PoseVLN: Learning Geometric State Evolution for Vision-Language Navigation

Pengyu Hou[1] Mingfei Han[2] Haihong Hao[2] Xiaolong Wu[2]
Bo Dong[1] Xiaojun Chang[2] Salman Khan Bin Shi[1]

[1]Xi'an Jiaotong University
[2]University of Science and Technology of China

Abstract Vision-Language Navigation (VLN) requires an embodied agent to relate past experience, current spatial state, and future motion while following natural-language instructions. Recent approaches improve these capabilities through long-horizon memory, state-aware reasoning, and future prediction, yet these components are often modeled separately, leaving navigation state evolution insufficiently connected across time. We argue that camera pose provides a natural geometric representation for bridging past, present, and future states. Based on this insight, we propose PoseVLN, a pose-aware VLN framework that jointly learns navigation decisions and geometric state evolution. PoseVLN organizes visual memory and the current observation in an episode-relative coordinate frame, and jointly predicts the next K navigation actions together with a local ΔPose over the same horizon. The predicted ΔPose is composed with the current trajectory-level pose to obtain the future pose state, which is recursively propagated throughout navigation, forming a continuous geometric state evolution from past memory to future motion. Experiments on the R2R-CE and RxR-CE Val-Unseen splits demonstrate the effectiveness of explicitly modeling geometric state evolution, with PoseVLN outperforming the strongest compared baselines by 2.6 and 2.9 percentage points in success rate, respectively, using only single RGB observations.

1 Introduction

Vision-Language Navigation (VLN) requires an embodied agent to follow natural-language instructions and navigate to a target by interpreting visual observations and taking appropriate actions. Recent advances have extended VLN beyond direct instruction and observation matching, with increasing attention to memory, spatial reasoning, and predictive modeling. Memory-based approaches preserve information from previous observations and decisions [21, 25, 36, 39, 46], while state-aware and geometric methods support navigation through progress reasoning, structured scene representations, or explicit spatial features [18, 30, 34, 39, 46]. Predictive approaches further anticipate future actions, observations, landmarks, or latent dynamics before execution [8, 13, 28, 29, 38, 42, 47]. Together, these directions highlight the importance of relating past experience, the current navigation context, and future motion.

Despite these advances, two aspects of spatial modeling remain insufficiently addressed. First, past and current observations are not necessarily anchored in an explicit common spatial reference. Retrieving a relevant memory therefore does not by itself establish its viewpoint's position and orientation relative to the agent, leaving this relationship to be inferred for spatial decision-making. Second, future prediction does not necessarily specify an explicit geometric transition of the navigation state. Anticipated actions or visual content can guide action selection without providing an updated position and orientation for subsequent decisions. These gaps motivate a common geometric representation that explicitly relates past observations to the current state and describes how that state changes over the upcoming action horizon.

Camera pose provides a compact representation for addressing both needs. A trajectory-level pose specifies the position and orientation of the camera in a shared coordinate frame, associating each past and current observation with the spatial state from which it was acquired. Relative poses between these observations then make their spatial relationships explicit. For future motion, a local pose change describes the translation and rotation over the upcoming action horizon. Composing this change with the current pose yields a future pose in the same coordinate frame, allowing the resulting state to be carried forward and associated with subsequent observations. Together, trajectory-level poses and local pose

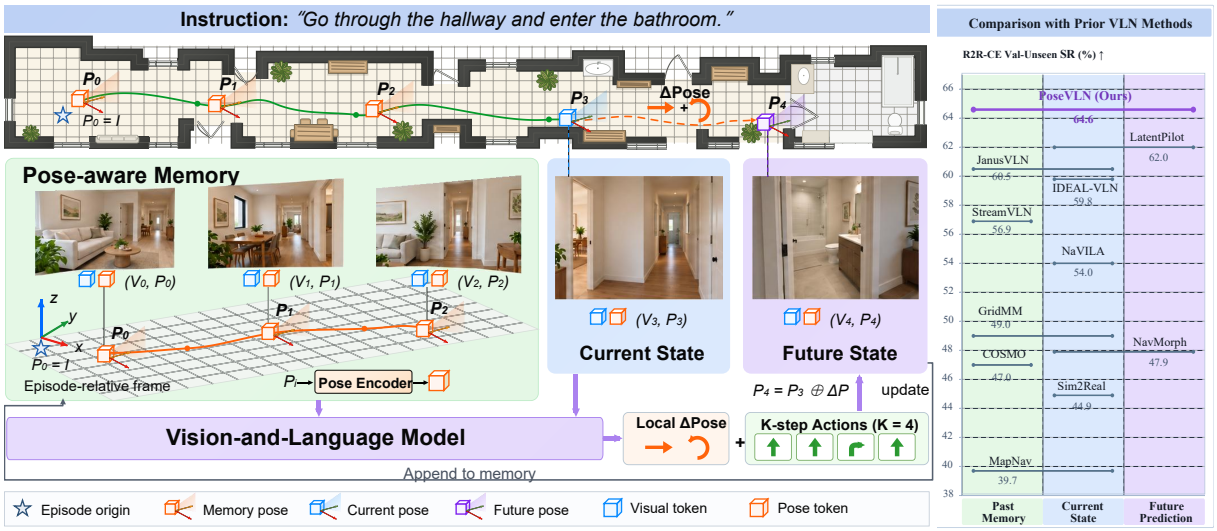


Figure 1. The PoseVLN framework and comparison with prior methods. Left: pose-aware memory and geometric state evolution. Right: comparison with prior VLN methods by navigation performance and temporal focus.

changes provide a common geometric description of past observations, the current viewpoint, and future motion. We refer to this continual maintenance and update of the agent's spatial state as geometric state evolution.

Based on this insight, we introduce PoseVLN, a pose-aware VLN framework that explicitly models navigation as geometric state evolution in a trajectory-level coordinate space. We define the first observation as the origin of an episode-relative coordinate system and associate each subsequent visual observation in memory with a trajectory-level pose state, allowing past observations and the current state to be organized within a common spatial reference. As illustrated in Figure 1, given the instruction, current observation, visual memory, and their associated pose states, PoseVLN jointly predicts the next K navigation actions and a local ΔPose that describes the geometric state transition over the same action horizon. The predicted ΔPose is composed with the current trajectory-level pose to obtain the future pose state, which is stored back into memory and reused in subsequent navigation. During training, ground-truth trajectory poses provide supervision for geometric state transitions. At inference, PoseVLN operates solely on a monocular RGB stream, without depth or external odometry. The first observation is assigned the identity pose $P_0 = I$ to define the episode-relative frame. No ground-truth pose is used, and all subsequent pose states are recursively predicted by the model. Our main contributions are three-fold:

- We propose PoseVLN, a pose-aware VLN framework that models geometric state evolution through joint action and local ΔPose prediction, recursively propagating poses without step-wise ground-truth pose or odometry at inference.
- We demonstrate strong performance on R2R-CE and RxR-CE, showing that geometric state evolution improves long-horizon navigation using only monocular RGB observations.
- We conduct comprehensive ablations and pose evaluations to analyze the relationship between pose estimation and navigation, providing new insights into their interplay.

2 Related Work

2.1 Memory and State-Aware Navigation

Video-based approaches such as NaVid [45] and Uni-NaVid [44] demonstrate the effectiveness of VLM/VLA models for continuous navigation. To support longer horizons, StreamVLN [36] combines fast short-term context with slow-updating compressed memory, while Towards Long-Horizon VLN [25] introduces multi-granularity dynamic memory. MapNav [46], ProFocus [39], MemVLN [21], and JanusVLN [43] further improve memory utilization through semantic maps, selective reasoning, episodic/procedural memory, or decoupled semantic–spatial memory. Complementarily, Dynam3D [34] constructs hierarchical 3D representations from posed RGB-D observations, GA-VLN [40] uses depth and camera poses to build geometry-aware bird's-eye-view representations, and Progress-Think [30] models

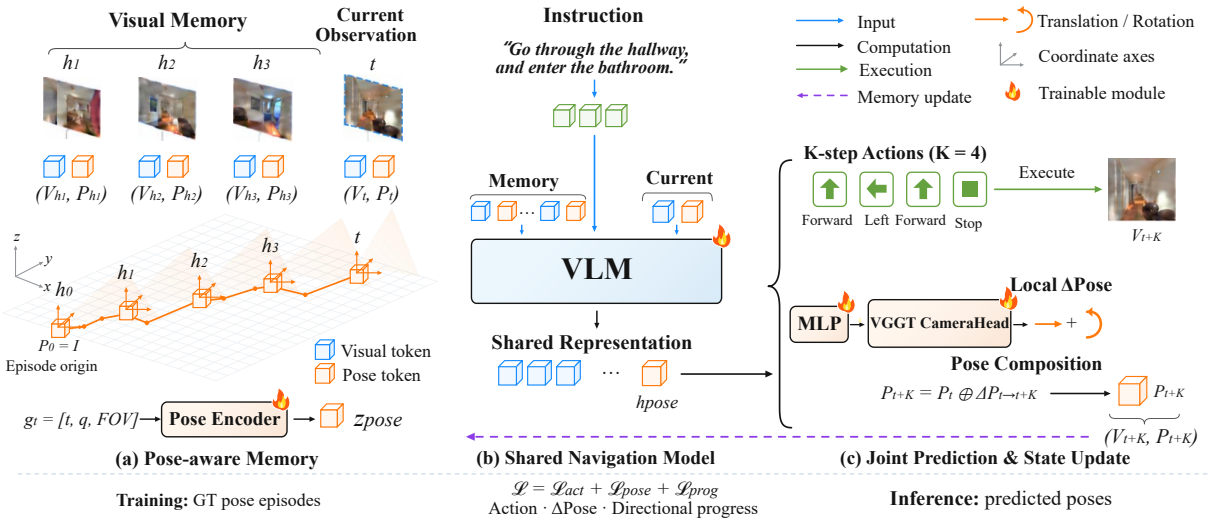


Figure 2. Overview of PoseVLN. Pose-aware memory associates observations with trajectory-level poses. A shared multimodal representation jointly predicts K-step actions and a local ΔPose, which is recursively propagated to update the geometric state.

semantic instruction progress from observation memory. Beyond VLN, Cambrian-P [41] highlights camera pose as a geometric signal for relating observations across viewpoints.

2.2 Predictive and World-Model-Based Navigation

Predictive approaches anticipate future outcomes to support navigation decisions. NavMorph [42] models latent environmental dynamics for foresighted planning, while NavQ [38] estimates the future prospects of candidate actions. MonoDream [29] predicts current and future panoramic cues from monocular observations, while SpaAct [19] learns action-conditioned visual transitions through future-frame selection. WNM-3D [13] further couples future-view prediction with navigation actions under geometry-aware scene representations. LatentPilot [10] incorporates latent visual reasoning to anticipate future scene information for navigation. These methods support navigation through future prediction and structured scene representations. PoseVLN focuses on explicitly propagating a camera-pose state: trajectory-level poses associate past and current observations within a shared coordinate frame, while local ΔPose specifies the state transition over the predicted action horizon.

3 Method

Framework Overview. We propose **PoseVLN**, a pose-aware VLN framework built upon StreamVLN under the single-RGB streaming setting. As illustrated in Figure 2, pose-aware memory associates observations with camera poses in a shared episode-relative coordinate frame, where poses are encoded into embeddings and inserted alongside the corresponding visual tokens. Conditioned on these inputs and the instruction, the VLM produces a shared representation for jointly predicting the next K navigation actions and a local ΔPose over the same horizon. The predicted transition is composed with the current trajectory-level pose, and the resulting future pose is paired with the newly acquired observation and written back into memory. At inference, this process proceeds recursively from $P_0 = I_4$, requiring neither step-wise ground-truth poses nor odometry.

