CANVAS-VLN: FROM FUTURE PLANS TO EXECUTION-GROUNDED MEMORIES FOR VISION-LANGUAGE NAVIGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-and-Language Navigation (VLN) requires an agent to make long-horizon, closed-loop decisions in continuous environments by following natural-language instructions. Recent methods translate the outputs of vision-language models into executable local plans by predicting action chunks, image-space goal points, or pixel trajectories. However, existing trajectory-based approaches typically treat trajectories as disposable action outputs, while representing historical context using sequences of raw observations or textual action chunks. Consequently, future planning and past execution experience reside in different representation spaces. We present Canvas-VLN, a unified canvas–trajectory navigation framework in which pixel trajectories serve three roles: supervision targets, online plans, and navigation memory. The model predicts a future pixel trajectory on the current navigation canvas, and a local controller executes a short prefix of the predicted trajectory. The waypoints actually traversed by the agent are then reprojected onto historical anchor images to construct a Gated Execution-Trajectory Memory. This memory adaptively commits local trajectory segments according to visibility, waypoint budget, and turning events, while explicit turn states connect consecutive historical anchors. It thereby maintains a long-horizon, execution-grounded navigation context under a fixed visual-memory budget. Canvas-VLN thus establishes a predict–execute–write-back loop, in which a predicted future trajectory becomes historical trajectory evidence after execution and subsequently informs the next planning step. Trained solely on the official R2R-CE and RxR-CE navigation demonstrations, without additional training data or DAgger augmentation, Canvas-VLN achieves a Success Rate (SR) of 53.83% on R2R-CE Val-Unseen and 60.56% on RxR-CE Val-Unseen, establishing a new state of the art on RxR-CE among methods trained exclusively on task-specific navigation data.

1 INTRODUCTION

Vision-and-Language Navigation in Continuous Environments (Anderson et al., 2018b; Krantz et al., 2020) requires an embodied agent to translate a natural-language instruction into a sequence of closed-loop decisions under partial visual observability. Beyond grounding the next action in the current observation, the agent must continuously infer which parts of the instruction have already been completed and how its executed motion has changed the navigation state. This requirement becomes increasingly challenging over long trajectories, where local planning errors, camera rotations, and repeated observations make execution progress difficult to recover from visual history alone.

Recent VLM-based navigators have progressively shifted from predicting discrete actions or action chunks to planning directly in image space (Zhang et al., 2024; Cheng et al., 2025). Pixel-goal methods ground navigable waypoints in the current observation, establishing a spatial interface between visual reasoning and robot control (Bao et al., 2026), while pixel-trajectory methods further describe the local interaction process leading toward a waypoint. By representing motion as an ordered sequence of image coordinates, trajectory prediction provides richer geometric supervision and a more executable local plan than an isolated target point (Fu et al., 2026).

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.

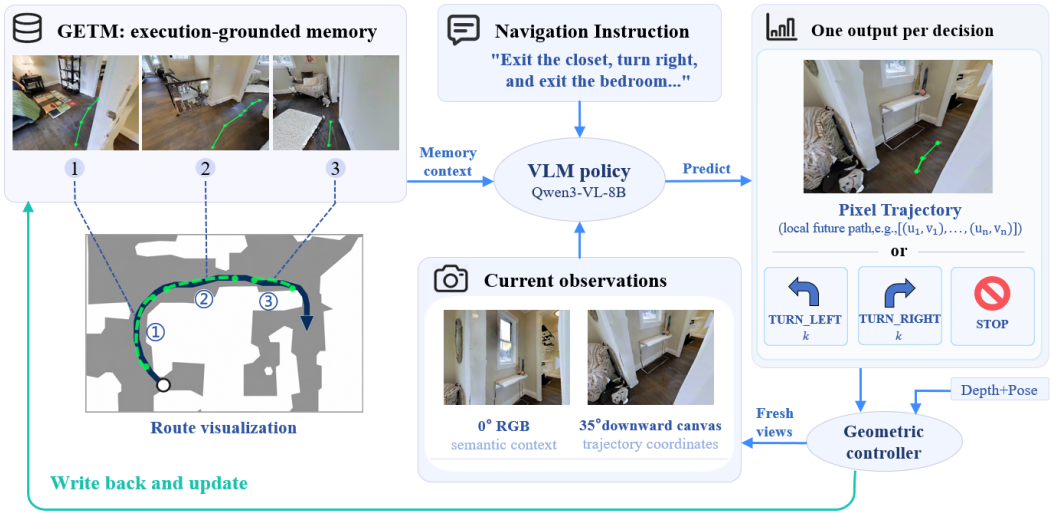

Figure 1: Canvas-VLN: Navigation with an evolving record of execution.

Nevertheless, this progress has primarily changed how an agent represents its future motion. Historical context is still commonly represented using raw observations, selected keyframes, compressed visual tokens, or textual action records (Wei et al., 2026b; Yang et al., 2026). These representations preserve what the agent has observed, but leave the motion actually executed in each observation implicit. As a result, a predicted trajectory is discarded after execution, and the model must reconstruct navigation progress from heterogeneous visual and action histories. We refer to this discrepancy as an open representation loop between future planning and past execution.

Our key insight is that a pixel trajectory need not be a disposable action output. It can instead serve as a shared visual language for both intended future motion and realized past experience.

Based on this insight, we introduce Canvas-VLN, a unified canvas-trajectory framework that closes the loop between prediction, execution, and memory. At each decision step, the model receives two complementary current views: a horizontal RGB image that preserves distant landmarks and semantic context, and a 35° pitched-down trajectory canvas that provides a stable image coordinate frame for local motion. Conditioned on these views, the navigation instruction, and the execution history, the model generates exactly one of three output types: an ordered pixel trajectory for forward motion, `TURN_LEFT` k or `TURN_RIGHT` k for k consecutive 15° rotations, or STOP. The explicit grammar ensures that pixel coordinates always refer to the trajectory canvas and that non-translational actions remain unambiguous.

The central component of Canvas-VLN is Gated Execution-Trajectory Memory (GETM). Rather than writing a predicted plan directly into memory, the controller first executes only a short prefix of the plan. The positions actually reached by the agent are then reprojected into the coordinate frame of a historical anchor observation and rendered as an ordered trajectory overlay. Consequently, each memory image records not only what was visible at that location, but also how the agent actually moved away from it.

A single anchor cannot represent an arbitrarily long route because the executed path eventually leaves its visible region or undergoes a substantial change in orientation. GETM therefore maintains a local execution segment until a deterministic gating event is reached, including loss of visibility, exhaustion of the trajectory-point budget, or a sufficiently large accumulated turn. The segment is then committed and a new anchor is created. Natural-language yaw transitions connect consecutive anchors, while a compact short-horizon turn trace records recent rotations that have not yet resulted in visible translational motion. Under a fixed visual-memory budget, this design represents long navigation histories as a temporally ordered sequence of local trajectory coordinate frames.

Canvas-VLN thus forms a closed-loop predict–execute–write-back process. An expert future path first appears as a trajectory supervision target. During inference, the model converts the same form

of trajectory into a local plan. After partial execution, the realized portion of that plan becomes historical trajectory evidence that informs the next decision.

We train Canvas-VLN through supervised fine-tuning on the two most prominent VLN-CE (Krantz et al., 2020)benchmarks: R2R-CE and RxR-CE (Anderson et al., 2018b; Ku et al., 2020), without synthetic navigation data, reinforcement learning, or DAgger (Ross et al., 2011). The resulting model achieves an SR of 53.83% on R2R-CE val-unseen and 60.56% on RxR-CE val-unseen. These results demonstrate that the proposed trajectory-centered interface supports robust navigation performance across both concise and long-form navigation instructions.

