grid, the executed endpoint equals the ideal endpoint. Off the grid, each commanded distance and turn differs from its grid value by at most half a tick. We do not assert a universal cumulative endpoint bound here.
3. If an open-loop register mistakes k blocked forward moves for real forward motion, the drift between the register and the true pose is at most $k\Delta$.

A.2.1 Geometry of a RETURN step

Let q_t be the displacement from the established origin and let $r_t = \|q_t\|$. Let w_t be the unit vector along the current heading, and β_t the signed angle from that heading to the origin. After a forward step of length Δ,

$$q_{t+1} = q_t + \Delta w_t, \qquad r_{t+1}^2 = r_t^2 - 2r_t\Delta\cos\beta_t + \Delta^2,$$

and the new bearing satisfies

$$\sin\beta_{t+1} = \frac{r_t\sin\beta_t}{r_{t+1}}, \qquad \cos\beta_{t+1} = \frac{r_t\cos\beta_t - \Delta}{r_{t+1}}.$$

These relations do not imply that the bearing error contracts after a forward move. With $r_t = 1$ m, $\Delta = 0.25$ m and $|\beta_t| = 15^\circ$,

$$r_{t+1} = \sqrt{1 - 0.5\cos 15^\circ + 0.25^2} \approx 0.761 \text{ m}, \qquad |\beta_{t+1}| = \mathrm{atan2}(\sin 15^\circ, \cos 15^\circ - 0.25) \approx 19.9^\circ.$$

Thus a forward move can increase the bearing error even when the pre-move error is small. The controller described in Section 3.4 uses a finite horizon as an experimental budget, but these equations do not yield a universal finite-step or safe-budget theorem. It reads the origin register alone.

A.2.2 Endpoint Error of METRIC

Write $\Psi_i = \sum_{j=1}^{i} \theta_j$ and $\widetilde{\Psi}_i = \sum_{j=1}^{i} \widetilde{\theta}_j$ for the ideal and snapped headings of segment i in the initial register frame. The ideal and executed displacements from the start are

$$p^\star = \sum_{i=1}^{n} d_i u(\Psi_i), \qquad \hat{p}_T = \sum_{i=1}^{n} \widetilde{d}_i u(\widetilde{\Psi}_i).$$

If $d_i \in \Delta\mathbb{N}$ and $\Psi_i \in \delta\mathbb{Z}$ the snap is the identity and hence $\hat{p}_T = p^\star$. Off the grid, each segment still satisfies $|\widetilde{d}_i - d_i| \le \Delta/2$ and $|\widetilde{\theta}_i - \theta_i| \le \delta/2$. Let $e_i = \widetilde{\Psi}_i - \Psi_i$ be the cumulative heading error at segment i (with headings represented on the real line before wrapping). The exact unit-vector identity gives the valid per-segment bound

$$\left\| \widetilde{d}_i u(\widetilde{\Psi}_i) - d_i u(\Psi_i) \right\| \le |\widetilde{d}_i - d_i| + 2d_i \left| \sin\!\left(\frac{e_i}{2}\right) \right|, \qquad |e_i| \le \frac{i\delta}{2}.$$

Consequently,

$$\|\hat{p}_T - p^\star\| \le \frac{n\Delta}{2} + 2\sum_{i=1}^{n} d_i \left| \sin\!\left(\frac{e_i}{2}\right) \right|.$$

We do not replace this expression by a single n-dependent sine term, because $|\sin(\cdot)|$ is not globally monotone. The experiments therefore use the action-grid fact and report measured endpoint errors rather than a universal simplified endpoint guarantee. □

A.2.3 Drift from Blocked Forward Moves

One blocked F leaves the true pose in place while a register without collision correction advances by Δ along the current heading. The sum of the difference vectors produced by k blocked moves satisfies

$$\|p_T - \hat{p}_T\| \le \sum_{j=1}^{k} \Delta = k\Delta.$$

If $\|\hat{p}_T - g\| \le \rho - k\Delta$, the triangle inequality gives

$$\|p_T - g\| \le \|p_T - \hat{p}_T\| + \|\hat{p}_T - g\| \le \rho.$$

The magnitude is attained when the k collisions occur along the same direction, so $k\Delta$ is a tight worst-case bound. □

