ROUTEGUIDE: Fast VLN Adaptation from a Successful Video

Anonymous ACL submission

Abstract

Vision-and-Language Navigation (VLN) agents are typically evaluated based on whether they can reach a goal once, yet practical deployment also requires reliable re-execution after a successful trial or human correction. We introduce ROUTEGUIDE, a demonstration-guided VLN strategy that improves post-success repeatability without retraining or fine-tuning the underlying navigation policy. Given a single successful egocentric video, ROUTEGUIDE converts the trajectory into reusable route memory, represented as discrete route codes learned from depth-aware visual transitions. At inference time, the base VLN policy remains frozen: it proposes action and goal candidates, while a lightweight Hierarchical Route Prior retrieves relevant route codes and reranks these candidates toward the route structure of the successful execution. In a focused ten-episode VLN-CE val-unseen evaluation, ROUTEGUIDE improves SR from 32.0% to 40.0%, OS from 42.0% to 46.0%, SPL from 0.213 to 0.275, and NE from 8.21 to 7.69 over 50 rollouts.

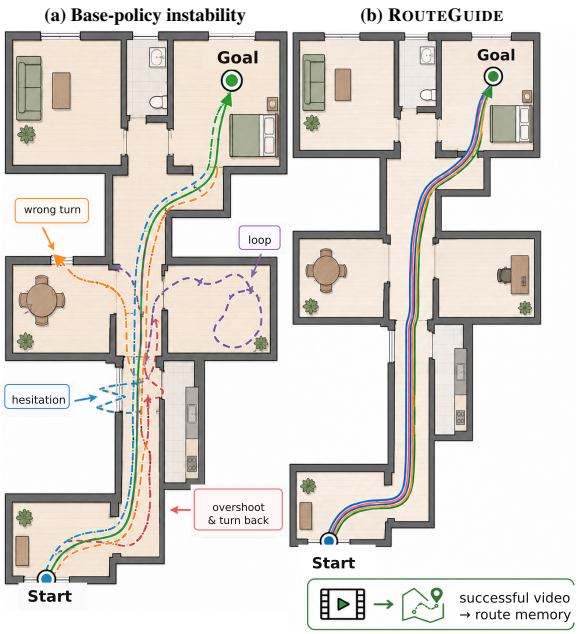

Figure 1: Problem illustration from a top-down view. (a) Repeated executions by the base VLN policy can diverge across the same instruction, including hesitation, loops, and wrong turns. (b) ROUTEGUIDE converts one successful video into route memory, which guides later executions toward the demonstrated route without updating the base policy.

1 Introduction

Vision-and-Language Navigation (VLN) requires embodied agents to follow natural-language instructions in visually grounded environments (Anderson et al., 2018; Krantz et al., 2020). Recent systems improve benchmark performance with stronger vision-language models, long-context reasoning, waypoint prediction, and low-level action generation (Wei et al., 2025a,b; Song et al., 2024). These advances are usually measured by single-execution metrics such as success rate (SR) and Success weighted by Path Length (SPL). However, a single successful rollout can be accidental: it may result from favorable sampling, a local ambiguity being resolved in the right direction, or one lucky recovery. For a user or evaluator, confidence comes only when the same instruction can be executed successfully again, not merely when it succeeds once. Figure 1 illustrates this gap, where repeated runs of the same instruction can diverge through hesitation, loops, wrong turns, or overshooting even after one success has been observed.