3.1 Preliminary

Navigation Task. Given an instruction I, the agent navigates from egocentric RGB observations $\{V_t\}_{t=0}^{T}$. At step t, it observes V_t and maintains pose-aware memory

$$\mathcal{M}_t = \{(V_{h_i}, P_{h_i})\}_{i=1}^{N_h}, \tag{1}$$

where P_{h_i} denotes the trajectory-level pose associated with memory observation V_{h_i}. The action space is $\mathcal{A} = \{\texttt{Forward}, \texttt{Left}, \texttt{Right}, \texttt{Stop}\}$, and the model predicts a K-step action chunk

$$A_{t:t+K-1} = (a_t, \ldots, a_{t+K-1}), \qquad a_{t+k} \in \mathcal{A}, \tag{2}$$

with $K = 4$ in our experiments.

Camera Pose Tokens. Let $P_i \in SE(3)$ denote the camera-to-episode transformation of V_i, with $P_0 = I_4$. For tokenization, we encode its inverse $E_i^{\text{w2c}} = P_i^{-1}$:

$$g_i = [\mathbf{t}_i, \mathbf{q}_i, \bar{f}^h, \bar{f}^w] \in \mathbb{R}^9, \tag{3}$$

where $\mathbf{t}_i$ and $\mathbf{q}_i$ are the extrinsic translation and unit WXYZ rotation quaternion of E_i^{w2c}. The FoV entries are fixed sensor constants. A pose encoder maps g_i to $z_i^{\text{pose}} = \text{PoseEncoder}(g_i)$, which is concatenated with the corresponding visual tokens. Coordinate and parameterization details appear in Appendices A.1 and A.3.

3.2 PoseVLN

Joint Action and Pose Prediction. The instruction I, pose-aware memory $\mathcal{M}_t$, and current observation–pose pair (V_t, P_t) are jointly processed by the VLM to predict navigation actions and a local pose transition over the same K-step horizon:

$$(I, \mathcal{M}_t, (V_t, P_t)) \longrightarrow \left(A_{t:t+K-1}, \Delta\hat{P}_{t\to t+K}\right). \tag{4}$$

The shared representation generates the K action tokens, while the current pose-token hidden state passes through an MLP and a VGGT CameraHead [27] for iterative local-pose regression. The predicted transition is then decoded and composed with the current pose to update the geometric state.

Pose-aware Geometric State Update. All memory and current poses share the same episode-relative frame. Using column-vector coordinates, the local transition is $\Delta P_{t\to t+K} = P_t^{-1} P_{t+K}$. The CameraHead parameterizes this transition through its inverse w2c encoding, $\Delta E_{t\to t+K}^{\text{w2c}} = (\Delta P_{t\to t+K})^{-1}$. During decoding, raw quaternion outputs are normalized and converted to rotation matrices. The future trajectory-level pose is then obtained by

$$\hat{P}_{t+K} = P_t \oplus \Delta\hat{P}_{t\to t+K} = P_t \left(\Delta\hat{E}_{t\to t+K}^{\text{w2c}}\right)^{-1}, \tag{5}$$

where $\oplus$ denotes rigid-body pose composition. After the action chunk is executed and V_{t+K} becomes available, the pair $(V_{t+K}, \hat{P}_{t+K})$ is written back into memory for subsequent predictions. At inference, this update is applied recursively from $P_0 = I_4$, with P_t given by the previously predicted state. Local-transform construction and recursive propagation are detailed in Appendices A.2 and A.4.

3.3 Training Objective

PoseVLN is trained with action prediction, pose regression, and directional progress objectives.

Action Loss. The predicted action chunk is supervised with an autoregressive cross-entropy loss:

$$\mathcal{L}_{\text{act}} = -\frac{1}{K} \sum_{k=0}^{K-1} \log p_\theta \left(a_{t+k}^* \mid a_{t:t+k-1}^*, I, \mathcal{M}_t, V_t, P_t\right). \tag{6}$$

Pose Loss. Let $(\mathbf{t}^\Delta, \mathbf{q}^\Delta)$ encode the ground-truth inverse local transform $\Delta E_{t\to t+K}^{\text{w2c},*}$. GT rotations are deterministically converted to canonical WXYZ quaternions with $\|\mathbf{q}^\Delta\|_2 = 1$ and $q_w^\Delta \geq 0$. Following the component-wise regression of Cambrian-P [41] and VGGT [27], we optimize

$$\mathcal{L}_{\text{pose}} = \sum_{s=1}^{S} w_s \left(\|\hat{\mathbf{t}}^{(s)} - \mathbf{t}^\Delta\|_1 + \|\hat{\mathbf{q}}^{(s)} - \mathbf{q}^\Delta\|_1\right), \tag{7}$$

where w_s assigns larger weights to later refinement stages. The quaternion term regresses the canonical target from the raw output; it is not a geodesic or sign-invariant loss. Predicted quaternions are normalized only during decoding, not when computing the regression loss.

Directional Progress Loss. To discourage insufficient predicted translation along the target direction, we complement the pose regression loss with a directional progress constraint. Specifically, we constrain the predicted translation's projection onto the target direction in the inverse-extrinsic encoding. Let $d = \|\mathbf{t}^{\Delta}\|_2$; for $d > \tau$, we define

$$\mathcal{L}_{\text{prog}} = \max\left(0,\ \rho d - \hat{\mathbf{t}}^{\top}\frac{\mathbf{t}^{\Delta}}{d}\right), \qquad d > \tau, \tag{8}$$

where $\hat{\mathbf{t}}$ is the final-stage translation, ρ specifies the required projection ratio, and τ filters negligible translations. The loss is zero for $d \leq \tau$. Both translation terms represent inverse-extrinsic components, rather than camera-center displacements. The complete formulation and hyperparameters appear in Appendix C.

The overall objective is

$$\mathcal{L} = \mathcal{L}_{\text{act}} + \lambda_{\text{pose}}\mathcal{L}_{\text{pose}} + \lambda_{\text{prog}}\mathcal{L}_{\text{prog}}. \tag{9}$$

Here, λ_{pose} and λ_{prog} weight pose regression and directional progress supervision, respectively.

Training and Inference. Initialized from StreamVLN [36], PoseVLN undergoes supervised cold-start training followed by iterative closed-loop DAgger refinement [10, 36], without introducing additional datasets. Following LatentPilot's [10] training pipeline, PoseVLN further incorporates pose-aware inputs and geometric supervision. During rollout collection, predicted pose states are recursively propagated, allowing pose errors to affect subsequent decisions and visited states. Simulator camera poses are recorded for pose context and local ΔPose supervision, and are not used as regular step-wise policy inputs except for re-anchoring during DAgger expert recovery. At inference, only the first observation defines the episode-relative origin ($P_0 = I$); all subsequent poses are recursively predicted without ground-truth poses or external odometry. Further details on pose propagation and implementation are provided in Appendices A.4 and E, respectively.

4 Experiments

4.1 Experimental Setup

Simulation Environments and Metrics. Following recent continuous Vision-Language Navigation methods [7, 15, 36, 43], we evaluate PoseVLN on two standard VLN-CE benchmarks, R2R-CE [3] and RxR-CE English Val-Unseen [17]. Both benchmarks are built upon Matterport3D environments and evaluated in the Habitat simulator [24]. We report results on the Val-Unseen splits. For navigation evaluation, we report Navigation Error (NE), Oracle Success Rate (OS), Success Rate (SR), Success weighted by Path Length (SPL), and normalized Dynamic Time Warping (nDTW). For pose evaluation, we report translational and rotational Relative Pose Error (RPE) over different temporal horizons. All baseline results are taken from their respective papers, with observation settings indicated in the comparison tables.

Implementation Details. PoseVLN is initialized from StreamVLN [36], whose navigation backbone is built on LLaVA-Video 7B with Qwen2-7B as the language model. We follow the same single-RGB streaming setting and train on trajectories from R2R [3], RxR [17], and ScaleVLN [35], together with closed-loop DAgger rollouts [10, 36], without introducing additional datasets. PoseVLN augments the backbone with pose-aware inputs and geometric supervision. Detailed implementation settings and training hyperparameters are provided in Appendix E.

Table 1. Comparison with state-of-the-art methods on the R2R-CE Val-Unseen split. Pano., Odo., Depth, and S.RGB denote panoramic view, odometry, depth, and single RGB observations, respectively. All baseline results are reported from their respective papers.

Method	Observation				DAgger	R2R-CE Val-Unseen			
	Pano.	Odo.	Depth	S.RGB		NE↓	OS↑	SR↑	SPL↑
HPN+DN [ICCV21] [16]	✓	✓	✓			6.31	40.0	36.0	34.0
CMA [CVPR22] [11]	✓	✓	✓			6.20	52.0	41.0	36.0
Sim2Sim [ECCV22] [14]	✓	✓	✓			6.07	52.0	43.0	36.0
VLN↻BERT [CVPR22] [11]	✓	✓	✓			5.74	53.0	44.0	39.0
Ego^2-Map [ICCV23] [12]	✓	✓	✓			5.54	56.0	47.0	41.0
DreamWalker [ICCV23] [26]	✓	✓	✓			5.53	59.0	49.0	44.0
GridMM [ICCV23] [32]	✓	✓	✓			5.11	61.0	49.0	41.0
Reborn [CVPR22] [1]	✓	✓	✓			5.40	57.0	50.0	46.0
ScaleVLN [ICCV23] [35]	✓	✓	✓			4.80	–	55.0	51.0
ETPNav [TPAMI24] [2]	✓	✓	✓		✓	4.71	65.0	57.0	49.0
Seq2Seq [ECCV20] [15]			✓	✓	✓	7.77	37.0	25.0	22.0
R2R-CMTP [CVPR21] [5]	✓	✓	✓			7.90	38.0	26.4	22.7
InstructNav [CoRL24] [22]	✓	✓	✓			6.89	–	31.0	24.0
CM2 [CVPR22] [9]		✓	✓	✓		7.02	41.5	34.3	27.6
LAW [EMNLP21] [23]		✓	✓	✓	✓	6.83	44.0	35.0	31.0
WS-MGMap [NIPS22] [6]		✓	✓	✓		6.28	47.6	38.9	34.3
MapNav [ACL25] [46]		✓	✓	✓	✓	4.93	53.0	39.7	37.2
NaVid-4D [ICRA25] [20]			✓	✓		5.99	55.7	43.8	37.1
Sim2Real [CoRL24] [33]		✓	✓	✓		5.95	55.8	44.9	30.4
COSMO [ICCV25] [48]	✓					–	56.0	47.0	40.0
AO-Planner [AAAI25] [4]	✓		✓			5.55	59.0	47.0	33.0
g3D-LF [CVPR25] [31]		✓	✓	✓		5.70	59.5	47.2	34.6
NavMorph [ICCV25] [42]			✓	✓		5.75	56.9	47.9	33.2
GA-VLN [CVPR26] [40]		✓	✓	✓		4.80	67.6	61.0	55.2
NaVid [RSS24] [45]				✓	✓	5.47	49.1	37.4	35.9
Uni-NaVid [RSS25] [44]				✓	✓	5.58	53.3	47.0	42.7
NaVILA [RSS25] [7]				✓		5.22	62.5	54.0	49.0
StreamVLN [ICRA26] [36]				✓	✓	4.98	64.2	56.9	51.9
IDEAL-VLN [ICML26] [37]				✓		4.62	64.9	59.8	52.9
JanusVLN [ICLR26] [43]				✓	✓	4.78	65.2	60.5	56.8
LatentPilot [ECCV26] [10]				✓	✓	4.41	66.3	62.0	58.0
PoseVLN (Ours)				✓	✓	**4.01**	**69.4**	**64.6**	**59.4**

4.2 Main Results

Results on R2R-CE. As shown in Table 1, we compare PoseVLN with a broad range of VLN methods on the R2R-CE Val-Unseen split under different observation settings. PoseVLN operates solely from a single RGB stream, without panoramic observations, depth, or step-wise odometry.