In summary, our contributions are threefold:

- We formulate continuous VLN as a trajectory-centered closed-loop process in which translational motion shares a common canvas representation across expert supervision, online planning, and execution-grounded memory.
- We introduce GETM, which writes actually executed trajectories onto adaptively committed historical anchors and preserves cross-anchor orientation through explicit yaw transitions and short-horizon turn states under a fixed visual-memory budget.
- Comprehensive experiments on VLN-CE benchmarks demonstrate that Canvas-VLN achieves competitive short-horizon navigation performance and state-of-the-art long-horizon performance without additional training data. These results validate execution-grounded trajectory write-back as an effective memory principle for continuous VLN.

2 Related work

2.1 Representing navigation history

Vision-and-Language Navigation (VLN) requires an embodied agent to follow natural-language instructions through multi-step interaction with partially observable environments (Anderson et al., 2018b; Krantz et al., 2020). Because each observation reveals only a limited portion of the environment, representing navigation history is crucial for retaining previously observed landmarks, tracking instruction progress, and grounding current decisions in executed motion. Prior methods encode history through recurrent cross-modal states, sequences of egocentric frames, or explicit long-horizon visual encoders (Hong et al., 2021; Chen et al., 2021; Zhang et al., 2024). Other approaches construct structured scene or map memories (Wang et al., 2023a; Zhang et al., 2025b; Yang et al., 2026; Huang et al., 2026a), while recent video-based policies compress long histories through multi-timescale visual tokens, slow–fast contexts, or separate visual-semantic and spatial-geometric memories (Zhang et al., 2025a; Wei et al., 2026b; Zeng et al., 2026).

Most closely related to our history representation, Goal2Pixel overlays the traversed path on selected historical keyframes, demonstrating that executed motion provides useful context beyond raw visual appearance (Bao et al., 2026). In Goal2Pixel, however, past motion annotates retained keyframes, whereas future planning is expressed as a single goal pixel. Canvas-VLN instead organizes navigation history as bounded, execution-grounded trajectory segments on historical visual anchors. It thereby represents realized motion in the same pixel-trajectory form used for local planning.

2.2 Pixel-space planning in continuous environments

Continuous VLN requires language-conditioned decisions to be converted into physically executable motion (Krantz et al., 2020). Earlier methods directly predict primitive actions or action chunks, or introduce learned waypoint interfaces between high-level instruction following and low-level control (Krantz et al., 2021; Hong et al., 2022; Zhang et al., 2024; 2025a). Recent approaches increasingly move this interface into the image plane through image-grounded subgoal prediction (Wei et al., 2026a; Bao et al., 2026). Goal2Pixel, for example, predicts a visible navigable pixel and recovers a three-dimensional waypoint through camera geometry. However, an individual goal pixel conveys limited information about the path leading to that destination.

Trajectory-level approaches instead represent future motion with multiple ordered waypoints (Shi et al., 2026; Fu et al., 2026). Most directly, Traj-VLN autoregressively generates image-space trajectories rather than isolated pixel goals, providing richer geometric supervision and local motion

intent. Canvas-VLN builds on this trajectory-based planning paradigm but extends pixel trajectories beyond transient planning outputs: predicted trajectories guide local execution, while the motion actually completed by the agent is written back as historical trajectory evidence. This creates a trajectory-centered predict–execute–write-back loop that connects future planning with past execution.

3 METHOD

3.1 TASK DEFINITION AND METHOD OVERVIEW

Vision-and-language navigation in continuous environments (VLN-CE) requires an embodied agent to follow a natural-language instruction $\mathcal{I}$ and reach an unobserved goal using its onboard observations. At decision step t, a typical VLN policy reasons over the current observation O_t and a history of past observations $O_{<t}$, and predicts a structured plan, an action chunk, or one or more pixel-space navigation targets. A VLN policy must therefore jointly understand the instruction, current visual context, and navigation history, and repeatedly predict the next action until it reaches the target location. This perception–prediction–execution process is repeated until the agent terminates the episode. An episode is successful only when the agent explicitly stops within $3\,\mathrm{m}$ geodesic distance of the goal.

Canvas-VLN operates at two temporal resolutions. The index t denotes VLM decision steps, whereas τ denotes low-level environment actions executed between decisions t and $t+1$. At each decision step, the VLM receives the instruction, bounded GETM history, recent rotation trace, and two current visual views. It predicts a pixel trajectory, a rotation command, or termination. The controller executes the prediction, and only the motion actually realized by the agent is written into GETM before the next decision.

3.2 GATED EXECUTION-TRAJECTORY MEMORY

Gated Execution-Trajectory Memory (GETM) represents navigation history by drawing the agent's actually executed trajectory on historical egocentric observations. Unlike a predicted trajectory, which may be only partially executed because of collisions or local-control deviations, the executed trajectory records the motion actually realized by the agent.

At the beginning of an episode, the initial 35° RGB-D observation defines an active memory anchor a. The anchor stores its RGB image $O_a^{35^\circ}$, depth map $D_a^{35^\circ}$, and world-to-camera transformation $\mathbf{T}_{a\leftarrow\mathrm{w}}$. As the agent moves, its executed world positions are sampled according to accumulated translation with an interval of $\delta = 0.5\,\mathrm{m}$. A sampled position $\mathbf{x}_\tau^{\mathrm{w}}$ is projected into the active anchor as

$$\mathbf{p}_\tau^a = \pi\left(\mathbf{K}\,[\mathbf{I}_3 \quad \mathbf{0}]\,\mathbf{T}_{a\leftarrow\mathrm{w}}\begin{bmatrix}\mathbf{x}_\tau^{\mathrm{w}}\\ 1\end{bmatrix}\right), \quad H_a \;= \mathrm{Draw}\left(O_a^{35^\circ}, E_a\right), \tag{1}$$

where K is the camera intrinsic matrix, $\pi(\cdot)$ denotes perspective division, and $E_a = (\mathbf{p}_1^a, \ldots, \mathbf{p}_{|E_a|}^a)$ is the ordered sequence of visible execution points. A point is visible if it has positive depth, lies inside the valid image region, and agrees with the depth stored at the anchor. Connecting these points produces the execution-trajectory image H_a, which describes how the agent actually moved after observing $O_a^{35^\circ}$.

The active anchor is retained only while it can reliably represent subsequent motion. Specifically, GETM applies the following deterministic gating rule:

$$g_a = \begin{cases} \textsc{Commit}(\text{large-turn}), & |\Delta\psi_a| \geq \psi_{\max}, \\ \textsc{Commit}(\text{point-budget}), & |E_a| \geq N_p, \\ \textsc{Commit}(\text{out-of-view}), & V_a(\mathbf{x}_{\mathrm{new}}^{\mathrm{w}}) = 0, \\ \textsc{Keep}, & \text{otherwise} \end{cases}, \tag{2}$$

where $\Delta\psi_a$ is the accumulated same-direction heading change, $\psi_{\max} = 75^\circ$, $N_p = 8$ is the point budget, and $V_a(\cdot)$ denotes the visibility test. The first condition is evaluated when an in-place rotation chunk ends, whereas the latter conditions are evaluated when a new translation sample is produced.