This repeatability requirement changes what counts as convincing evidence. A successful retry or human correction does show that a feasible route exists, but the success itself is not enough to establish that the agent has become reliable on that route. The stronger test is whether the successful execution can be replayed as usable guidance that makes later attempts succeed consistently. Existing correction mechanisms often require deployment-time parameter updates, explicit pose estimation, topo-

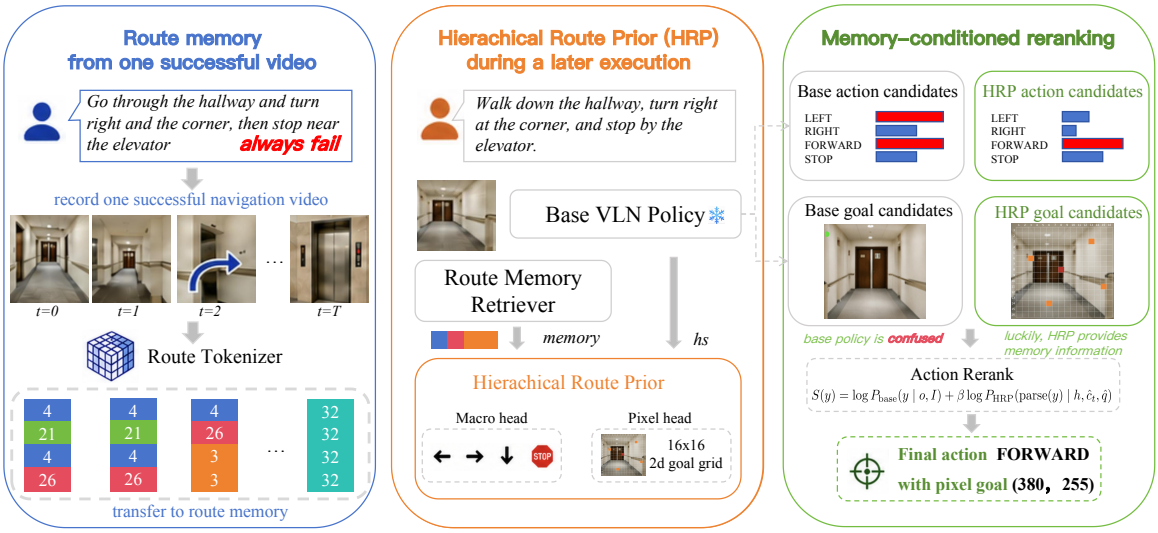

Figure 2: Overview of ROUTEGUIDE. A successful egocentric execution is compressed into discrete route codes by a depth-aware VQ route tokenizer. During later executions, the route codes are not inserted into the language prompt. Instead, the frozen base policy proposes action or goal candidates, while HRP uses the policy hidden state, local route code, and progress bin to produce route-conditioned priors. The final action is selected by reranking the base-policy candidates with the HRP prior.

logical mapping, or 3D reconstruction before a successful run can influence future behavior (Chiang et al., 2024; Yu et al., 2025), making this repeatability test difficult to perform quickly.

We propose ROUTEGUIDE, a demonstration-guided VLN strategy that converts one successful navigation video into reusable *route memory*. Route memory is a sequence of discrete route codes that records local egocentric transitions along the demonstrated route. Because first-person navigation actions are not directly visible, unlike many manipulation videos (Ye et al., 2025; Bu et al., 2025; Routray et al., 2026), we learn these codes with a depth-aware route tokenizer that emphasizes route-relevant scene and depth changes rather than superficial appearance variation.

As shown in Figure 2, ROUTEGUIDE keeps the pretrained VLN policy fixed at inference time. The base policy proposes action and goal candidates, while a lightweight Hierarchical Route Prior (HRP) retrieves the route code for the predicted progress bin and reranks the parsed candidates. This design avoids online fine-tuning or map construction, preserves the base policy's general ability, and makes route influence measurable through correct-code versus wrong-code comparisons. Our evaluation follows this causal chain with logged candidate reranking under correct-code and wrong-code controls, followed by Habitat navigation on unstable val-unseen episodes.

Our contributions are:

- We formulate **single-video VLN adaptation** as a repeated-execution problem, where the goal is not to celebrate one successful rollout but to test whether that success can be reproduced on later attempts.
- We introduce **route memory** and **ROUTEGUIDE**, which encode a successful egocentric video into discrete depth-aware route codes and use HRP to rerank frozen-policy action candidates.
- We design a staged evaluation with logged reranking and Habitat navigation, using correct-code versus wrong-code controls to verify that route codes change decisions in the intended direction.