A.3 Proposition 2 on Privileged Imitation under a Hidden Goal

Proposition A.2. Suppose two episodes e and e' share the same student history $h_\tau = h'_\tau$ at decision time τ, while their hidden goals require mutually exclusive actions $a^\star \neq a'^\star$ that cannot be recovered once the wrong one is taken. Any student that depends on h_τ alone succeeds on average at most half the time on a symmetric two-way branch and at most $1/K$ on a symmetric K-way branch. Under a proper scoring loss the population optimum of behaviour cloning is the conditional expectation of the privileged expert action given the student history.

Proof. Because $h_\tau = h'_\tau$, the student emits the same distribution on both episodes. Let $q = \pi(a^\star \mid h_\tau)$. The success probability on e is at most q, while on e' we have $\pi(a'^\star \mid h_\tau) \le 1 - q$, so the average is at most $1/2$. Summing over K mutually exclusive actions gives at most $1/K$ in the same way.

Let the privileged expert be $\pi^\star(\cdot \mid s_\tau, g)$ and let the student risk be

$$R(\pi) = \mathbb{E}_{h_\tau, g}\left[\ell\big(\pi(\cdot \mid h_\tau), \pi^\star(\cdot \mid s_\tau, g)\big)\right].$$

For a proper scoring loss such as cross entropy the pointwise minimizer is

$$\pi^{\mathrm{BC}}(\cdot \mid h_\tau) = \mathbb{E}\left[\pi^\star(\cdot \mid s_\tau, g) \mid h_\tau\right].$$

If two hidden goals occur with probability $1/2$ each under the same history and demand $a^\star$ and $a'^\star$ respectively, the optimal student assigns $1/2$ to each action. Enlarging student capacity only brings it closer to that posterior mixture and cannot create the missing goal condition. □

A.4 Proposition 3 on Decomposing the Conditional Effect and the Operable Skyline

Proposition A.3. Let the sample space decompose into mutually exclusive slices $\Omega = \bigsqcup_k \Omega_k$ with $p_k = \Pr(\Omega_k)$. The law of total probability gives

$$\mathrm{SR} = \sum_k p_k \mathrm{SR}_k.$$

Let $\mathcal{K}_{\mathrm{take}}$ index the slices in which the wrapper actually overrides a base action, SR_k^0 be the base success rate in slice k, and σ_k its wrapped success rate. If the executor changes outcomes only on these slices, then

$$\mathrm{SR} - \mathrm{SR}^0 = \sum_{k \in \mathcal{K}_{\mathrm{take}}} p_k(\sigma_k - \mathrm{SR}_k^0).$$

Since $0 \le \sigma_k, \mathrm{SR}_k^0 \le 1$ we obtain

$$|\mathrm{SR} - \mathrm{SR}^0| \le \sum_{k \in \mathcal{K}_{\mathrm{take}}} p_k.$$

The right-hand side is the operable density skyline of the given takeover condition. It explains why a large macro gain is observable on NavSpace Precise while R2R-CE, with a 5.22% parser-triggered subset, should not be evaluated by RGB PNav macro SR alone. Start-in-goal labels are excluded from causal attribution to an executable program. □

A.5 Proposition 4 on Trace Identity for Episodes without Takeover

Proposition A.4. Assume that the base policy is deterministic given its history, that both arms start from the same initial state and the same random seed, and that the wrapper never takes over on the episode. Whenever it does not take over it sends $\pi_\theta(h_t)$ verbatim. Then the wrapped system and the base policy produce identical actions, observations, internal caches, paths and success outcomes.

Proof. We induct on time t. At $t = 0$ the two histories agree. Assume $h_t = h'_t$. Since the base policy is deterministic, $\pi_\theta(h_t) = \pi_\theta(h'_t)$. The wrapper sends that action because it does not take over, so both arms send the same a_t. The environment produces the same next state and observation under the same state, seed and action, and the policy cache is updated from the same input, hence $h_{t+1} = h'_{t+1}$. Inducting to termination gives identical trajectories and identical success indicators. □

If the wrapper takes over on any step, the history after takeover may differ from the base policy, so the proposition does not apply. This is why identity on episodes without takeover cannot be widened into global no-harm.