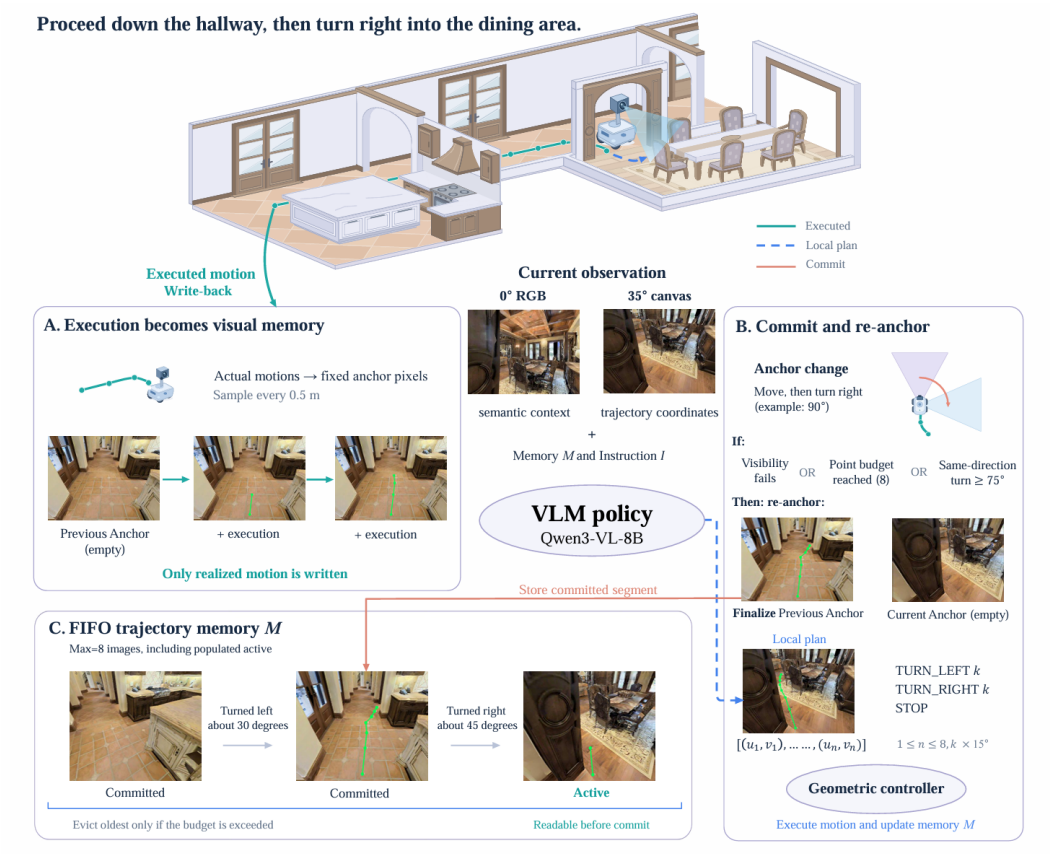

Figure 2: **Overview of Gated Execution-Trajectory Memory (GETM).** Actually executed positions are projected and drawn on an active egocentric anchor. A deterministic gate commits the trajectory image upon a large turn, point-budget exhaustion, or loss of visibility, after which the current observation initializes a new anchor. Textual turn transitions connect successive anchor frames, forming a bounded, temporally ordered execution memory.

When the gate returns COMMIT, the current trajectory image is finalized and the resulting after-action observation becomes a new anchor. Subsequent execution points are represented in this new local coordinate frame. If the heading changes between two anchors, GETM inserts a textual transition $\Delta_{a_i \to a_{i+1}}$, such as "Turned left about 90 degrees." The trajectory image records where the agent moved within an anchor, while the transition describes how two consecutive anchor coordinate frames are related.

At decision step t, the bounded GETM is

$$\mathcal{M}_t = \left[H_{a_1}, \Delta_{a_1 \to a_2}, H_{a_2}, \ldots, \Delta_{a_{m-1} \to a_m}, H_{a_m}\right], \qquad m \le N_m, \tag{3}$$

where $N_m = 8$ is the maximum number of memory images. A transition is omitted when no meaningful heading change occurs. When the memory budget is exceeded, the oldest committed segment is removed. GETM is thus bounded, causal, and constructible online without future expert information.

3.3 CANVAS-VLN ARCHITECTURE

As illustrated in Fig. 2, Canvas-VLN performs navigation through a repeated perception–prediction–execution–memory cycle. Before decision t, the model receives

$$\mathcal{X}_t = \left(\mathcal{I}, \mathcal{M}_t, S_t, O_t^{0^\circ}, C_t^{35^\circ}\right), \tag{4}$$

where $\mathcal{I}$ is the instruction, $\mathcal{M}_t$ is GETM, and S_t is a short-horizon turn trace containing up to six rotations executed since the most recent translation. The two current visual inputs have comple-

mentary roles: $O_t^{0^\circ}$ is a horizontal RGB image that provides distant semantic and landmark context, whereas $C_t^{35^\circ} = \mathrm{Canvas}(O_t^{35^\circ})$ is constructed from the 35°-pitched RGB image and defines the only coordinate frame in which trajectory points may be predicted.

Canvas-VLN uses a single autoregressive VLM to generate one of four mutually exclusive outputs:

$$Y_t \in \{[(u_1, v_1), \ldots, (u_n, v_n)], \quad \texttt{TURN_LEFT}\ k, \quad \texttt{TURN_RIGHT}\ k, \quad \texttt{STOP}\}. \tag{5}$$

Here, $1 \le n \le 8$ denotes the number of predicted waypoints. The first form is an ordered forward trajectory containing at most eight points in the valid region of $C_t^{35^\circ}$. The two turn commands represent k consecutive 15° in-place rotations, with the direction specified explicitly. The final command terminates navigation. During inference, predicted trajectory pixels are converted from canvas coordinates to the underlying 35° RGB coordinates and back-projected using the current depth observation. The resulting 3D waypoints are validated for local navigability and passed to controller. The controller executes a configurable prefix of the trajectory before requesting another VLM decision. A turn command directly executes the specified rotation primitives, while `STOP` ends the episode. Let $\widehat{\tau}_t^{\mathrm{exec}}$ denote the complete action-level sequence actually executed between decisions t and $t+1$, including the resulting positions, headings, and 35° RGB-D observations. GETM is updated after every executed environment action:

$$\mathcal{M}_{t+1} = \mathrm{GETMUpdate}\left(\mathcal{M}_t, \widehat{\tau}_t^{\mathrm{exec}}\right). \tag{6}$$

Therefore, unexecuted portions of the predicted trajectory are never written into memory. After the update, the next horizontal observation $O_{t+1}^{0^\circ}$ and trajectory canvas $C_{t+1}^{35^\circ}$ are constructed, and the same decision cycle is repeated.

3.4 Training supervision and data construction

We train Canvas-VLN on 1,247,361 decision-level SFT samples collected from 30,815 navigation episodes. We construct expert demonstrations exclusively from the official training splits of R2R-CE and RxR-CE. Decision states are created at the initial and terminal poses, at the boundaries of consecutive in-place rotation chunks, and after every four forward actions. At these states, we store a 35°-pitched RGB-D observation and a horizontal RGB. Depth and camera geometry are used to construct supervision and execute predicted trajectories, but are not provided to the vision-language model. The supervision target is determined by the next expert behavior. A consecutive rotation chunk is represented as `TURN_LEFT` k or `TURN_RIGHT` k, where k is the number of 15° rotations, and the terminal state is represented as `STOP`. For forward motion, future expert positions are sampled at approximately $0.5\,\mathrm{m}$ intervals, projected onto the current trajectory canvas, and filtered using image bounds and depth consistency. At most eight ordered trajectory points are retained.

Gated execution-trajectory memory (GETM) supervision is constructed by replaying each expert trajectory through the same memory update procedure used during inference. After every expert environment action, the memory is updated using the resulting position, heading, and 35° RGB-D observation. The current memory snapshot is then stored at every decision state. The GETM used during inference records motion actually executed by the learned policy.