PoseVLN achieves an NE of 4.01, an OS of 69.4, an SR of 64.6, and an SPL of 59.4, obtaining the best performance among all compared methods across all four metrics. Compared with StreamVLN [36], which serves as our navigation backbone, PoseVLN improves SR from 56.9 to 64.6 and SPL from 51.9 to 59.4, corresponding to absolute gains of 7.7 and 7.5 points, while reducing NE from 4.98 to 4.01 and increasing OS from 64.2 to 69.4. These gains indicate that explicitly modeling and propagating geometric state provides complementary spatial information beyond the visual-memory representation of the

Table 2. Comparison with state-of-the-art methods on the RxR-CE Val-Unseen split.

Method	Observation				DAgger	RxR-CE Val-Unseen			
	Pano.	Odo.	Depth	S.RGB		NE↓	SR↑	SPL↑	nDTW↑
CMA [CVPR22] [11]	✓	✓	✓			8.76	26.5	22.1	47.0
VLN↻BERT [CVPR22] [11]	✓	✓	✓			8.98	27.0	22.6	46.7
Reborn [CVPR22] [1]	✓	✓	✓			5.98	48.6	42.0	63.3
ETPNav [TPAMI24] [2]	✓	✓	✓		✓	5.64	54.7	44.8	61.9
LAW [EMNLP21] [23]		✓	✓	✓	✓	10.90	8.0	8.0	38.0
Seq2Seq [ECCV20] [15]			✓	✓	✓	12.10	13.9	11.9	30.8
Sim2Real [CoRL24] [33]		✓	✓	✓		8.79	25.5	18.1	–
NavMorph [ICCV25] [42]			✓	✓		8.85	30.8	22.8	44.2
AO-Planner [AAAI25] [4]	✓		✓			7.06	43.3	30.5	50.1
GA-VLN [CVPR26] [40]		✓	✓	✓		5.88	55.4	45.2	–
Uni-NaVid [RSS25] [44]				✓	✓	6.24	48.7	40.9	–
NaVILA [RSS25] [7]				✓		6.77	49.3	44.0	58.8
StreamVLN [ICRA26] [36]				✓	✓	6.22	52.9	46.0	61.9
JanusVLN [ICLR26] [43]				✓	✓	6.06	56.2	47.5	62.1
LatentPilot [ECCV26] [10]				✓	✓	5.19	58.2	49.9	67.5
IDEAL-VLN [ICML26] [37]				✓		5.23	59.1	50.7	64.5
PoseVLN (Ours)				✓	✓	**4.76**	**62.0**	**53.5**	**70.5**

navigation backbone, improving both goal reaching and path efficiency without introducing additional sensing modalities. Notably, these gains are achieved without using ground-truth pose or odometry during inference, supporting our hypothesis that camera pose can be learned and propagated as an internal geometric state for navigation.

Results on RxR-CE. As shown in Table 2, PoseVLN achieves 62.0% SR, 53.5% SPL, and 70.5 nDTW, with an NE of 4.76 m on RxR-CE Val-Unseen. Compared with StreamVLN, PoseVLN improves SR, SPL, and nDTW by 9.1, 7.5, and 8.6 points, respectively. These improvements indicate higher navigation success, better path efficiency, and closer agreement with the reference trajectories on this challenging benchmark. Nevertheless, the remaining unsuccessful episodes motivate further analysis of accumulated pose errors and stopping behavior over long navigation horizons.

4.3 Ablation and Analysis

We conduct a series of ablation and diagnostic experiments to better understand how pose-aware learning affects navigation performance and geometric state estimation.

Effect of Training Objectives and Pose Inputs. We conduct cumulative ablations during cold-start training on R2R, RxR, and ScaleVLN. As shown in Table 3, action-only replay reduces the reproduced StreamVLN baseline from 56.2% to 53.5% SR. This is consistent with replaying only the cold-start data shifting the policy away from its later optimized state. Adding pose inputs and $\mathcal{L}_{\text{pose}}$ recovers SR to 56.3%, while introducing $\mathcal{L}_{\text{prog}}$ further improves it to 57.2%. Roughly one quarter of the training segments contain pure rotations with little translation, which may partly explain the tendency toward near-zero translation predictions under pose regression alone. The progress loss discourages insufficient translation along the target direction within the adopted pose parameterization. Additional diagnostics in Appendix G show that the final checkpoint trained with the progress loss has a relative translation magnitude deficit of 8.65%, compared with 79.23% without it, while the fraction of predictions below 0.1 m decreases from 64.32% to 4.61%.

We further apply DAgger refinement using the same aggregated closed-loop rollout data. The action-

Table 3. Geometric supervision and training ablations.

Training	Objective	Pose	SR (%)	SPL (%)	NE (m)
StreamVLN	–	None	56.2	50.0	5.10
w/o DAgger	$\mathcal{L}_{act}$	None	53.5	48.0	5.25
	$\mathcal{L}_{act} + \mathcal{L}_{pose}$	Pred.	56.3	51.4	4.88
	$\mathcal{L}_{act} + \mathcal{L}_{pose} + \mathcal{L}_{prog}$	Pred.	57.2	52.2	4.71
+ DAgger	$\mathcal{L}_{act}$	None	57.0	53.4	4.86
	$\mathcal{L}_{act}$	A-DR	47.1	42.8	5.83
	$\mathcal{L}_{act} + \mathcal{L}_{aux} + \mathcal{L}_{prog}$	None	57.5	54.2	4.92
	$\mathcal{L}_{act} + \mathcal{L}_{aux} + \mathcal{L}_{prog}$	A-DR	52.4	48.9	5.45
	$\mathcal{L}_{act} + \mathcal{L}_{pose} + \mathcal{L}_{prog}$	Pred.	**64.6**	**59.4**	**4.01**
	$\mathcal{L}_{act} + \mathcal{L}_{pose} + \mathcal{L}_{prog}$	A-DR	63.2	58.8	4.16

Table 4. Pose prediction comparison across horizons. PoseVLN composes predictions, while pretrained VGGT observes endpoint images.

	#Pairs	Pos. RPE (m)		Rot. RPE (°)	
H	Moving/All	PoseVLN	VGGT	PoseVLN	VGGT
4	20,836/27,912	0.071	**0.015**	7.68	**0.70**
8	21,277/26,069	0.145	**0.065**	**11.67**	11.98
12	20,253/24,228	0.242	**0.215**	**14.73**	23.24
16	18,660/22,387	**0.362**	0.453	**17.40**	31.35

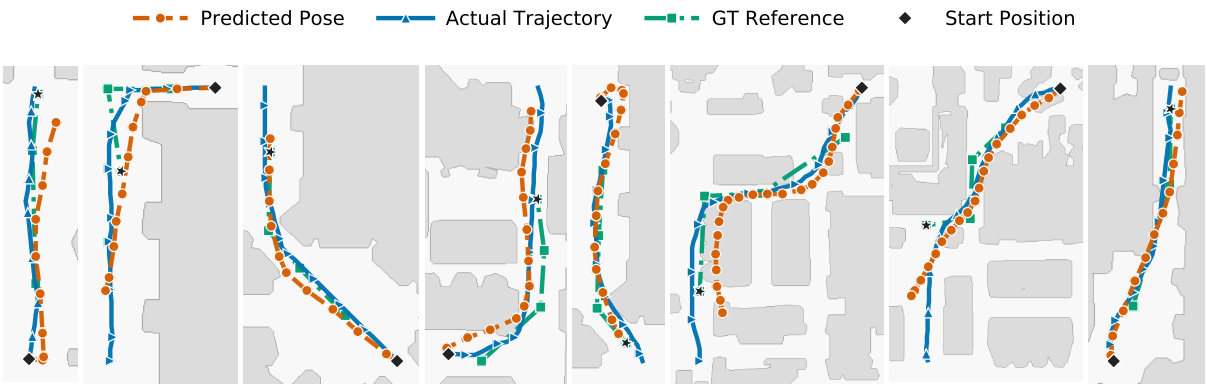


Figure 3. Qualitative comparison between predicted pose trajectories and actual navigation trajectories on R2R-CE Val-Unseen.

only baseline reaches 57.0% SR, whereas full PoseVLN achieves 64.6%, indicating that additional closed-loop training alone cannot explain the final gain. To separate auxiliary geometric learning from pose-conditioned navigation, we train a variant with action loss, auxiliary ΔPose regression ($\mathcal{L}_{aux}$), and directional progress loss ($\mathcal{L}_{prog}$), but without current or historical pose inputs during training or inference. Here, $\mathcal{L}_{aux}$ uses the same regression formulation as $\mathcal{L}_{pose}$, and both geometric losses retain the weights and schedules used in full PoseVLN. This variant achieves 57.5% SR, improving over action-only training by 0.5 percentage points but remaining 7.1 points below full PoseVLN. Together, these results support a jointly trained geometric-state framework rather than auxiliary pose regression alone: geometric supervision helps the model interpret pose information, pose-conditioned training enables effective use of that information, and learned recursive feedback provides additional navigation gains.

We also evaluate Action-based Dead Reckoning (A-DR), which estimates poses by composing nominal rigid-body transforms associated with executed actions without measuring their actual displacement. We provide A-DR poses as text prompts at inference to both the action-only model and the model trained with action and auxiliary geometric supervision. SR decreases by 9.9 and 5.1 points relative to their respective no-pose settings, suggesting that auxiliary geometric supervision reduces sensitivity to these additional prompts, although direct prompting still degrades performance. For full PoseVLN, replacing predicted poses with A-DR estimates through the existing pose-token interface yields 63.2% SR, compared with 64.6% using learned pose feedback. Thus, the full model can use action-integrated states, while learned states provide better navigation performance in this evaluation.

Pose prediction compared with VGGT. We compare PoseVLN with VGGT [27] over horizons $H \in \{4, 8, 12, 16\}$. PoseVLN recursively composes successive $K = 4$ predictions, incorporating newly observed RGB frames, updated memory, and the previously predicted pose at each prediction step. VGGT instead performs direct two-view inference from the endpoint image pair $[I_i, I_{i+H}]$. Since VGGT's monocular two-view reconstruction is subject to global scale ambiguity, translation errors are evaluated after scale alignment. For each valid prediction–GT pair, both outputs are converted to local $c2w$ displacements. A non-negative translation scale is independently estimated as $s^* = \max(0, \hat{\mathbf{t}}^\top \mathbf{t}^{GT} / \|\hat{\mathbf{t}}\|_2^2)$, with the same per-pair alignment procedure applied to both methods. This evaluation measures translation agreement up to scale rather than metric displacement accuracy. Position RPE is evaluated on pairs with GT displacement

Table 5. Pose analysis on R2R-CE Val-Unseen. Left: comparison of pose errors between successful and failed navigation episodes. Right: ablation of memory and current pose inputs.

Each entry: Successful / Failed (S/F)			
H	Raw Pos. (m) ↓	Direction (°) ↓	Rotation (°) ↓
4	**0.145** / 0.383	**7.67** / 12.54	**7.43** / 7.96
8	**0.262** / 0.769	**8.78** / 17.72	**11.09** / 12.29
12	**0.396** / 1.193	**10.67** / 19.29	**13.85** / 15.64
16	**0.554** / 1.660	**12.29** / 22.01	**16.25** / 18.55

Method	Pose Input		R2R-CE Val-Unseen			
	Hist.	Curr.	NE↓	OS↑	SR↑	SPL↑
PoseVLN	✓	✓	**4.01**	**69.4**	**64.6**	**59.4**
w/ Action-DR Pose	✓	✓	4.16	65.7	63.2	58.8
w/o Memory Pose		✓	4.05	68.1	63.3	58.1
w/o Current Pose	✓		4.15	68.1	62.9	57.9
w/o Mem./Curr. Pose			4.11	67.7	62.9	57.7

greater than 0.05 m, while Rotation RPE is computed on all valid pairs. All errors are micro-averaged over the corresponding valid pairs. As shown in Table 4, VGGT achieves lower Position RPE for $H = 4, 8, 12$, whereas PoseVLN performs better at $H = 16$. For Rotation RPE, VGGT achieves lower error at $H = 4$, while PoseVLN is comparable or better at longer horizons.