A.6 Proposition 5 on Coordinate Invariance of relabel

Proposition A.5. Let the local pose under an origin $O_j = (p_j, \psi_j)$ be

$$r_t^{(j)} = \Big(R(-\psi_j)(\hat{p}_t - p_j), \mathrm{wrap}(\hat{\psi}_t - \psi_j) \Big) .$$

At time τ we execute `relabel(j→k)` with $O_k = (\hat{p}_\tau, \hat{\psi}_\tau)$. For any later estimated pose $\hat{s}_t$, the global position recovered from the k coordinates is

$$p_k + R(\psi_k)R(-\psi_k)(\hat{p}_t - p_k) = \hat{p}_t.$$

Hence `relabel` changes the representation alone and leaves the global estimated trajectory determined by the same action history unchanged. It can clear the local offset of the previous clause, but it can neither reduce the drift of $\hat{p}_t$ itself with respect to the true pose nor create goal information. □

A.7 The Return Controller in Full

Section 3.4 states the return rule in words. The following case form describes the reported RGB controller, with β_t denoting the error that points back to the established origin.

$$\pi_{\mathrm{home}}(\hat{s}_t) = \begin{cases} \mathsf{L}, & \beta_t > \delta/2, \\ \mathsf{R}, & \beta_t < -\delta/2, \\ \mathsf{F}, & \|\hat{p}_t\| > \rho_{\mathrm{stop}} \ \wedge \ |\beta_t| \le \delta/2, \\ \mathsf{S}, & \|\hat{p}_t\| \le \rho_{\mathrm{stop}} \ \wedge \ |\beta_t| \le \delta/2. \end{cases}$$

Alignment is checked first. A bearing outside $\pm\delta/2$ produces a turn even when the register is already inside ρ_{stop}, and stop is emitted only when the bearing is inside that band and $\|\hat{p}_t\| \le \rho_{\mathrm{stop}}$. This is the rule stated in Section 3.4. The value ρ_{stop} is an experimental stopping margin chosen inside the success radius ρ. The geometric relations above do not assert a universal guarantee for endpoint safety or a finite budget.

B Grammar, Diagnostic Closure and Offline Labels

B.1 Diagnostic Closure

Removing the matched program span leaves L_{res}, and the closure bit is

$$c(L) = \mathbf{1}\left[L_{\mathrm{res}} \in \{\epsilon\} \cup \{\text{contains a stop instruction only}\} \right].$$

The compiler uses the frozen grammar released with the code. A residual that still names a scene, a target or a perceptual condition leaves the bit unset. This bit is a diagnostic definition.

B.2 Offline Information Labels

Section 3.2 uses three nested diagnostic layers. With κ the ideal-kinematics execution of P,

$$L_1 = \mathbf{1}[P \neq \bot], \qquad L_2 = \mathbf{1}[L_1 = 1 \wedge \text{the residual is closed}], \qquad L_3 = \mathbf{1}[L_2 = 1 \ \wedge \ \|\kappa(P; s_0) - g\| \le \rho] , \tag{6}$$

so $L_3 \Rightarrow L_2 \Rightarrow L_1$. Here L_1 records parser recognition, L_2 records compiler-estimated residual closure, and L_3 checks the ideal endpoint against g offline. Semantic closure still requires an independent assessment of whether the whole instruction is executable. The parser's c flag is only an estimate. This three-layer record is diagnostic. The current result export does not contain all of P, c and L_3 per episode, so the conditional table in the main paper does not report an L_3 population.

Its reproducible labels come from `tools/class_audit.py`, which reads the episode metadata offline.

1. Mark frame-relative instructions first, because their object-defined origin is unavailable from self-motion alone.
2. Otherwise mark episodes with RETURN language whose start is within ρ of the hidden goal. Mark other episodes already inside that radius as trivial starts. Both labels for starts inside the goal are descriptive and do not establish completion of intermediate instruction semantics.
3. For the remainder, execute the parsed metric plan under ideal kinematics from s_0 and mark metric endpoint-compatible if its endpoint lies within ρ of g. Mark all other episodes as the grounded remainder.