4 Experiments

4.1 Experimental setup

Simulation environments and metrics. We evaluate Canvas-VLN on the R2R-CE Val-Unseen and RxR-CE Val-Unseen, two standard continuous-environment VLN benchmarks. Both benchmarks use real indoor environments from Matterport3D (Chang et al., 2017) and are executed in Habitat (Savva et al., 2019). Following prior work, we report Navigation Error (NE), Oracle Success (OS), Success Rate (SR), Success weighted by Path Length (SPL) (Anderson et al., 2018a), and normalized Dynamic Time Warping (nDTW) (Ilharco et al., 2019).

Real-world evaluation setup. We deploy Canvas-VLN on a custom-built mobile robot running ROS 2 Humble, with two Intel RealSense D455 RGB-D cameras providing the horizontal semantic and downward-tilted trajectory views. An onboard laptop with an NVIDIA RTX 5090 runs the Canvas-VLN model, while the geometric controller converts predicted pixel trajectories into robot motion commands.

4.2 MAIN RESULTS

Results on VLN-CE benchmark. We compare against panoramic methods (Krantz et al., 2021; Hong et al., 2022; Wang et al., 2023a), geometry-assisted methods (Zhang et al., 2025b; Wang & Lee, 2025; Liu et al., 2025; Long et al., 2025; Chen et al., 2025; Yao et al., 2025; 2026), and single-RGB methods (Wang et al., 2025; Zhang et al., 2024; 2025a; Cheng et al., 2025; Wei et al., 2026b; Yang et al., 2026; Zeng et al., 2026; Zhang et al., 2026; Bao et al., 2026; Fu et al., 2026) in Table 1. On the relatively short-horizon R2R-CE benchmark, Canvas-VLN achieves competitive performance without using additional training data, obtaining an SR of 53.8% and an SPL of 48.0%. Its SR remains within 1.7 percentage points of the strongest prior method trained without additional data. These results indicate that introducing explicit execution memory preserves competitive navigation accuracy on shorter routes.

The advantage of Canvas-VLN becomes substantially more pronounced on RxR-CE, which contains longer trajectories and requires more sustained instruction-progress tracking. Among methods trained only on the standard R2R-CE and RxR-CE data, Canvas-VLN establishes a new state of the art across all reported metrics, achieving SR of 60.6% and SPL of 49.8%. Compared with the strongest previous additional-data-free result for each metric, Canvas-VLN reduces NE by 0.77 m and improves SR, SPL, and nDTW by 7.3, 4.4, and 4.3 percentage points, respectively. Notably, it also outperforms all listed methods that use additional training data on the RxR-CE metrics. This performance pattern suggests that the benefit of explicit execution-trajectory memory is more evident under the longer-horizon navigation demands of RxR-CE. All methods share the same navmesh-assisted waypoint controller F.

Real-world qualitative results. Figure 3 presents two representative successful real-world navigation runs, with one route exceeding 25 m and the other exceeding 15 m. By condensing the executed navigation history into trajectory memory, GETM enables our model to effectively integrate past motion with current egocentric observations and successfully complete these long-horizon tasks. Additional analyses are provided in the Appendix E.

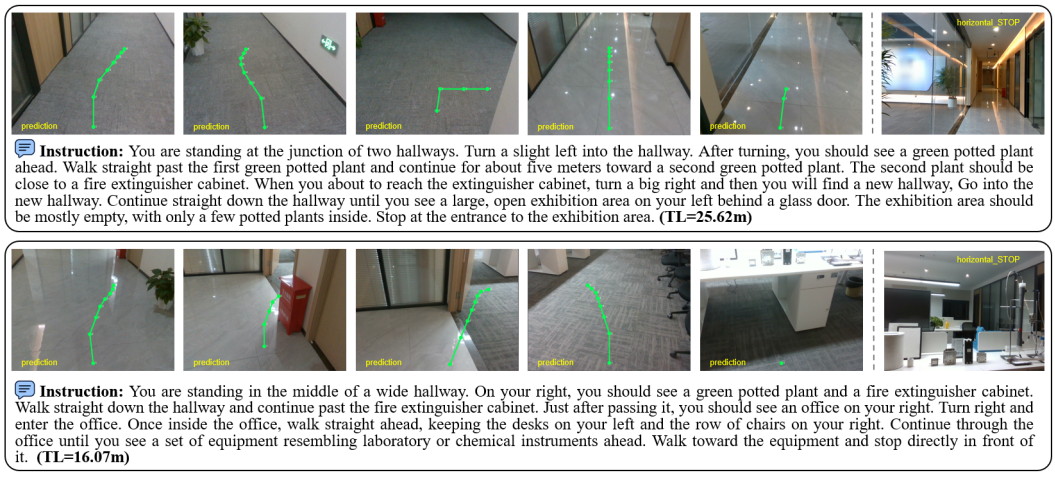

Figure 3: Qualitative results of Canvas-VLN in the real-world.

4.3 ABLATION STUDY

Contributions of execution memory and horizontal context. Table 2 evaluates the main components of Canvas-VLN under a matched evaluation protocol. Replacing GETM with raw anchor observations consistently degrades all metrics on both datasets. GETM improves SR/SPL by 1.74/2.04 points on R2R-CE and by 1.61/1.49 points on RxR-CE. It also improves nDTW by 0.98 and 1.31 points, respectively, indicating that encoding the executed trajectory provides a more informative navigation history than retaining the same anchor observations alone.

Table 1: Comparison with SOTA methods on R2R-CE and RxR-CE Val-Unseen. Additional data includes training data beyond the standard R2R/RxR-CE expert trajectories (e.g., ScaleVLN, EnvDrop, DAgger, and general VQA data (Wang et al., 2023b; Tan et al., 2019; Ross et al., 2011)). Following Zeng et al. (2026), a training sample is counted at the action level for navigation data or as a QA pair for VQA data. Generic foundation-model pre-training data is excluded.

Method	R2R Val-Unseen				RxR Val-Unseen				Training
	NE↓	OS↑	SR↑	SPL↑	NE↓	SR↑	SPL↑	nDTW↑	Additional Data
HPN+DN [ICCV'21]	6.31	40.0	36.0	34.0	–	–	–	–	–
CWP-CMA [CVPR'22]	6.20	52.0	41.0	36.0	–	–	–	–	–
VLN↻BERT [CVPR'22]	5.74	53.0	44.0	39.0	–	–	–	–	–
GridMM [ICCV'23]	5.11	61.0	49.0	41.0	–	–	–	–	–
MapNav [ACL'25]	4.93	53.0	39.7	37.2	–	–	–	–	–
g3D-LF [CVPR'25]	5.70	59.5	47.2	34.6	–	–	–	–	–
NaVid-4D [ICRA'25]	5.99	55.7	43.8	37.1	–	–	–	–	$>$ 1507K
InstructNav [CoRL'24]	6.89	–	31.0	24.0	–	–	–	–	0K
CA-Nav [TPAMI'25]	7.58	48.0	25.3	10.8	10.37	19.0	6.0	13.5	0K
NavMorph [ICCV'25]	5.75	56.9	47.9	33.2	8.85	30.8	22.8	44.2	–
SC^2-WM [ICML'26]	5.37	58.8	50.9	37.2	8.36	35.8	27.2	44.9	–
Sim2Real [CoRL'24]	5.95	55.8	44.9	30.4	8.79	36.7	25.5	18.1	0K
NaVid [RSS'24]	5.47	49.1	37.4	35.9	8.41	23.8	21.2	–	953K
Uni-NaVid [RSS'25]	5.58	53.3	47.0	42.7	6.24	48.7	40.9	–	3577K
NaVILA* [RSS'25]	5.37	57.6	49.7	45.5	–	–	–	–	12574K
NaVILA [RSS'25]	5.22	62.5	54.0	49.0	6.77	49.3	44.0	58.8	13132K
StreamVLN [ICRA'26]	4.98	64.2	56.9	51.9	6.22	52.9	46.0	61.9	$\sim$26330K
GA-VLN [CVPR'26]	4.80	67.6	61.0	55.2	5.88	55.4	45.2	–	$>$ 9207K
JanusVLN* [ICLR'26]	5.17	58.0	52.8	49.2	6.46	51.4	44.3	59.1	0K
JanusVLN [ICLR'26]	4.78	65.2	60.5	56.8	6.06	56.2	47.5	62.1	10692K
FutureNav-8B† [arXiv'26]	5.15	61.6	55.5	51.4	5.93	53.3	45.4	59.8	0K
Goal2Pixel [arXiv'26]	4.80	58.3	53.9	52.7	6.91	48.1	44.7	63.0	0K
Traj-VLN [arXiv'26]	4.74	66.2	58.4	52.2	6.02	52.6	43.4	62.8	$\sim$1140K
Canvas-VLN (Ours)	5.45	61.2	53.8	48.0	**5.16**	**60.6**	**49.8**	**67.3**	0K