Comparison between predicted and actual trajectories. Figure 3 qualitatively compares the recursively predicted pose trajectories with the actual navigation trajectories on R2R-CE Val-Unseen. The predicted trajectories generally capture the overall motion direction, major turns, and trajectory shape. In some long-horizon examples, the overall structure remains recognizable despite noticeable endpoint drift, illustrating that coarse trajectory consistency can coexist with accumulated localization errors. These examples suggest that PoseVLN captures meaningful geometric structure over multiple navigation steps, while accurate long-horizon pose propagation remains a challenge.

Pose accuracy in successful and failed navigation episodes. The relationship between geometric state estimation and navigation success is examined by grouping R2R-CE Val-Unseen episodes according to their final outcome (SR $= 1/0$). Here, $H = 4$ denotes the direct $K = 4$ prediction, while $H = 8, 12, 16$ are obtained by autoregressively composing successive $K = 4$ pose transitions. As shown on the left side of Table 5, successful episodes consistently exhibit lower pose errors across all horizons, with larger absolute gaps under longer-horizon composition. At $H = 16$, the mean raw position error reaches 1.660 m for failed episodes, nearly $3\times$ that of successful episodes (0.554 m). Similarly, translation-direction errors are lower in successful episodes than in failed ones: 7.67° versus 12.54° at $H = 4$, and 12.29° versus 22.01° at $H = 16$. These results indicate an association between navigation success and more accurate geometric state estimation, particularly in translation position and direction over long-horizon propagation.

Effect of pose inputs. We further study the contribution of memory and current pose information through an inference-time input ablation. All variants use the same trained PoseVLN checkpoint, with the corresponding pose inputs masked only during evaluation. As shown on the right of Table 5, using both memory and current poses achieves the best performance, with 64.6% SR and 59.4% SPL. Removing either pose source consistently degrades navigation performance, while removing both yields the lowest SR and SPL. Masking the current pose leads to a slightly larger degradation than masking memory poses, while the full model performs best across all reported metrics. These results indicate that both current and memory pose states provide useful geometric context under the jointly trained model.

Episode-Level Pose Error and Navigation Success. To examine long-term geometric consistency beyond local pose prediction, we evaluate accumulated errors over complete R2R-CE Val-Unseen trajectories. Episodes are independently ranked by translational ATE and rotational RMS error and divided into five groups, with 368 episodes in each of Q1–Q4 and 367 in Q5, covering all 1,839 episodes. As shown in Table 6, SR decreases from 71.2% to 45.0% as mean ATE increases from 0.330 to 8.126 m. Similarly, SR decreases from 73.1% to 46.0% as rotational RMS error increases from 8.00° to 68.55°. Although not strictly monotonic, these results associate larger accumulated pose errors with lower navigation success, supporting the value of maintaining useful geometric states over extended trajectories.

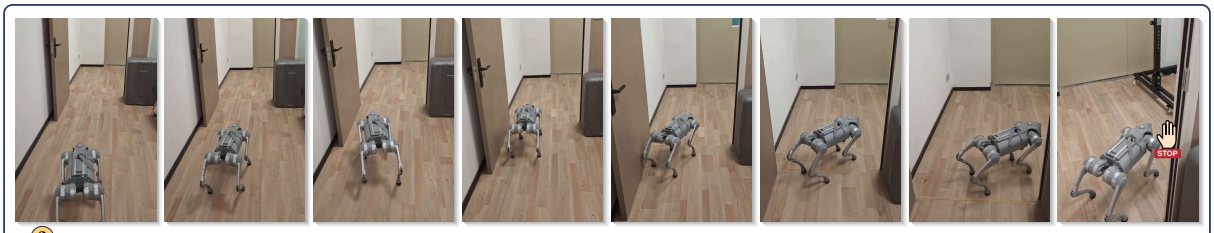


Figure 4. Real-world navigation of PoseVLN on a Unitree Go2.

Learned Pose Feedback Under Execution Discrepancies. We further compare learned ΔPose with Action-based Dead Reckoning (Action-DR), which integrates nominal action transforms through the same pose-token interface without retraining. Table 7 groups 1,839 matched tasks by collision counts in Action-DR rollouts. While Action-DR performs slightly better without collisions (69.5% versus 68.2% SR), learned pose feedback achieves higher SR in every collision-containing group, reaching 23.5% versus 12.3% when collisions exceed ten. In two representative scenes, learned ΔPose also achieves lower unaligned ATE in groups with at least three Action-DR collisions (Appendix F). These results support the navigation benefit of learned geometric state feedback under execution discrepancies. Nevertheless, PoseVLN does not explicitly correct accumulated pose errors after execution, and long-horizon drift remains a limitation.

Table 6. Episode-level pose errors and navigation success on R2R-CE Val-Unseen.

		Translation		Rotation	
Group	**Episodes**	**ATE (m)**	**SR (%)**	**RMS (°)**	**SR (%)**
Q1	368	0.330	71.2	8.00	73.1
Q2	368	0.629	71.5	12.88	73.4
Q3	368	1.008	68.5	19.45	66.6
Q4	368	1.701	66.8	30.63	63.9
Q5	367	8.126	45.0	68.55	46.0
All	1,839	–	64.6	–	64.6

Table 7. Collision-stratified navigation success on R2R-CE Val-Unseen. Collisions and median final horizontal drift are measured from Action-DR rollouts.

Coll.	**Tasks**	**A-DR SR (%)**	**PoseVLN SR (%)**	**A-DR Drift (m)**
0	946	**69.5**	68.2	≈0
1–2	575	63.3	**66.6**	0.072
3–5	169	60.4	**63.3**	0.275
6–10	68	44.1	**50.0**	0.939
> 10	81	12.3	**23.5**	10.411
All	1,839	63.2	**64.6**	–

4.4 Real-World Deployment

We further conduct a preliminary real-world deployment of PoseVLN on a Unitree Go2 equipped with an onboard RGB camera, with inference performed on a remote RTX 4090 server. As shown in Figure 4, the robot successfully follows the instruction, traverses the doorway, and stops after leaving the room. We additionally conduct a quantitative evaluation across two real-world scenes and six navigation task settings, with three trials per task. PoseVLN successfully completes 11 of 18 trials (61.1%), with at least two successful trials in four of the six task settings. Detailed results and trajectory visualizations are provided in Appendix M. This result demonstrates the applicability of PoseVLN to physical robot navigation under real visual observations, while broader evaluation across diverse environments and instructions remains necessary.

5 Limitations

PoseVLN still has several limitations. First, recursive pose composition can accumulate estimation errors and lead to long-horizon drift. Second, while the auxiliary-only ablation helps distinguish the contributions of geometric supervision and recursively propagated pose inputs, their interactions remain to be fully investigated. Finally, the complementarity between learned ΔPose and action-derived pose estimates remains to be further investigated. Future work may explore more robust long-horizon pose propagation, further analysis of the interaction between geometric supervision and state representation, and hybrid geometric estimation that combines learned pose transitions with action-derived motion cues.

6 Conclusion

We introduce PoseVLN, a pose-aware VLN framework that connects past memory, current spatial state, and future motion through geometric state evolution. PoseVLN organizes observations in an episode-relative coordinate frame and jointly predicts navigation actions and local ΔPose, whose recursive propagation maintains an evolving spatial state throughout navigation. Using only single-RGB observations without step-wise ground-truth pose or odometry at inference, PoseVLN achieves strong performance on R2R-CE and RxR-CE. These results demonstrate the effectiveness of camera pose as a compact geometric bridge for temporal state reasoning in VLN.

AI use statement

In this work, we did not use generative AI tools for tasks requiring mandatory disclosure under the ICLR 2027 AI Policy for Authors. Tasks involving mathematical proofs and qualitative or thematic data analysis are not applicable to this work. Additionally, we used ChatGPT to improve the language and readability of the manuscript. We have reviewed all AI-assisted work. Specifically, the authors checked and revised the suggested edits to ensure that they preserved the intended technical meaning and accurately reflected the methods and experimental results. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Reproducibility statement

To facilitate reproducibility, we describe the architecture, pose representation, training objectives, and inference procedure of PoseVLN in Section 3. The datasets, implementation details, training settings, and evaluation protocols are provided in Section 4 and Appendices. The ablation studies and pose-error analyses in Section 4 further document the contributions of the main components and the evaluation of pose prediction.

References

[1] Dong An, Zun Wang, Yangguang Li, Yi Wang, Yicong Hong, Yan Huang, Liang Wang, and Jing Shao. 1st place solutions for rxr-habitat vision-and-language navigation competition (cvpr 2022). *arXiv preprint arXiv:2206.11610*, 2022.

[2] 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. *arXiv preprint arXiv:2304.03047*, 2023.

[3] 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/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[4] Jiaqi Chen, Bingqian Lin, Xinmin Liu, Lin Ma, Xiaodan Liang, and Kwan-Yee K Wong. Affordances-oriented planning using foundation models for continuous vision-language navigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23568–23576, 2025.

[5] Kevin Chen, Junshen K. Chen, Jo Chuang, Marynel Vazquez, and Silvio Savarese. Topological planning with transformers for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[6] Peihao Chen, Dongyu Ji, Kunyang Lin, Runhao Zeng, Thomas H. Li, Mingkui Tan, and Chuang Gan. Weakly-supervised multi-granularity map learning for vision-and-language navigation. *arXiv preprint arXiv:2210.07506*, 2022.

[7] 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. *arXiv preprint arXiv:2412.04453*, 2024.

[8] Guangzhao Dai, Shuo Wang, Zihan Wang, Guo-Sen Xie, Yang Yang, Jinshan Pan, Qianru Sun, and Xiangbo Shu. History to future: Evolving agent with experience and thought for zero-shot vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15177–15187, 2026.

[9] Georgios Georgakis, Karl Schmeckpeper, Karan Wanchoo, Soham Dan, Eleni Miltsakaki, Dan Roth, and Kostas Daniilidis. Cross-modal map learning for vision and language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[10] Haihong Hao, Lei Chen, Mingfei Han, Changlin Li, Dong An, Yuqiang Yang, Zhihui Li, and Xiaojun Chang. Latentpilot: Scene-aware vision-and-language navigation by dreaming ahead with latent visual reasoning. *arXiv preprint arXiv:2603.29165*, 2026.

[11] 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 *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15418–15428. IEEE, 2022.

[12] Yicong Hong, Yang Zhou, Ruiyi Zhang, Franck Dernoncourt, Trung Bui, Stephen Gould, and Hao Tan. Learning navigational visual representations with semantic map supervision. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3032–3044. IEEE, 2023.

[13] Yuehao Huang, Yunzi Wu, Xiaotao Zhang, Xinhai Li, Jiankun Dong, Jiajun Lv, Chi Zhang, Chenjia Bai, Yong Liu, and Xuelong Li. Wnm-3d: A world navigation model with 3d scene conditioning for closed-loop vln. *arXiv preprint arXiv:2608.07267*, 2026.

[14] Jacob Krantz and Stefan Lee. Sim-2-sim transfer for vision-and-language navigation in continuous environments. In *European conference on computer vision*, pages 588–603. Springer, 2022.

[15] 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*, pages 104–120. Springer, 2020.

[16] Jacob Krantz, Aaron Gokaslan, Dhruv Batra, Stefan Lee, and Oleksandr Maksymets. Waypoint models for instruction-guided navigation in continuous environments. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15142–15151. IEEE, 2021.

[17] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[18] Kailing Li, Tianwen Qian, Lijin Yang, Yuqian Fu, Jingyu Gong, Xiaoling Wang, and Liang He. Bridging the 2d-3d gap: A hierarchical semantic-geometric map for vision language navigation. *arXiv preprint arXiv:2606.00095*, 2026.

[19] Pengna Li, Kangyi Wu, Shaoqing Xu, Fang Li, Hanbing Li, Lin Zhao, Kailin Lyu, Long Chen, Zhi-Xin Yang, and Nanning Zheng. Spaact: Spatially-activated transition learning with curriculum adaptation for vision-language navigation. *arXiv preprint arXiv:2604.27620*, 2026.