The ordering matters. The 16 cases with RETURN language and 90 other trivial starts are separate rows, while 146 remaining episodes have metric endpoints that match the hidden goal. The offline classifier uses $\leq \rho$ and the benchmark success check uses $< \rho$. None of the 1,228 current label distances is exactly on the radius, so this boundary convention changes no reported label. The labels were regenerated with the current metric parser and paired to the 54.2% raw Precise outcomes by episode key.

The classifier is offline only. It may read the hidden goal to group outcomes after evaluation, while the wrapper receives neither the goal nor these information labels. The runtime parser acts only on the instruction and its own register state.

C Experimental Protocol and Statistical Details

C.1 The PNav Column

- The frozen local control is the official `TidalYang/SNav-7B` (Yang et al., 2026) with no override, and a published number is never used as the paired baseline of an experiment we ran ourselves. Published leaderboard numbers are quoted as context only, and no subtraction is taken across the two blocks.
- PNav is the same checkpoint with the fixed RGB parser and origin register of Section 3 attached. The 54.2% Precise run disables `residual_handoff`, clause `relabel`, `R_bump` and closure sidestep.
- Both local arms use RGB observations and the same action interface. PNav integrates emitted actions and the collision bit in its origin register.
- The history holds 16 frames, at most 4 actions are taken per step, decoding is greedy and the seed is fixed.
- The horizon is $T_{\max} = 70$ on NavSpace.
- Simulated depth, the true pose, maps, the hidden goal, navmesh-to-goal, undeclared prompt injection and STOP vetoes are all forbidden.
- Main-paper Tables 1 and 2 use this RGB pair. Geometry, privileged-oracle and cross-corpus diagnostics use their own stated observation contracts and are not pooled into the NavSpace paired comparison.

For the additional corpora, R2R-CE (Anderson et al., 2018; Krantz et al., 2020), RxR-CE (Ku et al., 2020; Krantz et al., 2020) and REVERIE-CE (Qi et al., 2020; Li et al., 2022) follow their respective val-unseen protocols. The horizon, success radius and metrics are locked before a run and written into the raw configuration.

C.2 The Benchmark Action Space and the Four-token Model Protocol

Yang et al. (2026) state the NavSpace evaluation action set directly. At each step the agent selects one action: move forward 0.25 m, turn left 30°, turn right 30°, or stop. The same paper says the controller has no backward primitive, so a walk-backward instruction is carried out by turning

180° and moving forward, and its chance-level random baseline is uniform over these four actions. Section 3.1 uses this set, $\mathcal{A} = \{\mathsf{F}, \mathsf{L}, \mathsf{R}, \mathsf{S}\}$, with $\Delta = 0.25$ m and $\delta = 30^\circ$. The benchmark success test is positional, $\|p_T - g\| < \rho$.

The released simulator configuration registers a larger action map. Besides those four actions it includes a backward translation of Δ and two camera tilts of 30°, and stop ends the episode, so the map has seven names,

$$\mathcal{A}_{\text{bench}} = \{\mathsf{F}, \mathsf{L}, \mathsf{R}, \mathsf{B}, \mathsf{U}, \mathsf{D}, \mathsf{S}\}.$$

Actions F and B translate along and against the current heading. Actions L and R rotate in place. Actions U and D tilt the camera without moving the agent. The annotation interface that collected the reference trajectories exposes the same extras as `W`, `A`, `S`, `D` for forward, left, backward and right, together with `Q`/`E` for the two tilts. Reference `action_sequence` fields contain `look_up`, `look_down` and `backward`. This seven-name vocabulary belongs to the simulator and to the annotation interface. It is not the action set the NavSpace paper assigns to an evaluated policy.