Note: "–" denotes an unreported result. In the Additional Data column, it indicates that the amount is unavailable or cannot be reliably converted to the unified action/QA-level count. $\sim$ denotes an approximate value, and $>$ denotes a lower bound. * denotes variants trained without additional navigation data; other external data may still be used. † denotes the FutureNav-8B variant trained only on the standard R2R-CE/RxR-CE expert trajectories.

On R2R-CE, removing transition text leaves performance essentially unchanged, this suggests that explicit cross-anchor relations are less important on shorter routes, where fewer memory segments and coordinate-frame changes are encountered. In contrast, removing transition text on RxR-CE decreases OS, SR, SPL, and nDTW by 3.05, 4.22, 3.17, and 2.40 points, respectively. Moreover, the resulting variant performs worse than Raw Anchor on RxR-CE, despite retaining the executed- trajectory overlays. This result suggests that the model may incorrectly relate trajectories expressed in different anchor-centric coordinate frames without an explicit description of the intervening rotation.

Removing the horizontal RGB view produces a larger degradation, especially on the longer-horizon RxR-CE benchmark. Specifically, SR and SPL decrease by 1.57 and 1.60 points on R2R-CE, but by 3.90 and 3.29 points on RxR-CE. The corresponding nDTW reductions of 2.30 and 2.05 points show that horizontal semantic context also improves path alignment. These results suggest that the trajectory canvas provides the geometric action space, while the horizontal view supplies complementary scene and landmark information that becomes increasingly important over long routes.

Table 2: Component ablations on R2R-CE Val-Unseen and RxR-CE Val-Unseen under the same evaluation protocol. The w/o Transition variant retains executed-trajectory overlays but removes cross-anchor transition text. Raw Anchor uses the same anchor selection, commit timing, and memory budget as GETM, but removes both trajectory overlays and transition text.

Dataset	Variant	NE↓	OS↑	SR↑	SPL↑	nDTW↑
R2R-CE	Full Canvas-VLN	5.45	61.23	53.83	**48.03**	**62.16**
	w/o Transition	**5.32**	**61.34**	**54.00**	47.99	62.12
	Raw Anchor	5.59	60.36	52.09	45.99	61.18
	w/o Horizontal RGB	5.80	58.51	52.26	46.43	59.86
RxR-CE	Full Canvas-VLN	**5.16**	**69.77**	**60.56**	**49.77**	**67.32**
	w/o Transition	5.46	66.72	56.34	46.60	64.92
	Raw Anchor	5.22	68.98	58.95	48.28	66.01
	w/o Horizontal RGB	5.47	67.27	56.66	46.48	65.27

Table 3: GETM gains over Raw Anchor on the reachable RxR-CE Val-Unseen episodes, grouped into equal-sized quartiles by reference-path length: Q1 ($\leq 8.22\,\mathrm{m}$), Q2 (8.22–$13.75\,\mathrm{m}$), Q3 (13.75–$20.04\,\mathrm{m}$), and Q4 ($> 20.04\,\mathrm{m}$). Values denote GETM minus Raw Anchor in percentage points.

Metric	Q1	Q2	Q3	Q4
ΔSR	$+2.24$	-1.23	$+4.26$	$+1.34$
ΔSPL	$+2.03$	-0.47	$+3.38$	$+1.21$

Effect across navigation horizons. Table 3 stratifies the paired RxR-CE episodes by reference-path length. GETM achieves its largest gains in Q3 (+4.26 pp SR and +3.38 pp SPL) and remains beneficial in Q4 (+1.34 pp and +1.21 pp). Collectively, Q3–Q4 outperform Raw Memory by 2.80 pp in SR and 2.29 pp in SPL, indicating that GETM's advantage is concentrated on longer paths and is consistent with the value of explicit execution history over extended navigation horizons. Appendix D reports absolute performance and uncertainty.

5 Conclusion

In this work, we introduce Canvas-VLN, an execution-aware framework for continuous vision-and-language navigation. At its core, Gated Execution-Trajectory Memory (GETM) transforms the agent's actual motion into bounded visual history, allowing past execution and future planning to share a common pixel-trajectory representation. By committing a local trajectory memory when its anchor can no longer represent the executed motion compactly and connecting successive local frames through explicit transitions, GETM preserves navigation progress without relying on future information or storing unexecuted plans. Canvas-VLN further combines complementary semantic and trajectory views with a unified autoregressive action interface, enabling closed-loop prediction, execution, and memory updating. Experiments on R2R-CE and RxR-CE show that Canvas-VLN remains competitive on shorter routes and achieves state-of-the-art performance on long-horizon navigation among methods trained without additional data. Overall, Canvas-VLN demonstrates that execution-grounded trajectory memory provides a simple and effective direction for long-horizon embodied navigation.

AI use statement

In this work, we used generative AI tools to assist with implementing parts of the software pipeline within an author-defined methodology, data pipeline, and code framework. The authors designed the overall method and system architecture, the training-data collection procedure, the interface between collected data and model inputs, and the initial software scaffold, including core inputs/outputs,

prompts, and serialization formats. The authors also manually collected and assessed the training data. Generative AI was subsequently used to complete parts of the implementation, improve code structure and robustness, and debug the end-to-end pipeline.

We did not use generative AI to identify or formulate the research gap, propose or refine research hypotheses, design the methodology, data-collection procedure, or main and ablation experiments, assess the collected training data, formulate mathematical claims or proofs, or interpret experimental results; the remaining required-disclosure tasks are not applicable.

Additionally, we used generative AI for selected visual elements in scientific figures, polishing author-written text, searching for and summarizing related literature, general brainstorming that did not involve hypothesis formulation, and assisting with running pre-specified experiments and monitoring their progress.

We reviewed all AI-assisted work. AI-assisted text, code, literature summaries, and visual elements were manually checked by the authors; code was tested before incorporation, and literature-related outputs were verified against the original publications. AI assistance in experiments was limited to executing pre-defined configurations and monitoring run status. The overall figure design and initial manuscript draft were produced by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Ethics Statement

Canvas-VLN may support beneficial applications such as assistive navigation, domestic service robotics, and search and rescue. However, autonomous navigation systems are inherently dual-use and could enable unauthorized surveillance, unsafe operation, or military applications. Moreover, biases and limited environmental diversity in navigation datasets may reduce reliability when the system is deployed in unfamiliar settings. The current system does not provide the safety guarantees required for deployment in safety-critical or human-populated environments without further validation. We intend this work to advance embodied AI for socially beneficial purposes and advocate responsible deployment with appropriate privacy protection, human oversight, fail-safe mechanisms, and compliance with applicable regulations.