2 Related Work

Our work connects vision-and-language navigation, memory-augmented navigation, demonstration-guided navigation, and latent action learning.

Vision-and-Language Navigation. Vision-and-language navigation (VLN) requires an embodied

agent to follow natural-language instructions in visually grounded environments. Early studies formulate VLN over discrete navigation graphs (Anderson et al., 2018), while later work extends it to continuous environments that better reflect real-world deployment (Krantz et al., 2020). Recent methods further improve VLN agents with stronger vision-language models, long-context reasoning, waypoint prediction, and low-level trajectory generation (Wei et al., 2025a,b; Song et al., 2024). These systems are still mainly evaluated by single-run metrics such as success rate (SR) and Success weighted by Path Length (SPL). Such metrics measure whether an agent succeeds once, but they do not show whether the same instruction can be executed reliably after a successful trial. Our work targets this deployment-oriented gap by studying repeated execution after a route-specific demonstration is available.

Memory-Augmented Navigation. Memory has been widely used to support long-horizon VLN. Sequential methods encode historical observations with recurrent models or full-history attention (Chen et al., 2021), graph-based methods build topological maps over visited viewpoints for global planning (Chen et al., 2022; Hong et al., 2025), and retrieval-augmented methods recall relevant past observations or experiences to reduce long-context reasoning cost (Xu et al., 2025; Li et al., 2026; Zheng et al., 2024; Krantz et al., 2023). These approaches mainly store the agent's own online history, such as where it has been and what it has observed. Deployment correction needs a different type of context: how a successful execution of the same instruction moved through the environment. RouteGuide therefore treats a demonstration video as external behavioral memory and uses it to condition action selection without constructing an explicit spatial map.

Demonstration-Guided Navigation and One-Shot Adaptation. Demonstrations have long been used for policy learning through behavior cloning or imitation learning (Inglin et al., 2026; Zhang et al., 2025). Although effective, these methods typically require training or fine-tuning, making them less suitable for immediate correction of a deployed robot. Recent one-shot VLA/VLN studies aim to adapt from a single demonstration, such as ViVLA for manipulation (Chen et al., 2025), low-shot policy learning, and video-to-action transfer (Song et al., 2025; Zhou and Jia, 2025; Wang et al., 2024). In navigation, UniNaVid (Zhang et al., 2024) and RoomTour3D (Han et al., 2025) explore video-based embodied navigation and generalization. Other demonstration-guided systems, including MINT (Chiang et al., 2024) and RANGER (Yu et al., 2025), use videos as inference-time guidance but often depend on explicit spatial reconstruction, pose estimation, or reliable 3D structure. These dependencies can be fragile in dynamic, textureless, or incompletely observed environments. Our approach avoids explicit reconstruction by converting a successful navigation video into route-conditioned priors over the base policy's parsed action candidates.

Latent Action Learning. Latent action learning infers action-like representations from videos by modeling visual transitions between frames. Representative methods include LAPA, which learns discrete latent actions with a VQ-VAE-based inverse dynamics model (Ye et al., 2025); UniVLA, which learns instruction-conditioned latent actions for cross-embodiment transfer (Bu et al., 2025); and ViPRA, which combines latent action learning with video prediction and motion consistency constraints (Routray et al., 2026). First-person navigation differs from manipulation because the motion cue is expressed through global scene change and depth variation rather than visible hand or object motion. We adapt latent action learning to this setting by learning depth-aware route codes and using them as lightweight inference-time priors for single-video VLN adaptation.