Every policy route shipped with the release asks for the paper's four actions. Table 9 lists the three routes alongside the annotation interface. The SNav route names four actions in its prompt and parses the decoded string with a matcher whose tables contain only those four tokens. That matcher is replicated by `gold_parser` and used by both arms of every pair in this paper. The LLM route's prompt also lists four actions, while its matcher additionally accepts `move backward`, `look up` and `look down`. StreamVLN is prompted for four actions and its matcher emits only those four. No evaluation route therefore offers the seven-action map as the policy's action set. The published numbers in Table 1 are quoted from Yang et al. (2026), whose evaluation statement is this four-action controller. They are context. The release does not run every published system through `eval_snav.py`.

Two consequences follow for the tables of this paper. First, the local comparison is symmetric. The local control and PNav are decoded by one matcher with four tokens, the action set stated by Yang et al. (2026). Published numbers in the upper block of Table 1 are quoted from that paper as context. They are not all outputs of `eval_snav.py`, and no subtraction crosses the published block and the local block. Second, PNav's executor loses no reachability by staying inside the same subset, $\mathcal{A}_{\text{exec}} = \{\mathsf{F}, \mathsf{L}, \mathsf{R}, \mathsf{S}\} \subseteq \mathcal{A}_{\text{bench}}$. Success is positional, $\|p_T - g\| < \rho$, while U and D change only the camera pitch and never the agent position read by the criterion. No tilt can therefore move the graded quantity. Every action other than stop, including a tilt, still spends one step of the $T_{\max} = 70$ budget. A tilt is a cost rather than a free observation. It would also shift the input distribution of the frozen policy instead of extending its skill set because the released render parameters fix the camera height and use only 0.25 m translations and 30° rotations. Backward does not create a new state either. Six δ turns, one Δ step and six turns back reproduce one B exactly on the executor grid. This construction uses thirteen actions instead of one, so the subset gives up step efficiency inside the budget rather than reachability. The released reference trajectories do use the extra primitives, and `look_up`, `look_down` and `backward` occur inside their `action_sequence` fields. Since the criterion reads position rather than pitch, the reference position trajectory remains expressible in the subset with four tokens. No axis of the suite is excluded by construction.

C.3 Runtime Signals and Constants

Table 10 collects the quantities that executor and verifier read but that Section 3 and Appendix B state only symbolically.

The heading reference is a term the executor cannot remove by construction. The offset between the establishment frame and the true initial heading does not by itself displace a relative program, because a constant frame offset cancels between the commanded turn and the commanded distance. Any remaining discrepancy can come from a convention mismatch between the register angle and the instruction, from establishing the register at a moment other than the intended one, or from blocked-motion drift. Section 4.2.3 names numeric parsing, quantization, blocked-motion drift, register alignment and stopping time as possible causes, and states that the current export does not separate their shares. The heading offset sits inside that unresolved split. We do not report a measured decomposition.

Table 9: The model protocol with four tokens inside a simulator map of seven actions. Yang et al. (2026) state the four-action evaluation controller. All three policy routes shipped with the release ask for those four actions, and only the LLM route matcher accepts the other three. The seven-action vocabulary belongs to the simulator and the annotation interface. Both arms of every pair in this paper run under the first row. Published numbers in Table 1 are quoted from the paper, not produced by that script alone.

Route	**What the policy is told**	**Tokens its matcher can emit**
SNav, `eval_snav.py`	"predict the next n low-level actions from: FORWARD (↑) moves 25 cm, TURN LEFT (←) rotates 30°, TURN RIGHT (→) rotates 30°, STOP ends navigation"	$\{\mathsf{F}, \mathsf{L}, \mathsf{R}, \mathsf{S}\}$
LLM API, `run_llm_eval.py`	"You can only choose from the following four actions: Move forward, Turn left, Turn right, Stop"	$\{\mathsf{F}, \mathsf{L}, \mathsf{R}, \mathsf{B}, \mathsf{U}, \mathsf{D}, \mathsf{S}\}$
StreamVLN (Wei et al., 2026), `eval_streamvln.py`	"Devise an action sequence to follow the instruction using the four actions"	$\{\mathsf{F}, \mathsf{L}, \mathsf{R}, \mathsf{S}\}$
Annotation interface	keyboard `W`/`A`/`S`/`D` and `Q`/`E`, with no policy query	$\mathcal{A}_{\text{bench}}$

Table 10: Runtime signals and constants. The success radius is set per axis by the benchmark rather than by one global value. The executor stops inside a margin of that radius so that the half-tick quantization of Appendix A.1 stays inside the success test.