Reproducibility statement

To ensure the reproducibility of our study, we provide nonymized evaluation code in Appendix A and detailed implementation and experimental settings for Canvas-VLN in Appendix B. To promote transparency and facilitate further research, we will release our source code, model configurations, and checkpoints under the applicable licenses. These resources will enable other researchers to reproduce our results and build upon our work.

References

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

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

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang,

Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Muyi Bao, Yuxin Cai, Hang Xu, Zongtai Li, Jinxi He, Jingfan Tang, Chen Lv, Ji Zhang, Yaqi Xie, and Wenshan Wang. Goal2Pixel: Grounding goals to pixels for vision-language navigation. *arXiv preprint arXiv:2606.01621*, 2026. doi: 10.48550/arXiv.2606.01621.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. In *2017 International Conference on 3D Vision (3DV)*, pp. 667–676, 2017. doi: 10.1109/3DV.2017.00081.

Kehan Chen, Dong An, Yan Huang, Rongtao Xu, Yifei Su, Yonggen Ling, Ian Reid, and Liang Wang. Constraint-aware zero-shot vision-language navigation in continuous environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(11):10441–10456, 2025. doi: 10.1109/TPAMI.2025.3594204.

Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. History aware multimodal transformer for vision-and-language navigation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 5834–5847. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/2e5c2cb8d13e8fba78d95211440ba326-Paper.pdf.

An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, Jan Kautz, Erdem Biyik, Hongxu Yin, Sifei Liu, and Xiaolong Wang. NaVILA: Legged Robot Vision-Language-Action Model for Navigation. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.018.

Changfei Fu, Guangcheng Chen, Aoxiang Gu, Haoxiang Liang, Wenjun Xu, and Hong Zhang. TrajVLN: Learning pixel-space interaction via autoregressive trajectory generation. *arXiv preprint arXiv:2607.10744*, 2026. doi: 10.48550/arXiv.2607.10744. URL https://arxiv.org/abs/2607.10744.

Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. VLN BERT: A recurrent Vision-and-Language BERT for navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1643–1653, June 2021.

Yicong Hong, Zun Wang, Qi Wu, and Stephen Gould. Bridging the gap between learning in discrete and continuous environments for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15439–15449, June 2022.

Jingzhi Huang, Junkai Huang, Wenxuan Song, Haoyang Yang, Hailong Huang, Haoang Li, and Yi Wang. SEDualVLN: A spatially-enhanced dual-system for vision-language navigation. arXiv preprint arXiv:2605.17249, 2026a. URL https://arxiv.org/abs/2605.17249.

Xiangyun Huang, Xiangchen Wang, Runfeng Lin, Yihao Xu, Kangyu Huang, Hengchen Jiang, Xiwang Dong, and Jiarong Lin. From routes to steps: Separating semantic progress from local execution in vision-and-language navigation. arXiv preprint arXiv:2608.03143, 2026b. URL https://arxiv.org/abs/2608.03143.

Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. General evaluation for instruction conditioned navigation using dynamic time warping. In *NeurIPS 2019 Workshop on Visually Grounded Interaction and Language*, 2019. URL https://arxiv.org/abs/1907.05446.

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 (ECCV)*, pp. 104–120. Springer, 2020.

Jacob Krantz, Aaron Gokaslan, Dhruv Batra, Stefan Lee, and Oleksandr Maksymets. Waypoint models for instruction-guided navigation in continuous environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15162–15171, October 2021.

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

Haoran Liu, Weikang Wan, Xiqian Yu, Minghan Li, Jiazhao Zhang, Bo Zhao, Zhibo Chen, Zhongyuan Wang, Zhizheng Zhang, and He Wang. NaVid-4D: Unleashing spatial intelligence in egocentric RGB-D videos for vision-and-language navigation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10607–10615, 2025. doi: 10.1109/ICRA55743.2025.11128467.

Yuxing Long, Wenzhe Cai, Hongcheng Wang, Guanqi Zhan, and Hao Dong. InstructNav: Zero-shot system for generic instruction navigation in unexplored environment. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2049–2060. PMLR, 2025.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/ross11a.html.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9339–9347, 2019. doi: 10.1109/ICCV.2019.00943.

Haoxiang Shi, Xiang Deng, Haoyu Zhang, Qiaohui Chu, Yaowei Wang, and Liqiang Nie. Beyond Waypoints: A trajectory-centric waypointing paradigm for Vision-Language Navigation. arXiv preprint arXiv:2606.07244, 2026. URL https://arxiv.org/abs/2606.07244.

Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2610–2621, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1268.

Zihan Wang and Gim Hee Lee. g3D-LF: Generalizable 3D-language feature fields for embodied tasks. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 14191–14202, 2025.

Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, and Shuqiang Jiang. GridMM: Grid memory map for vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15625–15636, 2023a.

Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, and Shuqiang Jiang. Sim-to-real transfer via 3d feature fields for vision-and-language navigation. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2982–2995. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/wang25i.html.

Zun Wang, Jialu Li, Yicong Hong, Yi Wang, Qi Wu, Mohit Bansal, Stephen Gould, Hao Tan, and Yu Qiao. Scaling data generation in vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12009–12020, October 2023b.

Meng Wei, Chenyang Wan, Jiaqi Peng, Xiqian Yu, Yuqiang Yang, Delin Feng, Wenzhe Cai, Chenming Zhu, Tai Wang, Jiangmiao Pang, and Xihui Liu. Ground Slow, Move Fast: A dual-system foundation model for generalizable vision-language navigation. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=GK4rznYwhn.

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 *2026 IEEE International Conference on Robotics and Automation (ICRA)*, 2026b. URL https://arxiv.org/abs/2507.05240.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 40053–40063, June 2026.

Xuan Yao, Junyu Gao, and Changsheng Xu. NavMorph: A self-evolving world model for vision-and-language navigation in continuous environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5536–5546, October 2025. doi: 10.1109/ICCV51701.2025.00525.

Xuan Yao, Yuze Zhu, Junyu Gao, Zongmeng Wang, and Changsheng Xu. SC^2-WM: A self-correcting world model with closed-loop feedback for vision-and-language navigation in continuous environments. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026.

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 *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RnuB0Nlbd5.

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.

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 Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13032–13056, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.638. URL https://aclanthology.org/2025.acl-long.638/.

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.

A Anonymous Evaluation Code

The anonymized evaluation code and configurations used to reproduce our validation results are available at https://anonymous.4open.science/r/Canvas-VLN-C3A6.

B IMPLEMENTATION DETAILS.

We adopt Qwen3-VL-8B-Instruct (Bai et al., 2025) as the backbone. The model is trained for one epoch with the vision encoder frozen, while the vision–language projection modules and the language model are fine-tuned. Table 4 summarizes the shared training and inference settings. We optimize the standard autoregressive cross-entropy loss over the assistant action responses.

We present the prompt format used by Canvas-VLN. At every policy call, the prompt is reconstructed using the updated GETM, recent turns, and current observations; previous assistant outputs are not retained as conversation history.

System: *You are a visual navigation policy. Follow the instruction using past executed trajectory memory, recent turns if provided, and the current horizontal RGB and trajectory canvas. Predict the next action.*

User: *Navigation instruction:* `<instruction>`

Past start observation: `<GETM_1>`

Past yaw transition after segment 1: turned left about 90 degrees.