3 Method

RouteGuide uses one successful execution as route-specific memory for later executions of the same instruction. The successful egocentric video is compressed into a sequence of discrete route codes, where each route code is a vector-quantized codebook entry learned from depth-aware visual transitions. These codes are used only by a lightweight Hierarchical Route Prior (HRP), as summarized in Fig. 2. The pretrained InternVLA-N1 navigation policy (Wei et al., 2025a), including its vision tower, language head, and low-level controller, remains frozen. Route codes are never inserted into the language prompt, input IDs, or language-model labels.

At each navigation step, the frozen policy provides candidate log-probabilities and hidden states before candidate generation. HRP conditions on

the hidden state, the retrieved local route code, and a progress bin to score parsed actions and goals. The final decision combines the base score with the HRP prior:

$$S(y) = \log P_{\text{base}}(y \mid \mathbf{o}, \mathcal{I}) + \beta \log P_{\text{HRP}}(\text{parse}(y) \mid \mathbf{h}, \mathbf{c}, q), \quad (1)$$

where $\mathcal{I}$ is the instruction, $\mathbf{o}$ is the current observation, $\mathbf{h}$ is the frozen policy hidden state, $\mathbf{c}$ is the retrieved route code, and q is the progress bin. Candidate selection then uses $S(y)$ to rerank the base-policy outputs. This design makes the demonstration influence explicit while preserving the base model's general navigation ability.

3.1 Route Memory from a Successful Video

Route memory should preserve how the route was executed while discarding appearance changes that are irrelevant to navigation. Unlike tabletop manipulation, first-person navigation induces large viewpoint and scene changes, so image differences alone do not reliably reveal the executed action. We therefore learn a VQ-VAE-style route tokenizer that maps a short egocentric transition $(\mathbf{x}_t, \mathbf{x}_{t+\Delta})$ to $M{=}4$ route-code slots, each selected from a $K{=}32$ codebook. The tokenizer uses frozen Depth Anything V2 / DINOv2 features (Yang et al., 2024; Oquab et al., 2024), multi-layer patch projections, and a spatio-temporal Transformer with learnable action slots. The slot outputs $\mathbf{u}_m$ are discretized by nearest-codeword assignment, so a route code is the index of a learned codebook entry rather than a hand-written action label.

During training, we use NSVQ-style quantization. For a codebook $\{\mathbf{e}_k\}_{k=1}^{K}$ and nearest codeword $\text{NN}(\mathbf{u}_m)$, the quantized latent is

$$\hat{\mathbf{z}}_m = \mathbf{u}_m + \frac{\|\mathbf{u}_m - \text{NN}(\mathbf{u}_m)\|}{\|\boldsymbol{\epsilon}\|}\boldsymbol{\epsilon}, \quad (2)$$

where $\boldsymbol{\epsilon}$ is an isotropic noise vector. To encourage codebook usage, we compute a differentiable soft assignment

$$p_{m,k} = \frac{\exp(-\|\mathbf{u}_m - \mathbf{e}_k\|^2/\tau)}{\sum_{k'} \exp(-\|\mathbf{u}_m - \mathbf{e}_{k'}\|^2/\tau)}. \quad (3)$$

The tokenizer is trained as a transition autoencoder. The quantized route codes reconstruct future multi-level visual features while a depth-consistency term preserves geometry:

Code $(30, 12, 29, 25)$ **Code** $(6, 6, 15, 5)$

Figure 3: Examples of VQ route-code discretization. The left panel shows transitions assigned to code $(30, 12, 29, 25)$, corresponding to an open-area turning pattern. The right panel shows transitions assigned to code $(6, 6, 15, 5)$, corresponding to motion along a corridor side or narrow passage. The visualization illustrates that route codes group visually different frames by shared navigation-transition semantics.