[20] Haoran Liu, Weikang Wan, Xiqian Yu, Minghan Li, Jiazhao Zhang, Bo Zhao, Zhibo Chen, Zhongyuan Wang, Zhizheng Zhang, and He Wang. Na vid-4d: Unleashing spatial intelligence in egocentric rgb-d videos for vision-and-language navigation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10607–10615. IEEE, 2025.

[21] Yuqi Liu, Shengju Qian, Tianyuan Qu, Mingxian Lin, Zixuan Wang, Xin Wang, Bei Yu, and Jiaya Jia. Memvln: Episodic and procedural memory for vision-and-language navigation. *arXiv preprint arXiv:2607.23504*, 2026.

[22] Yuxing Long, Wenzhe Cai, Hongcheng Wang, Guanqi Zhan, and Hao Dong. Instructnav: Zero-shot system for generic instruction navigation in unexplored environment. *arXiv preprint arXiv:2406.04882*, 2024.

[23] Sonia Raychaudhuri, Saim Wani, Shivansh Patel, Unnat Jain, and Angel Chang. Language-aligned waypoint (law) supervision for vision-and-language navigation in continuous environments. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 4018–4028, 2021.

[24] 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)*, 2019.

[25] Xinshuai Song, Weixing Chen, Yang Liu, Weikai Chen, Guanbin Li, and Liang Lin. Towards long-horizon vision-language navigation: Platform, benchmark and method. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12078–12088. IEEE, 2025.

[26] Hanqing Wang, Wei Liang, Luc Van Gool, and Wenguan Wang. Dreamwalker: Mental planning for continuous vision-language navigation. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10839–10849. IEEE, 2023.

[27] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5294–5306. IEEE, 2025.

[28] Shuo Wang, Yongcai Wang, Wanting Li, Xudong Cai, Yucheng Wang, Maiyue Chen, Kaihui Wang, Zhizhong Su, Deying Li, and Zhaoxin Fan. Aux-Think: Exploring reasoning strategies for data-efficient vision-language navigation. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 30892–30916. Curran Associates, Inc., 2025. doi: 10.52202/085713-1040.

[29] Shuo Wang, Yongcai Wang, Zhaoxin Fan, Yucheng Wang, Maiyue Chen, Kaihui Wang, Zhizhong Su, Wanting Li, Xudong Cai, Yeying Jin, et al. Monodream: Monocular vision-language navigation with panoramic dreaming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 10074–10082, 2026.

[30] Shuo Wang, Yucheng Wang, Guoxin Lian, Yongcai Wang, Maiyue Chen, Kaihui Wang, Bo Zhang, Zhizhong Su, Yutian Zhou, Wanting Li, et al. Progress-think: Semantic progress reasoning for vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4076–4086, 2026.

[31] Zihan Wang and Gim Hee Lee. g3d-lf: Generalizable 3d-language feature fields for embodied tasks. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14191–14202. IEEE, 2025.

[32] Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, and Shuqiang Jiang. Gridmm: Grid memory map for vision-and-language navigation. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15579–15590. IEEE, 2023.

[33] Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, and Shuqiang Jiang. Sim-to-real transfer via 3d feature fields for vision-and-language navigation. *arXiv preprint arXiv:2406.09798*, 2024.

[34] Zihan Wang, Seungjun Lee, and Gim Hee Lee. Dynam3D: Dynamic Layered 3D Tokens Empower VLM for Vision-and-Language Navigation. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 153522–153544. Curran Associates, Inc., 2025. doi: 10.52202/085713-5133. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/e1a1847e39a7b79b41199176b152f0e6-Paper-Conference.pdf.

[35] 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)*, 2023.

[36] Meng Wei, Chenyang Wan, Xiqian Yu, Tai Wang, Yuqiang Yang, Xiaohan Mao, Chenming Zhu, Wenzhe Cai, Hanqing Wang, Yilun Chen, et al. Streamvln: Streaming vision-and-language navigation via slowfast context modeling. *arXiv preprint arXiv:2507.05240*, 2025.

[37] Zihao Xin, Wentong Li, Yixuan Jiang, Bin Wang, Piji Li, Jianke Zhu, Jie Qin, and Sheng-Jun Huang. Instruction decomposition and action alignment for vision-language navigation. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

[38] Peiran Xu, Xicheng Gong, and Yadong Mu. Navq: Learning a q-model for foresighted vision-and-language navigation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6327–6341. IEEE, 2025.

[39] Wei Xue, Mingcheng Li, Xuecheng Wu, Jingqun Tang, Dingkang Yang, and Lihua Zhang. Profocus: Proactive perception and focused reasoning in vision-and-language navigation. *arXiv preprint arXiv:2603.05530*, 2026.

[40] Jiahao Yang, Zihan Wang, Xiangyang Li, Xing Zhu, Yujun Shen, Yinghao Xu, and Shuqiang Jiang. Ga-vln: Geometry-aware bev representation for efficient vision-language navigation. *arXiv preprint arXiv:2605.22036*, 2026.

[41] Jihan Yang, Zifan Zhao, Xichen Pan, Shusheng Yang, Junyi Zhang, Bingyi Kang, Hu Xu, Shang-Wen Li, and Saining Xie. Cambrian-p: Pose-grounded video understanding. *arXiv preprint arXiv:2605.22819*, 2026.

[42] Xuan Yao, Junyu Gao, and Changsheng Xu. Navmorph: A self-evolving world model for vision-and-language navigation in continuous environments. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5536–5546. IEEE, 2025.

[43] Shuang Zeng, Dekang Qi, Xinyuan Chang, Feng Xiong, Shichao Xie, Xiaolong Wu, Shiyi Liang, Mu Xu, and Xing Wei. Janusvln: Decoupling semantics and spatiality with dual implicit memory for vision-language navigation. In *International Conference on Learning Representations*, volume 2026, pages 33001–33026, 2026.

[44] 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. *arXiv preprint arXiv:2412.06224*, 2024.

[45] 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. *arXiv preprint arXiv:2402.15852*, 2024.

[46] Lingfeng Zhang, Xiaoshuai Hao, Qinwen Xu, Qiang Zhang, Xinyao Zhang, Pengwei Wang, Jing Zhang, Zhongyuan Wang, Shanghang Zhang, and Renjing Xu. Mapnav: A novel memory representation via annotated semantic maps for vlm-based vision-and-language navigation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13032–13056, 2025.

[47] 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*, 2026.

[48] Siqi Zhang, Yanyuan Qiao, Qunbo Wang, Zike Yan, Qi Wu, Zhihua Wei, and Jing Liu. Cosmo: Combination of selective memorization for low-cost vision-and-language navigation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5511–5522. IEEE, 2025.

Appendix

This appendix provides additional details of PoseVLN, including the pose representation and recursive geometric state propagation, closed-loop DAgger collection, the Directional Progress Loss, training sample construction, and implementation details. We further clarify how geometric supervision is used during training. Simulator-recorded camera poses are used to construct pose-aware training context and local ΔPose targets, whereas at inference only the first observation defines the episode-relative origin and all subsequent pose states are recursively propagated from model predictions without ground-truth pose or external odometry.

A Pose Representation and Geometric State Propagation

A.1 Episode-relative Camera Coordinates

Using column-vector coordinates, let the simulator camera-to-world transformation of frame t be

$$T_t^{\mathrm{c2w}} = \begin{bmatrix} R_t & p_t \\ 0^\top & 1 \end{bmatrix} \in SE(3), \tag{10}$$

where R_t and p_t denote the camera orientation and center in the simulator global coordinate system.

We use the first camera frame as the episode reference:

$$\bar{T}_t^{\mathrm{c2w}} = \left(T_0^{\mathrm{c2w}}\right)^{-1} T_t^{\mathrm{c2w}}. \tag{11}$$

By construction, $\bar{T}_0^{\mathrm{c2w}} = I_4$. This rigid coordinate change preserves metric scale; no additional trajectory-scale normalization is applied. The main-text trajectory-level state is

$$P_t \equiv \bar{T}_t^{\mathrm{c2w}}, \tag{12}$$

which maps camera coordinates to the episode frame.

For pose tokenization, we use the inverse world-to-camera representation:

$$E_t^{\mathrm{w2c}} = P_t^{-1} = \left(\bar{T}_t^{\mathrm{c2w}}\right)^{-1}. \tag{13}$$

Thus, the trajectory state is stored in c2w form, while its pose token encodes the corresponding w2c rotation and translation.

A.2 Local ΔPose Supervision

For a current timestep i and a future timestep $j = i + K$, the local camera transformation is

$$\Delta T_{i\to j}^{\mathrm{c2w}} = \left(\bar{T}_i^{\mathrm{c2w}}\right)^{-1} \bar{T}_j^{\mathrm{c2w}}. \tag{14}$$

Writing

$$\bar{T}_i^{\mathrm{c2w}} = \begin{bmatrix} \bar{R}_i & \bar{p}_i \\ 0^\top & 1 \end{bmatrix}, \qquad \bar{T}_j^{\mathrm{c2w}} = \begin{bmatrix} \bar{R}_j & \bar{p}_j \\ 0^\top & 1 \end{bmatrix}, \tag{15}$$

gives

$$R_{ij}^{\mathrm{rel}} = \bar{R}_i^\top \bar{R}_j, \qquad p_{ij}^{\mathrm{rel}} = \bar{R}_i^\top (\bar{p}_j - \bar{p}_i). \tag{16}$$

Here, p_{ij}^{rel} is the displacement of the future camera center expressed in the current camera frame. The main-text transition is $\Delta P_{i\to j} = \Delta T_{i\to j}^{\mathrm{c2w}}$.

Substituting Eq. 11 into Eq. 14 yields

$$\Delta T_{i\to j}^{\mathrm{c2w}} = \left(T_i^{\mathrm{c2w}}\right)^{-1} T_j^{\mathrm{c2w}}, \tag{17}$$

showing that the local transformation is independent of the arbitrary global origin and orientation.

We supervise the CameraHead using the inverse local transform:

$$\Delta E_{i\to j}^{\mathrm{w2c}} = \left(\Delta T_{i\to j}^{\mathrm{c2w}}\right)^{-1} = E_j^{\mathrm{w2c}} \left(E_i^{\mathrm{w2c}}\right)^{-1}. \tag{18}$$

This transform maps current-camera coordinates to future-camera coordinates. Its rotation and translation satisfy

$$R_{ij}^{\mathrm{w2c}} = (R_{ij}^{\mathrm{rel}})^\top, \qquad \mathbf{t}_{ij}^{\Delta} = -(R_{ij}^{\mathrm{rel}})^\top p_{ij}^{\mathrm{rel}}. \tag{19}$$

Accordingly, the translation supervised by the pose and Directional Progress objectives is an inverse-extrinsic component, rather than the camera-center displacement p_{ij}^{rel}.

A.3 9D Camera Parameterization

We reuse the VGGT CameraHead architecture and follow the component-wise pose regression of Cambrian-P and VGGT [27, 41]. Our encoding explicitly uses WXYZ quaternion ordering:

$$g_i = [\mathbf{t}_i, \mathbf{q}_i, \bar{f}^h, \bar{f}^w] = [t_x, t_y, t_z, q_w, q_x, q_y, q_z, \bar{f}^h, \bar{f}^w] \in \mathbb{R}^9. \tag{20}$$

For trajectory-level tokens, $\mathbf{t}_i$ and $\mathbf{q}_i$ encode the translation and rotation of $E_i^{\mathrm{w2c}} = P_i^{-1}$. Local targets use the same encoding layout for $\Delta E_{i\to j}^{\mathrm{w2c}}$.