Past executed memory segment 2: `<GETM_2>`

Recent executed turns since last translation: R15, L15, R15 (net R15).

Current horizontal RGB view (semantic context): `<horizontal_RGB>`

Current trajectory canvas (coordinate frame): `<trajectory_canvas>`

Return exactly one of: `[(u1,v1),(u2,v2),...]` *for a forward trajectory on the current trajectory canvas;* `TURN_LEFT` k *or* `TURN_RIGHT` k *for* k *consecutive* 15° *rotations; or* `STOP`. *Never predict coordinates on the horizontal RGB view.*

Assistant: `[(384,449),(370,335),(373,267),(375,223)]`

Next call: `<instruction>`, `<updated_GETM>`, `<new_horizontal_RGB>`, `<new_canvas>`.

Assistant: `STOP`

Table 4: Shared model, training, and inference settings for the evaluated configuration.

Setting	Value
Backbone	Qwen3-VL-8B-Instruct
Training epochs / seed	1 / 42
Global batch size	128
Learning rate / schedule	2×10^{-5} / cosine
Warmup ratio / weight decay	0.03 / 0
Configured tokenizer context length	8,192 tokens
Training precision / sharding	BF16 / ZeRO-3
Current RGB views	0° and 35° downward pitch
RGB resolution / horizontal field of view	640×480 / 90°
Current trajectory canvas	768×512
Historical canvases (including populated active)	At most 8
Executed positions per historical canvas	At most 8
Predicted points per future trajectory	At most 8
Executed-position sampling interval	0.5 m
Same-direction turn threshold	75°
Visibility margin / depth tolerance	8 pixels / 0.35 m
Short-horizon turn records	At most 6, reset by translation
Greedy decoding maximum new tokens	128
Action limit / success threshold	500 / strictly below 3 m after STOP

Table 5: Matched execution-horizon analysis with fixed model weights, memory construction, controller, and episode sets. Calls are actual policy invocations per episode. RxR columns both use the same 3,570-episode current-navmesh reachable subset.

B	R2R-CE ($N = 1{,}839$)			RxR-CE ($N = 3{,}570$)		
	SR↑	**SPL↑**	**Calls**	**SR↑**	**SPL↑**	**Calls**
2	53.83	48.03	37.05	62.24	51.15	59.93
4	50.68	43.96	20.63	56.61	45.75	32.93
6	50.73	43.79	15.50	56.33	45.31	23.66

The historical canvases retain the RGB image plane of their anchor observations. The current prediction canvas embeds its 640×480 RGB image with a 64-pixel horizontal offset inside a 768×512 canvas; predicted coordinates refer to that current canvas.

C Forward execution budget

The forward budget B controls how many forward actions can be executed from a trajectory prediction before the next policy call using updated observations. We compare $B \in \{2, 4, 6\}$, corresponding to nominal forward-execution budgets of 0.5, 1.0, and 1.5 m per prediction. Table 5 pairs SR and SPL with mean policy calls on the same population in each column group.

For RxR, following Route2Step (Huang et al., 2026b), we additionally report a fixed current-navmesh reachable subset of 3,570 episodes. The other 99 episodes correspond to 33 routes with three English instructions each; their start and goal lie in disconnected components of the installed navigation mesh. The same exclusion set is applied to all execution budgets after evaluating the full split.

The two-forward-action setting has the highest success rate on both datasets among the three evaluated budgets. Moving from $B = 4$ to $B = 2$ increases SR by 3.15 percentage points on R2R and 5.63 points on the RxR reachable subset, with more frequent policy calls. Increasing the budget from four to six further reduces call counts, while SR remains close to the four-action setting.

D Statistical analysis

Navigation horizons. We further analyze the effect of navigation horizon on the reachable RxR-CE Val-Unseen episodes. Episodes are divided into four equal-sized groups according to reference-path length, with Q1 containing the shortest routes and Q4 the longest. We report paired GETM minus Raw-Anchor differences because both variants are evaluated on the same episode set.

GETM produces positive point estimates in three of the four groups. In Q1, it improves SR and SPL by 2.24 and 2.03 percentage points, respectively, although both confidence intervals cross zero. Q2 shows small negative differences of -1.23 points in SR and -0.47 points in SPL, again with intervals spanning zero. The clearest improvement occurs in Q3, where GETM increases SR by 4.26 points (95% CI: $[0.34, 8.08]$) and SPL by 3.38 points (95% CI: $[0.45, 6.42]$).

In Q4, the point estimates remain positive, at $+1.34$ points in SR and $+1.21$ points in SPL, but their confidence intervals, $[-2.68, 5.37]$ and $[-2.08, 4.49]$, cross zero. These results should therefore be interpreted as an uncertain positive trend rather than conclusive evidence of improvement on the longest routes. Nevertheless, when Q3 and Q4 are aggregated, GETM improves SR and SPL by 2.80 and 2.29 points, respectively, suggesting that its overall advantage is stronger in the longer half of the evaluation set.

Our failure analysis suggests two factors behind the gain reduction from Q3 to Q4. First, GETM retains at most eight anchor images under a fixed FIFO memory budget. As an episode becomes increasingly long, earlier trajectory segments are evicted, and the retained memory covers a smaller fraction of the complete navigation history. The benefit of explicit execution history therefore begins

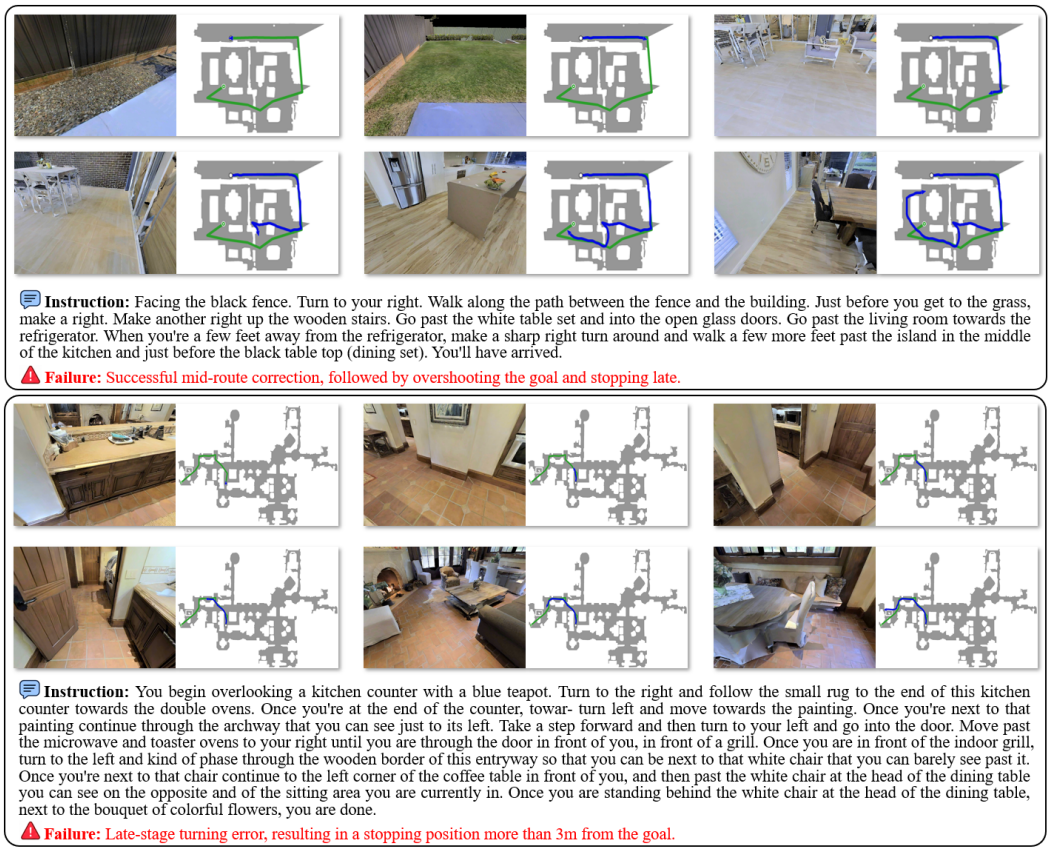

Figure 4: Visualization and presentation of the types of failure cases.

to saturate once the route length exceeds the effective memory horizon. Second, GETM records the trajectory actually executed by the agent rather than an oracle trajectory. Consequently, navigation, localization, or projection errors can also be written into memory. Longer episodes provide more opportunities for such errors to accumulate, making some stored segments less informative or potentially misleading for subsequent decisions. Together, bounded memory coverage and accumulated erroneous history provide a plausible explanation for the smaller and less certain gains observed in Q4.