Quantity	**Value**	**Where it comes from**
Forward step Δ	0.25 m	Action quantization of the benchmark and of the executor alike
Turn step δ	30°	Same
Horizon $T_{\max}$	70 steps	Locked before the run in the raw configuration
ρ, vertical perception	3.0 m	Per-axis benchmark success radius
ρ, viewpoint, spatial relationship, environment state	2.0 m	Same
ρ, precise movement, space structure	1.0 m	The tightest radius in the suite. Appendix A.2.2 bounds the grid contribution to endpoint error. Section 4.2.3 does not assign that contribution a measured share
ρ_{stop}	$0.75\,\rho$, floored at Δ	Executor stopping margin, so that alignment and quantization error stay inside the radius
Collision bit b_t	simulator collision feedback	The shared action interface reports whether an issued forward step was blocked. PNav uses this feedback when updating the origin register. A deployed robot can obtain the same event through bumpers, motor current or an IMU. Appendix A.2.3 shows how ignoring a blocked step would add Δ of phantom travel to the register
Heading reference $\hat{\psi}_0$	0 at the first control step	The register is initialised at the opening frame, so the program's accumulated heading shares the frame of the action integration and no external compass or true pose is read. Any offset between the establishment frame and the true initial heading is carried by every subsequent turn and is not removed by the register
Success test	$\|p_T - g\| < \rho$	Benchmark criterion that the executor never evaluates at runtime

C.4 Paired Statistics

Let n_{10} be the episodes PNav wins alone and n_{01} those the local control wins alone, and let n be the number of paired episodes. The exact McNemar test (McNemar, 1947) is a binomial test on the $n_{10} + n_{01}$ discordant episodes under the null hypothesis $p = 1/2$.

Episode-level intervals for a paired success difference are Newcombe's method 10 (Newcombe, 1998). Write $\delta = (n_{10} - n_{01})/n$. Let (ℓ_{10}, u_{10}) and (ℓ_{01}, u_{01}) be ordinary Wilson score intervals

for the single proportions n_{10}/n and n_{01}/n. The paired interval is

$$\delta - \sqrt{\left(\frac{n_{10}}{n} - \ell_{10}\right)^2 + \left(u_{01} - \frac{n_{01}}{n}\right)^2},\ \delta + \sqrt{\left(u_{10} - \frac{n_{10}}{n}\right)^2 + \left(\frac{n_{01}}{n} - \ell_{01}\right)^2}.$$

The intervals $[6.3, 9.8]$ and $[50.8, 67.2]$ in the main text are this construction, in percentage points. They are not scene-cluster intervals.

A separate 20,000-replicate scene-cluster bootstrap, resampling scenes with replacement inside each axis over 276 scenes, gives a six-axis macro difference of 8.13 points with a percentile 95% interval of [6.40, 9.87]. Template identifiers are absent from the released episode records, so no template-cluster interval is reported. All summaries are produced from per-episode outputs rather than transcribed by hand.

D Supplementary Evidence Matrix

Table 11: Completed local diagnostics and limits of the empirical claims. A parser flag, an offline information label and a closed-loop outcome measure different quantities.