Canonical Quaternion Regression. For a GT rotation R^*, we deterministically construct the target

$$\mathbf{q}^* = \mathrm{Canon}\left(\mathrm{Quat}_{\mathrm{WXYZ}}(R^*)\right), \qquad \|\mathbf{q}^*\|_2 = 1, \quad q_w^* \geq 0, \tag{21}$$

where canonicalization standardizes the quaternion sign. The CameraHead outputs an unconstrained vector $\hat{\mathbf{q}} \in \mathbb{R}^4$, and the rotation term directly minimizes $\|\hat{\mathbf{q}} - \mathbf{q}^*\|_1$ at each refinement stage. This is component-wise regression against a canonical target, rather than a geodesic or sign-invariant rotation loss. Predicted quaternions are normalized during geometric decoding, not before computing this regression loss.

Fixed Field of View. The two FoV entries are retained for compatibility with the 9D CameraHead interface. They are overwritten with known sensor constants during training and inference, contribute no effective FoV loss, and are excluded from geometric state propagation.

A.4 Recursive Pose Propagation

At inference, PoseVLN initializes the episode coordinate system with $\hat{P}_0 = P_0 = I_4$, without querying a GT camera pose. For the final-stage quaternion output, geometric decoding applies

$$\bar{\mathbf{q}} = \frac{\hat{\mathbf{q}}}{\max(\|\hat{\mathbf{q}}\|_2, \epsilon)}, \qquad \hat{R} = \mathrm{QuatToMat}_{\mathrm{WXYZ}}(\bar{\mathbf{q}}). \tag{22}$$

The decoded rotation and predicted translation form $\Delta\hat{E}_{i\to j}^{\mathrm{w2c}}$. We invert this transform before state composition:

$$\Delta\hat{T}_{i\to j}^{\mathrm{c2w}} = \left(\Delta\hat{E}_{i\to j}^{\mathrm{w2c}}\right)^{-1}. \tag{23}$$

For decoded w2c components $(\hat{R}, \hat{\mathbf{t}})$, the inverse has rotation $\hat{R}^\top$ and translation $-\hat{R}^\top\hat{\mathbf{t}}$. The trajectory-level state is updated by

$$\hat{P}_j = \hat{P}_i \Delta\hat{T}_{i\to j}^{\mathrm{c2w}}. \tag{24}$$

Equivalently, under the w2c convention,

$$\hat{E}_j^{\mathrm{w2c}} = \Delta\hat{E}_{i\to j}^{\mathrm{w2c}} \hat{E}_i^{\mathrm{w2c}}. \tag{25}$$

The propagated state $\hat{P}_j$ is paired with the new observation V_j and stored in memory:

$$\mathcal{M}_t = \{(V_{h_1}, \hat{P}_{h_1}), \ldots, (V_{h_{N_h}}, \hat{P}_{h_{N_h}})\}. \tag{26}$$

Its pose token is constructed from $\hat{E}_j^{\mathrm{w2c}} = \hat{P}_j^{-1}$. Thus, all non-initial current and memory states are model-predicted.

The resulting state evolution is

$$\hat{P}_0 \xrightarrow{\Delta\hat{P}_{0\to K}} \hat{P}_K \xrightarrow{\Delta\hat{P}_{K\to 2K}} \hat{P}_{2K} \to \cdots . \tag{27}$$

Pose errors therefore accumulate under the same recursive propagation used for navigation evaluation.

B Closed-Loop DAgger and Pose Collection

We collect additional training trajectories through closed-loop policy rollouts in the Habitat simulator. During collection, the policy executes predicted action chunks while its trajectory-level pose state is recursively propagated from predicted local ΔPose according to Appendix A.4. Consequently, accumulated pose errors can affect subsequent navigation decisions during data collection.

For each simulator state used to produce an RGB observation, we record the corresponding ground-truth camera pose T_t^{c2w}. Local pose targets are subsequently constructed from these temporally aligned camera poses following Appendix A.2. Each executed action is logged together with the RGB observation and camera pose at the simulator state from which that action is issued, ensuring temporal alignment among observations, actions, and pose supervision.

Expert Intervention. During a rollout, we compute the minimum distance d_t between the current agent position and the reference path. The Habitat shortest-path expert takes control when $d_t > 1.0\,\mathrm{m}$ and returns control to the policy when $d_t < 0.5\,\mathrm{m}$. Expert takeover is also triggered when the normalized action-error score exceeds 0.5, the accumulated action errors exceed 8, or the policy produces an invalid `Stop` action.

Pose Re-anchoring. When expert correction is triggered, the internally propagated pose state is re-anchored to the current simulator camera pose:

$$\hat{P}_t \leftarrow \left(T_0^{\mathrm{c2w}}\right)^{-1} T_t^{\mathrm{c2w}}. \tag{28}$$

Before control is returned to the policy, unexecuted action predictions and stale recurrent states are discarded, and closed-loop action and pose prediction resumes from the corrected state.

This re-anchoring is used only during training-data collection and is never applied during evaluation or inference. A corrected rollout therefore contains a policy-generated segment, an expert recovery segment, and a subsequent policy-generated segment. The simulator camera trajectory is recorded independently throughout the rollout and is later used to construct ground-truth pose context and ΔPose supervision for training.

C Directional Progress Loss

The basic pose objective supervises translation and rotation independently. We additionally introduce a directional constraint to discourage local translation predictions with insufficient progress along the ground-truth translation direction.

For the m-th valid local-motion target, let $\mathbf{t}_m^{\Delta}$ denote the ground-truth local w2c translation and $d_m = \|\mathbf{t}_m^{\Delta}\|_2$ its magnitude. For samples satisfying $d_m > \tau$, the normalized local translation direction is

$$\mathbf{u}_m^{\Delta} = \frac{\mathbf{t}_m^{\Delta}}{d_m}. \tag{29}$$

Let $\hat{\mathbf{t}}_m$ denote the translation predicted by the final CameraHead refinement. Its signed progress along the ground-truth direction is

$$s_m^{\mathrm{prog}} = \hat{\mathbf{t}}_m^{\top}\mathbf{u}_m^{\Delta}. \tag{30}$$

The complete Directional Progress Loss is

$$\mathcal{L}_{\text{prog}} = \frac{\sum_m \mathbb{I}(d_m > \tau) \max\left(0,\, \rho d_m - s_m^{\text{prog}}\right)}{\sum_m \mathbb{I}(d_m > \tau) + \epsilon}. \tag{31}$$

We use $\tau = 0.1\,\text{m}$ and $\rho = 0.8$.

This auxiliary objective penalizes predictions with insufficient progress along the ground-truth local translation direction, including near-zero or oppositely directed predictions. Importantly, $\mathcal{L}_{\text{prog}}$ is a local geometric constraint on the pose branch and should not be interpreted as task-level progress toward the navigation goal.

D Training Sample Construction

We construct training samples from trajectories in R2R, RxR, ScaleVLN, and the additional closed-loop rollouts described in Appendix B. For each trajectory, we sample an action window of length $N = 16$ and predict an action chunk every $K = 4$ steps, resulting in $M = N/K = 4$ prediction segments per complete window.

For a window starting at timestep s, the m-th current state is

$$i_m = s + mK, \qquad m \in \{0, \ldots, M-1\}, \tag{32}$$

with future target

$$j_m = \min(i_m + K, T_{\text{end}}). \tag{33}$$

Each valid segment provides

$$V_{i_m} \longrightarrow \left(A_{i_m:j_m-1}, \Delta P_{i_m \to j_m}\right). \tag{34}$$

The local transition is encoded as its inverse world-to-camera transform for CameraHead supervision.

The pose annotations associated with the current observation and historical memory observations are expressed in the episode-relative coordinate system and encoded into pose latents. For non-initial windows, we additionally sample at most $N_h = 8$ historical observations:

$$\mathcal{M}_{i_m} = \left\{(V_{h_n}, P_{h_n})\right\}_{n=1}^{N_h}. \tag{35}$$

These historical observation-pose pairs provide navigation context for the current prediction segment but do not receive independent action or local-pose supervision.

The closed-loop rollouts expose the policy to observation and action sequences induced by its own predictions. The synchronized camera trajectory provides geometrically aligned pose annotations and local ΔPose targets for constructing the corresponding training samples.

E Implementation Details

PoseVLN is built upon StreamVLN under the single-RGB streaming setting. The navigation backbone is initialized from the corresponding StreamVLN checkpoint, while the pose encoder and VGGT CameraHead are initialized from scratch.

We use trajectory windows of length $N = 16$, at most $N_h = 8$ memory observations, and an action-prediction horizon of $K = 4$. The pose encoder maps the 9D camera representation to the 3,584-dimensional language-model embedding space. The VGGT CameraHead operates on a 2,048-dimensional representation with four transformer blocks, 16 attention heads, and four iterative refinements.

Training is performed on four GPUs using AdamW with a learning rate of 5×10^{-6}, zero weight decay, a cosine schedule with 3% warmup, BF16 precision, gradient checkpointing, and DeepSpeed ZeRO-2. The per-device batch size is 1 with 32 gradient-accumulation steps, giving an effective batch size of 128.

The action and pose loss weights are both 1.0. Within the pose objective, translation and quaternion regression each use unit weight. The inherited FoV weight is 0.5, but its loss is zero because the predicted FoV entries are overwritten with the target constants. For the Directional Progress Loss, we use $\tau = 0.1\,\text{m}$

Table 8. Collision-stratified mean episode-level translational ATE (m, lower is better) in two representative scenes. Groups are defined by Action-DR collision counts. Both variants use matched tasks without scale alignment.

Scene	Action-DR collisions	Episodes	Action-DR ATE (m)	Learned ΔPose ATE (m)
EU6Fwq7SyZv	0	61	**0.318**	1.099
	1–2	39	**0.309**	1.484
	≥3	32	18.249	**16.061**
	Overall	**132**	**4.662**	4.840
x8F5xyUWy9e	0	53	**0.263**	0.989
	1–2	18	**0.153**	1.710
	≥3	31	10.464	**6.377**
	Overall	**102**	3.344	**2.754**

and $\rho = 0.8$. Its coefficient is linearly increased from 0.05 to 0.5 during the first 1,920 supervised-training steps and fixed to 0.2 during subsequent DAgger refinement.

Unless otherwise specified, the remaining settings follow StreamVLN [36].

Appendices F–N present additional experiments, analyses, and clarifications included in the revised manuscript during the rebuttal period.

F Scene-Level Geometric Error Under Execution Discrepancies

The main text reports episode-level pose errors and navigation success (Table 6) and collision-stratified navigation results (Table 7). Here, we compare unaligned translational ATE of learned ΔPose and Action-based Dead Reckoning (Action-DR) in two representative scenes. Groups are defined by Action-DR collision counts. The variants share task identities and may follow different executed trajectories.

Table 8 shows lower ATE for Action-DR in collision-free and low-collision groups. For groups with at least three Action-DR collisions, learned ΔPose achieves lower ATE in both scenes (16.061 versus 18.249 m and 6.377 versus 10.464 m). The overall ATE comparison differs across scenes. Together with the navigation results in Table 7, these findings show the value of learned geometric feedback under larger execution discrepancies.

G Translation Underestimation and the Directional Progress Loss

We analyze translation underestimation and the effects of Directional Progress Loss on translation magnitudes and physical camera displacement, complementing the navigation ablation in Table 3.

Motivation and Experimental Setup. During exploratory R2R-train and RxR-train training with $\mathcal{L}_{\text{act}}$ and $\mathcal{L}_{\text{pose}}$, predicted translation magnitudes decreased at later checkpoints, which can contract recursively propagated trajectories. We compare intermediate and final checkpoints without the progress loss and a final checkpoint with the loss, using the same three checkpoints for magnitude and displacement analyses.