Failure case analysis. Our qualitative analysis on RxR-CE identifies two representative failure modes in Figure 4. In the top example, the agent makes an incorrect turn midway through the episode but recovers at the next intersection, suggesting that GETM provides useful history for self-correction. However, another turning error near the end causes the agent to pass the intended stopping region and stop late. This behavior may be amplified by the multi-step TURN-k protocol, which executes k consecutive 15° rotations from a single prediction; an error in either turn timing or k can therefore produce a substantial heading deviation before the policy is queried again. In the bottom example, the agent closely follows the expert path throughout most of the episode, but fails to execute the final turn and determine the correct stopping point. This indicates that successful long-horizon route following does not necessarily translate into reliable fine-grained control near the goal. Moreover, because Canvas-VLN is trained on expert demonstrations without DAgger-style on-policy data, it receives limited supervision for recovering from its own turning errors and handling perturbed near-goal states. These cases motivate more frequent observation updates during multi-step turns and targeted on-policy training for recovery and stopping decisions.

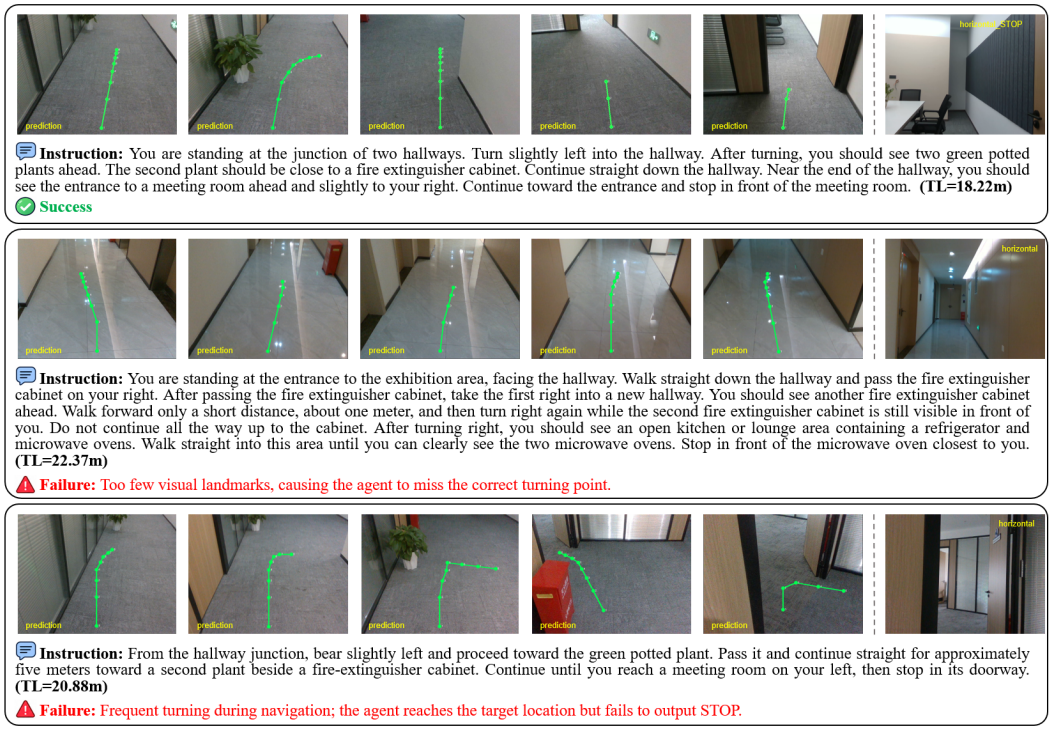

Figure 5: More results of Canvas-VLN on real-world.

E More real-world results

Figure 5 presents additional real-world results. On long routes with sparse visual landmarks, Canvas-VLN occasionally misses the instructed turning point and continues forward. Even when the turn is executed, variations in the resulting camera heading may prevent the agent from observing a kitchen or pantry that becomes visible only after two closely spaced 90° turns. We attribute these failures to the high visual similarity among GETM segments in landmark-sparse corridors, which makes the relevant location difficult to distinguish, together with sensitivity to post-turn viewpoints that can obscure narrow doors and entrances. In other cases, the agent reaches the target but fails to output `STOP`, likely because the final viewpoint does not reveal sufficient semantic evidence of the specified room.

F Pixel-trajectory execution in Habitat

Canvas-VLN predicts an ordered sequence of waypoints on the trajectory canvas, whereas Habitat accepts only discrete navigation actions. We therefore use a simulator-specific controller to translate pixel trajectories into executable actions. Each predicted waypoint is first mapped to the RGB image plane and back-projected into 3D using the current depth observation, camera intrinsics, and camera pose. If the corresponding depth value is invalid, the controller searches progressively larger local windows (3×3, 5×5, and 7×7) and selects a valid pixel whose depth is closest to the local median; waypoints without valid depth are discarded.

The resulting 3D waypoints are validated sequentially against Habitat's navigation mesh. A non-navigable waypoint is first snapped to the mesh and is rejected if snapping fails, the displacement exceeds $0.5\,\mathrm{m}$, or its vertical component exceeds $0.35\,\mathrm{m}$. Let d_E and d_G denote the Euclidean and geodesic distances, respectively, from the current validation anchor to the candidate waypoint. We additionally reject the waypoint if it is unreachable or if

$$d_G > 2d_E + 0.5\,\mathrm{m},$$

which filters projections that lie across walls or require excessive detours. Each accepted waypoint becomes the anchor for validating the next one. The validated sequence is executed using Habitat's navmesh-aware `ShortestPathFollower` with a $0.25\,\mathrm{m}$ goal radius. Execution is terminated after the prescribed forward-action budget, after which the policy is queried again with updated observations. If a prediction cannot be parsed, projected, or safely validated, the policy receives brief execution feedback and is allowed one immediate re-prediction.

The navigation mesh is accessed exclusively by the downstream controller and is never exposed to the VLM. In particular, the policy receives no goal coordinates, reference trajectory, distance-to-goal signal, or navigation-mesh representation. The controller and all of its parameters are held fixed across GETM, Raw Anchor Memory, and the other ablation variants.

Navmesh validation and `ShortestPathFollower` are specific to the Habitat evaluation protocol and are not components of the learned policy or GETM. In physical deployment, they are replaced by back-projection from onboard RGB-D observations and the robot's local motion-planning and control stack. Developing a fully onboard obstacle-aware controller is outside the scope of this work, which focuses on long-horizon trajectory memory and multimodal navigation policy learning.