Check	Observed evidence and boundary	Use in claims
Metric-parser census	On all 1,228 NavSpace instructions, 392 yield a metric segment and 27 carry the diagnostic export's residual-complete flag. All 27 flagged rows are on Precise. Same compiler and experiment as the auxiliary cohorts in Table 3. This flag count is not their denominator.	Parser diagnostic only
Radius and action grid	Offline share of the in-goal or metric-match total is 20.5% at one radius and 51.5% at three radii. Across nine tested action grids it is 16.29–20.60%. No navigation run uses a changed radius or grid.	Label sensitivity only
Scene clustering	The 20,000-replicate scene bootstrap gives an all-episode macro difference of 8.13 points, 95% CI [6.40, 9.87]. Template IDs are unavailable.	Scene robustness only
Second backbone	StreamVLN has full and sliced control/PNav rows in the main-paper ablation table.	Tested backbone
Action integration noise	A controlled 200-program Monte Carlo varies forward, lateral and turn perturbations. It is neither calibrated sensor noise nor simulator SR.	Diagnostic stress test
Cross-corpus screening	R2R-CE and English RxR-CE parser-trigger densities are 5.22% and 3.35%. REVERIE-CE and OctoNav have measured paired subsets. R2R-CE has a 96-episode paired outcome, and the RxR-CE subset has 13/123 frozen versus 16/123 PNav successes.	Cross-corpus outcomes
Held-out paraphrases	In 26 frozen hand-authored sentences, exact action-grid parsing succeeded for 72.2% of closed programs (13 of 18) and the parser rejected closure for 100% of mixed grounded instructions. The set was not independently blinded.	Parser coverage diagnostic
Unmeasured extensions	Independent parser accuracy, blinded parser precision, template clustering, matched full-corpus evaluation, geo component removal and calibrated odometry noise were not measured.	No claim made

In the 24 parser-selected RETURN episodes, the strict semantic gate takes over 0/24, so its 4.2% success rate is the frozen rate. Origin homing succeeds on 18/24 and same-parser retrace on 16/24. In the auxiliary paired ablation, Structure rises from 4/21 to 16/21 and Precise from 6/36 to 27/36, gains of 12 and 21 successes. This ablation probes two axes. The 27 residual-complete flags and these paired slices come from the same compiler and the same experiment. The flags count diagnostic-export instructions, 27 in all and all on Precise. The paired rows count episodes in the auxiliary cohorts, 21 on Structure and 36 on Precise. The 27 flags are not those 36 episodes and are not that

comparison's denominator. Structure's 21 episodes are in the paired slice and not among the 27 flags, because the flag counts metric residual-complete instructions only.

The 47 held-out human-labeled instructions comprise 15 fully executable (E), 18 mixed (M) and 14 grounding-required (G). Recognition takeover fires on 12/15, 8/18 and 5/14, while the semantic gate fires on 11/15, 2/18 and 0/14. On E, frozen EGO succeeds on 5/15, recognition PNav on 10/15 and semantic routing on 11/15. On M and G, semantic routing matches the frozen results of 8/18 and 6/14. Recognition PNav succeeds on 7/18 and 5/14. Across all 47, the three arms succeed on 19, 22 and 25 episodes, respectively.

E Reporting Rules

These rules describe how the results in this paper are produced and how the claims that follow from them are bounded.

1. The NavSpace RGB paired numbers use the no-privileged-geometry setting of Section 3.3. Other diagnostic rows may use simulated depth, geometry or a hidden-goal oracle and are never pooled with the RGB comparison.
2. The frozen local control is the official `TidalYang/SNav-7B` with no override, and a published number is never the paired baseline of an experiment we ran. Leaderboard figures provide context only, and no subtraction crosses the two blocks.
3. Trace identity is a conditional property of non-taken-over episodes and never global no-harm.
4. Start-in-goal episodes remain in benchmark SR denominators but are excluded from attribution of an executable-program effect. A RETURN-language label never establishes completion of intermediate instruction semantics.
5. We do not claim that the base policy understands numbers. The local parser census records the frozen compiler's outputs, not model number comprehension or independent semantic parse precision.
6. A gain on the offline slice whose ideal metric endpoints match the goal is a property of the combined parser and executor. It is not attributed to either component alone.
7. "Not obtainable" is limited to the frozen interface, the probes we ran and the stated thresholds, and it is not a claim that the target information is absent from the environment or that no readout could ever supply it.
8. Any change to the parser, grammar, radius, quantization or takeover condition invalidates the labels, so labels are regenerated and both arms rerun together.
9. The per-episode outputs do not jointly export program closure, established-origin state and failure causes. We therefore do not report a quantitative L_3 or true RETURN-completion slice from this 54.2% run.

F Relational Event Programs

Related to Section 4.3, Space Structure instructions use relational quantifiers rather than a number with a unit. The four reductions below do not yield a stable arrival program.