For each evaluated local prediction–target pair, let $\hat{d}_m = \|\hat{\mathbf{t}}_m\|_2$ and $d_m = \|\mathbf{t}_m^*\|_2$ denote the predicted and GT translation magnitudes, respectively. We report their means, $\bar{d}_{\text{pred}}$ and $\bar{d}_{\text{GT}}$, together with the difference of means and relative difference:

$$b = \bar{d}_{\text{pred}} - \bar{d}_{\text{GT}}, \qquad b_{\text{rel}} = \frac{\bar{d}_{\text{pred}} - \bar{d}_{\text{GT}}}{\bar{d}_{\text{GT}}} \times 100\%. \tag{36}$$

Negative values indicate aggregate magnitude underestimation. We also report the percentage of predictions with $\hat{d}_m < 0.1$ m. Each row includes the GT mean for its evaluated samples.

Observed Translation Underestimation. Table 9 shows that, without the progress loss, mean predicted translation decreases from 0.3410 to 0.1356 m while GT means remain near 0.65 m. The magnitude deficit increases from 47.55% to 79.23%, and predictions below 0.1 m rise from 18.40% to 64.32%.

Table 9. Translation underestimation and physical camera displacement diagnostics using the same three model checkpoints. **Top:** Translation magnitude statistics from exploratory training on R2R-train and RxR-train. Differences are computed between mean magnitudes, not per-sample absolute errors. **Bottom:** Physical displacement errors on a supplementary subset of 10 episodes and 157 four-step segments. Directional errors use 132 valid segments. The supplementary evaluations use independently generated rollouts for each checkpoint.

Model	Mean Pred. (m)	Mean GT (m)	Difference (m)	Relative Difference (%)	Pred. < 0.1 m (%)
Without progress, intermediate	0.3410	0.6501	−0.3091	−47.55	18.40
Without progress, final	0.1356	0.6527	−0.5171	−79.23	64.32
With progress, final	**0.5778**	0.6325	**−0.0547**	**−8.65**	**4.61**

Checkpoint	Mean Error (m)	RMSE (m)	Direction Error (°)
Without progress (1920)	0.3523	0.4153	16.64
Without progress (2560)	0.6125	0.6989	88.03
With progress (1920)	**0.1736**	**0.2009**	**11.01**

With the progress loss, the final checkpoint predicts 0.5778 m against a GT mean of 0.6325 m, reducing the deficit to 8.65% and predictions below 0.1 m to 4.61%. Together with the navigation ablation, this supports its effectiveness in mitigating translation underestimation.

Physical Camera Displacement Analysis. To further evaluate geometric accuracy in physical space, we examine the same checkpoints using camera-center displacement rather than inverse-extrinsic translation parameters. The predicted local world-to-camera transformation is converted into relative camera-center motion:

$$\hat{\mathbf{p}} = -\hat{\mathbf{R}}^\top \hat{\mathbf{t}}, \qquad \mathbf{p}^* = \left[(T_i^*)^{-1} T_{i+4}^* \right]_{:3,3}, \tag{37}$$

where $\hat{\mathbf{R}}$ and $\hat{\mathbf{t}}$ denote the predicted local world-to-camera rotation and translation, and T_i^* is the GT camera-to-world pose. This conversion incorporates the predicted rotation when estimating relative physical camera displacement.

For each segment, we measure the displacement vector error

$$e_{\text{disp}} = \|\hat{\mathbf{p}} - \mathbf{p}^*\|_2. \tag{38}$$

We report its mean and root-mean-square error (RMSE), together with the angle between the predicted and GT displacement vectors.

The supplementary evaluation covers 10 common episode identities and 157 four-step segments for each checkpoint. All 157 segments are included in the displacement error statistics. Directional errors are computed over 132 common segments with GT displacement greater than 0.1 m in all three evaluated rollouts, excluding negligible-motion segments for which displacement direction is unreliable.

As shown in Table 9, the with-progress checkpoint at step 1920 achieves a lower mean physical displacement error than the without-progress checkpoint at the same training step (0.1736 versus 0.3523 m), together with lower RMSE (0.2009 versus 0.4153 m). Mean directional error also decreases from 16.64° to 11.01°. The later without-progress checkpoint at step 2560 exhibits larger errors, reaching 0.6125 m in mean displacement error and 88.03° in directional error.

The progress-loss checkpoint improves both translation magnitudes and physical displacement estimates on the evaluated subset. Each checkpoint uses independently generated rollouts with shared episode identities; observations and executed trajectories can differ.

Connection to the Loss Formulation. The Directional Progress Loss penalizes insufficient projection onto the GT translation direction for targets with non-negligible translation:

$$\mathcal{L}_{\text{prog}} = \max\left(0, \rho d - \hat{\mathbf{t}}^\top \frac{\mathbf{t}^*}{d}\right), \qquad d = \|\mathbf{t}^*\|_2 > \tau. \tag{39}$$

Table 10. Navigation robustness under controlled visual perturbations on R2R-CE Val-Unseen. Performance changes are measured relative to the clean-input PoseVLN result.

Test Condition	SR (%) ↑	ΔSR (pp)
PoseVLN (Clean)	64.6	–
Brightness $\times 0.5$	62.5	-2.1
Gaussian Blur ($r = 2$)	58.4	-6.2
Action-only (Clean)	57.0	–

Table 11. Sources of current and historical pose context during optimization. All valid pose inputs are constructed from simulator-recorded camera poses.

Pose Source	Current (%)	Historical (%)
Ground-truth poses	100	100
Model-predicted poses	0	0

Targets with $d \leq \tau$ are excluded, so the objective does not encourage translation for negligible-motion targets. We use $\rho = 0.8$ and $\tau = 0.1$ m. Unlike a magnitude-only constraint, this term also accounts for the predicted translation's alignment with the target direction.

Directional Progress Loss regularizes translation underestimation in the adopted inverse-extrinsic parameterization; its direct constraint acts on translation projection, not camera-center motion. The reduced magnitude deficit and lower physical displacement errors, together with the navigation ablation, support its effectiveness under the evaluated conditions.

H Robustness to Controlled Visual Perturbations

We evaluate PoseVLN under two controlled visual perturbations: brightness reduction by a factor of 0.5 and Gaussian blur with radius 2. All other settings remain unchanged.

As shown in Table 10, SR changes from 64.6% on clean inputs to 62.5% with brightness $\times 0.5$ and 58.4% with Gaussian blur, decreases of 2.1 and 6.2 points.

Both conditions retain higher SR than the clean-input action-only baseline (57.0%). The larger degradation under blur indicates greater sensitivity to lost visual detail. These experiments measure robustness to the specified appearance changes.

I Pose Context and Closed-Loop Collection Audit

We distinguish pose context during optimization and DAgger collection, reporting ground-truth and predicted pose inputs and expert intervention statistics.

Pose Context During Optimization. During supervised optimization, all valid current and historical pose inputs are constructed from simulator-recorded camera poses. Therefore, 100% of the valid pose context is ground-truth-derived, while 0% is model-predicted. The same recorded camera trajectories are also used to construct the local ΔPose regression targets.

During policy-controlled DAgger rollouts, geometric states are recursively propagated using predicted ΔPose, exposing the policy to its own rollout distribution. Optimization continues to use recorded pose annotations, as summarized in Table 11.

Expert Intervention Frequency. During closed-loop collection, expert intervention is triggered when the agent deviates from the reference path by more than 1.0 m, when the normalized accumulated action-error score exceeds 0.5, when the accumulated number of action errors exceeds 8, or when the policy produces an invalid Stop action. The path-recovery controller returns control to the policy when the deviation falls below 0.5 m.

Table 12. Expert intervention statistics during closed-loop DAgger collection. The intervention rate denotes the percentage of completed episodes containing at least one expert takeover.

Dataset	Intervention Rate (%)	Mean Takeovers
R2R	96.68	1.09
RxR	88.50	1.01
Combined	91.32	1.04

Table 13. Statistical reliability and batch-one inference efficiency on R2R-CE Val-Unseen. Confidence intervals are estimated using 20,000 paired episode-level bootstrap resamples. Latency measurements report mean median and mean P95 end-to-end decision times.

Method	SR (%) [95% CI]	Mean Median Latency (ms)	Mean P95 Latency (ms)
StreamVLN	56.2 [53.94, 58.40]	546.97	767.83
PoseVLN	**64.6 [62.43, 66.78]**	560.12	781.09
Difference	+8.4 pp [5.87, 11.09]	+13.15 (+2.40%)	+13.26 (+1.73%)

Table 12 reports interventions in 96.68% of R2R episodes and 88.50% of RxR episodes, with 1.09 and 1.01 mean takeovers per episode. Combined, the rate is 91.32% with 1.04 takeovers per episode. Takeover counts exclude pose re-anchoring at policy return.

Action Conditioning and Pose Re-anchoring. The action decoder uses the shared multimodal representation, while CameraHead uses the current pose-token hidden state. The pose head predicts ΔPose jointly with actions and is not explicitly conditioned on the final generated or executed action sequence; it does not integrate decoded actions.

During expert–policy handoffs in DAgger collection, the internal pose state is re-anchored to the simulator-recorded camera pose in the original episode-relative coordinate frame. Unexecuted action predictions and stale recurrent states are discarded before policy control resumes. This prevents expert-executed motion from leaving the propagated state inconsistent with the subsequent visual observation.

Re-anchoring is limited to DAgger collection. During evaluation, non-initial poses are recursively predicted without GT correction or external odometry. The effects of predicted-context optimization and removing collection-time re-anchoring remain to be evaluated.

J Statistical Reliability and Batch-One Inference Efficiency

We assess navigation reliability using episode-level bootstrap analysis on R2R-CE and RxR-CE Val-Unseen, and inference efficiency using batch-one end-to-end decision latency.

Episode-Level Statistical Reliability. On R2R-CE Val-Unseen, we compare StreamVLN and PoseVLN using 20,000 paired bootstrap resamples of the same 1,839 episodes, preserving episode correspondence.

Statistical Reliability and Efficiency Results. Table 13 shows an 8.4-point SR gain over StreamVLN, with a paired 95% confidence interval of [5.87, 11.09] points, entirely above zero.

On RxR-CE English Val-Unseen, PoseVLN achieves 62.0% SR with a 95% confidence interval of [60.45, 63.58] across 3,669 evaluation episodes. These intervals quantify episode-level uncertainty for the evaluated checkpoints.

The navigation improvement comes with only 2.40% additional mean median latency and 1.73% additional mean P95 latency, corresponding to increases of 13.15 ms and 13.26 ms, respectively. Together, these results demonstrate a navigation gain with modest inference overhead under the evaluated batch-one setting.

Table 14. Relative end-to-end inference latency overhead of PoseVLN compared with StreamVLN under batch-one inference.

Latency Metric	Relative Overhead (%)
Mean median latency	+2.40
Mean P95 latency	+1.73

Table 15. Mean position error (m, lower is better) across prediction horizons without per-pair translation scale alignment. VGGT uses a single global translation scale calibrated on R2R training data and shared across all evaluated horizons and benchmarks.

Horizon H	PoseVLN error (m)	VGGT (fixed scale) error (m)
4	0.2568	**0.2547**
8	**0.5049**	0.6772
12	**0.7863**	1.0563
16	**1.1092**	1.4348

Batch-One Inference Efficiency. Table 14 summarizes batch-one end-to-end latency overhead relative to StreamVLN: 2.40% in mean median latency and 1.73% in mean P95 latency. These results quantify the added inference cost of pose encoding and local pose prediction.

K Fixed-Scale Position Error Comparison with VGGT

We complement the per-pair scale-aligned comparison in Table 4 with fixed-scale position errors over $H \in \{4, 8, 12, 16\}$. VGGT uses one global translation scale calibrated on R2R training data, fixed across evaluation pairs and horizons.

Fixed-Scale Calibration Protocol. We estimate a single global translation scale using 103,300 moving H=4 image pairs from 10,819 reference trajectories across 61 R2R training scenes. We verified that all calibration episode IDs belong to the R2R training split. The scale is obtained by minimizing the translation error over the calibration set $\mathcal{C}$:

$$s^* = \arg\min_{s>0} \sum_{p \in \mathcal{C}} \left\| s\mathbf{t}_p^{\mathrm{VGGT}} - \mathbf{t}_p^{\mathrm{GT}} \right\|_2^2 = 1.7772. \tag{40}$$

Here, $\mathbf{t}_p^{\mathrm{VGGT}}$ and $\mathbf{t}_p^{\mathrm{GT}}$ denote the VGGT-predicted and GT translation vectors, respectively. The resulting scale is shared across all evaluated horizons $H \in \{4, 8, 12, 16\}$ and both R2R-CE and RxR-CE evaluations. PoseVLN retains its original metric-scale predictions without additional scaling.