Table 12: Attempts to compile relational language into event programs (Section 4.3).

Candidate program	**Measurement**	**Conclusion**
Stop at the named crossing	Goal entry there is rare in the tested subset	An ordinal is not an action-event count
Use observed crossing count	41.7% of Spatial episodes have a larger count, with a tail to 19	An ordinal gives no path budget
Rank candidates by distance	Median candidate span: 0.05 to 0.54 m	Similar to the 0.25 m step, so candidates remain hard to separate
Stop on region entry	Only 3.3% to 8.3% lie inside the goal circle	Region entry is not goal arrival

G Additional Discussion on State Placement and Target Interfaces

G.1 Model-Internal Placement and Origin Scope

The placements inside the model summarized in Table 13 were weaker in the reported diagnostics. State written into the prompt was not consumed as persistent program state, and generated program text did not itself become executable state. These observations motivate an explicit execution interface, but they do not rule out other ways to realize one. The `establish` and `relabel` pair gives a uniform interface for origins established by the method's own motion. Relabeling at the clause level was disabled in the reported RGB run. Extending the interface to a map or object origin would require a way to establish, verify and release that origin. This extension was not evaluated here.

Table 13: Placing the same computation inside the model (Section 5.1)

Location	Minimal intervention	Observation	Mechanistic reading
Model input	Writing dead reckoning, remaining distance or counts into the prompt	Closed-loop results stay close to the base policy	A state that is seen is not a state that is consumed continuously
Model output	Letting the video-language model generate a `TURN/MOVE/STOP` program	Generation and execution still share an unauditable hidden state	Program text does not automatically become program state
Training objective	Imitating a shortest-path teacher that reads the hidden goal	Students of different capacities fall back to the base policy or to a posterior average	Expert actions are not a deterministic function of student observations

G.2 An Illustrative Target-State Interface

The grounding probes leave open how a future system should represent a target. One possible design would maintain an instance distribution $p(z \mid L, o_{\le t})$ and a local-position posterior $p(q_t(z) \mid z, h_t)$, then allow `establish(object)` only when both identity and position are sufficiently certain:

$$\alpha_t^{\text{object}} = \mathbf{1}\big[H\big(p(z \mid L, o_{\le t})\big) \le \tau_z \ \wedge \ \operatorname{tr}\Sigma_{q_t(z)} \le \tau_q\big] .$$

Here H is entropy and $\Sigma_{q_t(z)}$ is the covariance of the position posterior, with $\tau_z = 0.45$ nats and $\tau_q = 0.20\,\mathrm{m}^2$. High identity uncertainty calls for another observation, and high position uncertainty for a change of viewpoint. Table 14 shows that those two tests still hand control to the wrong referent unless the next view can reject it. Each cell has 240 episodes. Success is arrival within 0.8 m, a false commit is an opened gate on the wrong referent, and abstention is a closed gate. A commit to the correct referent that misses the arrival radius is none of these three, so the columns are not a partition. On the area-behind-the-sofa full gate the unrounded rates are 73.75, 0 and 22.5, and the remaining 3.75 points are correct commits that miss 0.8 m. The other rows have no such residual. The executor is idealized. This is not a NavSpace run.

Table 14: Belief gate before geometric takeover.

Condition	SR	False commit	Abstain
Named sofa, entropy + covariance	0.0	100	0.0
Named sofa, second view of the same track	0.0	64.6	35.4
Named sofa, full gate, passive re-look	0.0	0.0	100
Named sofa, full gate, active view	**72.9**	0.0	27.1
Named sofa, takeover then locked	**81.7**	0.0	18.3
Named sofa, attribute accuracy 0.55	1.3	0.4	98.3
Bare noun, two sofas, full gate	7.1	5.0	87.9
Area behind the sofa, full gate	**73.8**	0.0	22.5
Farthest room, entropy + covariance	30.0	70.0	0.0
Farthest room, full gate	**100**	0.0	0.0
Second door, one count	68.3	29.6	2.1
Second door, two agreeing passes	45.8	5.0	49.2
Second door, full gate	**75.8**	12.9	11.3