Evaluation GT is used for position-error measurement and moving-pair selection, never for scale fitting. Scale calibration uses R2R training trajectories that are separate from the evaluation episodes.

Fixed-Scale Position Error Results. As shown in Table 15, the two methods have comparable mean position error at $H = 4$ (0.2568 m for PoseVLN and 0.2547 m for VGGT). At longer horizons, PoseVLN achieves lower errors: 0.5049 versus 0.6772 m at $H = 8$, 0.7863 versus 1.0563 m at $H = 12$, and 1.1092 versus 1.4348 m at $H = 16$. Both methods exhibit increasing error with horizon, while PoseVLN shows lower mean position errors at $H \in \{8, 12, 16\}$ under this fixed-scale evaluation.

Observation Protocols and Interpretation. PoseVLN predicts causally from intermediate RGB observations, history, and propagated poses; VGGT uses endpoint image pairs. These are distinct observation protocols. The fixed-scale results provide complementary evidence of PoseVLN's longer-horizon geometric accuracy without per-pair scale alignment.

Table 16. Auxiliary geometric supervision versus full pose-conditioned navigation on RxR-CE English Val-Unseen. Both variants use pose-related training objectives; only full PoseVLN uses recursively predicted pose tokens as navigation context.

Model	SR (%) ↑	SPL (%) ↑	NE (m) ↓
Auxiliary-only (no pose input)	56.0	51.4	5.4
Full PoseVLN (predicted pose)	**62.0**	**53.5**	**4.8**

Table 17. Real-world navigation results on Unitree Go2 across an outdoor scene (Scene 1) and an indoor scene (Scene 2). Each task setting is evaluated three times with slightly varied starting positions. SR reports the number of successful trials out of three. S/F indicates whether at least two of the three trials succeed.

ID	Instruction	SR	S/F
Scene 1 (Outdoor)			
1	Walk straight toward the open doorway, pass through it, and stop once you have completely left the room.	3/3	S
2	Walk straight toward the open doorway and pass through it. Once you have completely left the room, turn right and stop.	2/3	S
3	Walk straight toward the open doorway, pass through it, and stop in front of the wall directly opposite the doorway.	0/3	F
Scene 2 (Indoor)			
4	Walk straight through the open doorway into the hallway, turn left to face down the hallway after you have completely left the room, and stop once you have completed the left turn.	3/3	S
5	Walk straight through the open doorway ahead and to your right, and stop once you have completely entered the room.	0/3	F
6	Walk straight toward the open doorway, pass through it, and stop once you have completely left the room.	3/3	S
Total	**Six task settings**	**11/18**	**4/6**

L Auxiliary Geometric Supervision Ablation on RxR-CE

We extend the ablation in Table 3 to RxR-CE English Val-Unseen. Auxiliary-only retains action, pose, and directional progress objectives without numerical pose inputs, while full PoseVLN uses pose-conditioned memory and recursive pose feedback.

Table 16 shows improvements from auxiliary-only to full PoseVLN: SR 56.0% to 62.0%, SPL 51.4% to 53.5%, and NE 5.4 to 4.8 m. Together with the R2R-CE comparison in the updated Table 3 (57.5% versus 64.6% SR), these results establish benefits on both benchmarks.

M Real-World Navigation Evaluation

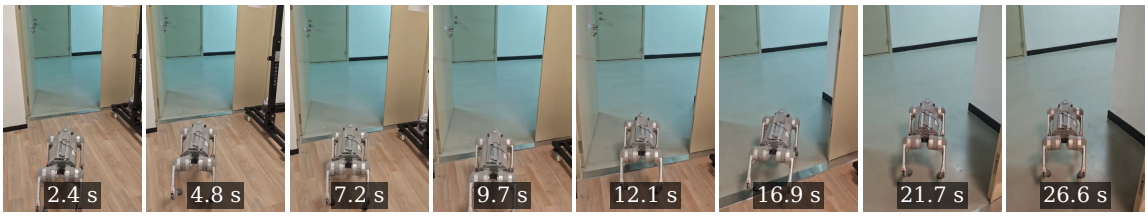
Evaluation Setup. We evaluate PoseVLN on Unitree Go2 in outdoor (Scene 1) and indoor (Scene 2) environments containing doorways, corridors, and obstacles. Each scene includes three tasks covering room exit, turning, stopping, and room entry. Three trials per task with varied starting positions yield 18 trials across six settings. Tasks 1 and 6 use the same instruction in different scenes.

Quantitative Results. Table 17 reports 11 successes in 18 trials (61.1%), with at least two successful trials in four of six task settings. Tasks 3 and 5 yield no successes in three trials each, involving wall-target stopping and room entry, respectively.

Trajectory Visualization. Figures 5 and 6 visualize representative recordings from the outdoor scene (Tasks 1–3) and the indoor scene (Tasks 4–6), respectively. Each figure contains three rows, with one recorded trial per task. Each row presents eight selected video frames in chronological order from left to right, together with the complete navigation instruction. The timestamps indicate elapsed time in the source video; the selected frames are not necessarily equally spaced.

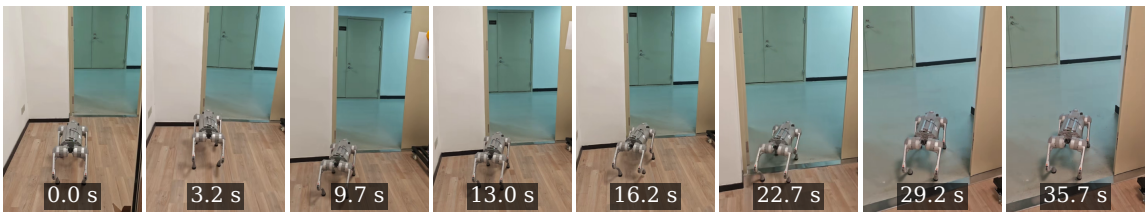
Scene 1

Task 1



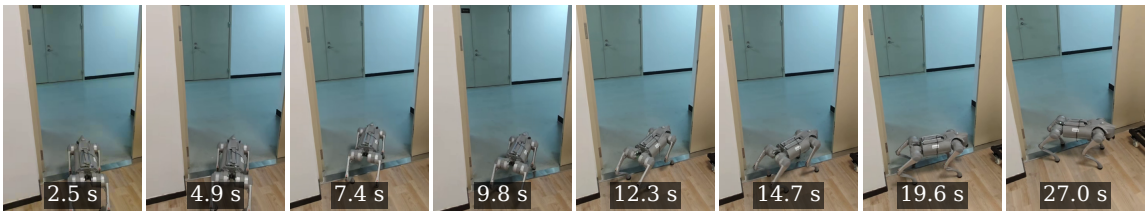
Instruction: Walk straight toward the open doorway, pass through it, and stop once you have completely left the room.

Task 2



Instruction: Walk straight toward the open doorway and pass through it. Once you have completely left the room, turn right and stop.

Task 3



Instruction: Walk straight toward the open doorway, pass through it, and stop in front of the wall directly opposite the doorway.

Figure 5. Representative real-world navigation trajectories of PoseVLN on Unitree Go2 in the outdoor scene (Scene 1). The three rows correspond to room exit (Task 1), room exit followed by a right turn (Task 2), and stopping at the wall opposite the doorway (Task 3). Frames are ordered chronologically from left to right, with source-video timestamps in seconds. The full instruction is shown below each trajectory.

The recordings show room exit, turning, targeted stopping, and attempted room entry, including the unsuccessful settings in Table 17. The figures show individual trials; the table aggregates all three repetitions per setting.

Discussion. The 11 successful trials (61.1%) demonstrate RGB-based language-guided navigation on Unitree Go2 in outdoor and indoor settings, including doorway traversal and turning. Tasks 3 and 5 remain challenging, with no successful trials for wall-target stopping or specified room entry. These results establish feasibility across the evaluated tasks and motivate broader tests of reliability across environments and instructions.

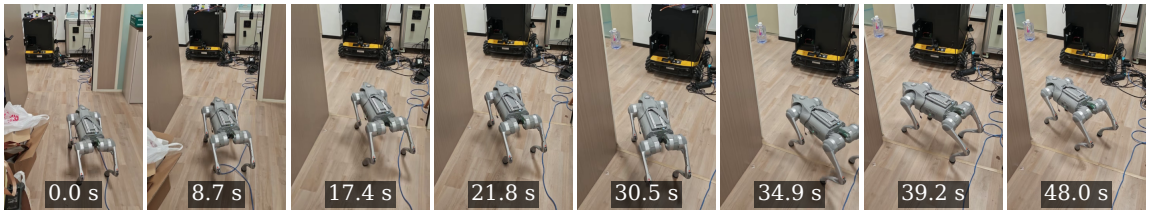
N Availability of R2R-CE Test Evaluation

The official VLN-CE test evaluation server was retired in January 2026, and ground-truth test goals are not publicly available. The organizers recommend using the Val-Unseen split for future method comparisons[1]. Following this recommendation, we report results on R2R-CE and RxR-CE Val-Unseen, achieving 64.6% and 62.0% SR, respectively.

[1]https://eval.ai/web/challenges/challenge-page/719

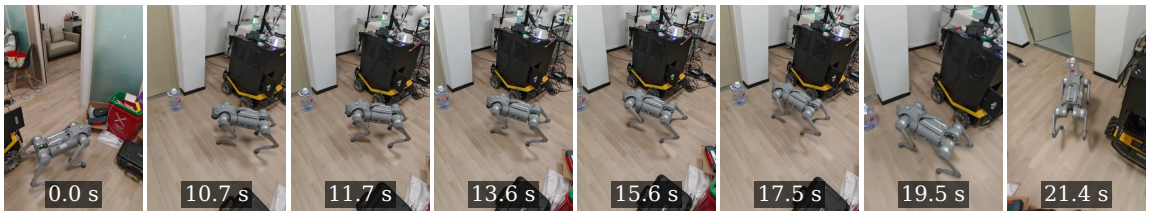
Scene 2

Task 4



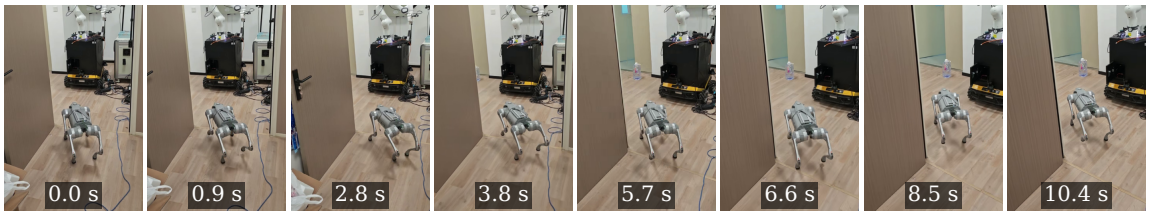
Instruction: Walk straight through the open doorway into the hallway, turn left to face down the hallway after you have completely left the room, and stop once you have completed the left turn.

Task 5



Instruction: Walk straight through the open doorway ahead and to your right, and stop once you have completely entered the room.

Task 6



Instruction: Walk straight toward the open doorway, pass through it, and stop once you have completely left the room.

Figure 6. Representative real-world navigation trajectories of PoseVLN on Unitree Go2 in the indoor scene (Scene 2). The three rows correspond to room exit followed by a left turn (Task 4), entering the room through the doorway ahead and to the right (Task 5), and room exit (Task 6). Frames are ordered chronologically from left to right, with source-video timestamps in seconds. The full instruction is shown below each trajectory.