$$\mathcal{L}_{\text{rec}} = \sum_{\ell \in \{4,11,17,23\}} w_\ell \left\| \hat{\mathbf{F}}^{\ell}_{t+\Delta} - \mathbf{F}^{\ell}_{t+\Delta} \right\|_1, \quad (4)$$

$$\mathcal{L}_{\text{depth}} = \left\| \text{DPT}(\hat{\mathbf{F}}_{t+\Delta}) - \text{sg}\left[\text{DPT}(\mathbf{F}_{t+\Delta})\right] \right\|_1, \quad (5)$$

$$\mathcal{L}_{\text{ent}} = \log K - H\left(\tfrac{1}{B}\textstyle\sum_b \mathbf{p}^{(b)}\right). \quad (6)$$

The final tokenizer loss is

$$\mathcal{L}_{\text{tok}} = \mathcal{L}_{\text{rec}} + \lambda_d \mathcal{L}_{\text{depth}} + \lambda_e \mathcal{L}_{\text{ent}}. \quad (7)$$

After training, the tokenizer and codebook are frozen. The learned codebook groups visually different examples by transition semantics, as illustrated in Fig. 3. Applying the tokenizer once to the successful video produces route memory $\mathbf{C}^{\text{demo}} = (\mathbf{c}_1^{\text{demo}}, \ldots, \mathbf{c}_T^{\text{demo}})$, whose temporal structure is shown in Fig. 4.

STOP is handled as a terminal route state rather than as a visual no-op transition. We reserve code index 32 for STOP and use $\mathbf{c}_{\text{STOP}} = [32, 32, 32, 32]$, giving HRP a 33-entry vocabulary: movement codes $0, \ldots, 31$ plus the terminal code. For events whose action is down, route memory is computed from the front-view motion transition before and after movement, not from the camera pitch change, so the code represents where the agent moves rather than where the camera looks.

3.2 Hierarchical Route Prior

HRP is trained on event-level decisions from the frozen navigation policy. Each event has an action

Figure 4: Example route-code sequence produced from a successful egocentric navigation video. The figure shows route-code sequences from five randomly selected episodes over navigation steps, where each visual transition is mapped to a compact tuple of VQ codebook entries. This temporal memory can be retrieved by HRP during later executions of the same instruction.

$m_t \in \{\text{left}, \text{right}, \text{down}, \text{STOP}\}$, where down denotes the forward movement mode that subsequently predicts a 2D pixel goal. If $m_t = \text{down}$, the policy predicts this goal and maps it to a 16×16 grid id $g_t \in \{0, \ldots, 255\}$. Each event provides a positive route code $\mathbf{c}_t^+$, a negative route code $\mathbf{c}_t^-$ for code-swap contrast, and a progress bin $q_t = \text{round}(127t/(T-1))$. For STOP events, $\mathbf{c}_t^+ = \mathbf{c}_{\text{STOP}}$, $q_t = 127$, and the goal target is omitted.

The frozen policy only supplies hidden states and candidate log-probabilities. We read the stop-gradient hidden state immediately before the front-view action answer as $\mathbf{h}_t^f$, and the corresponding goal-answer state as $\mathbf{h}_t^d$ when an action labeled down requires a 2D pixel goal. HRP then maps a hidden state, a route code, and a progress bin to route-conditioned priors.

HRP is implemented as a small conditioner plus two prediction heads. The Route Conditioner first embeds the route-code tuple. For a code $\mathbf{c}_t = (c_{t,1}, \ldots, c_{t,M})$ with $M = 4$, each slot adds a codebook-entry embedding and a learned slot embedding; the four slot embeddings are then averaged:

$$\mathbf{z}_t = \frac{1}{M} \sum_{m=1}^{M} \left(\mathbf{E}_{\text{code}}(c_{t,m}) + \mathbf{E}_{\text{slot}}(m)\right). \quad (8)$$

The conditioner then projects the policy hidden state and route embedding into the same route space and combines them with a multiplicative interaction and a progress embedding:

$$\mathbf{h}_t^r = \mathbf{W}_h \mathbf{h}_t, \qquad \mathbf{z}_t^r = \mathbf{W}_z \mathbf{z}_t,$$
$$\mathbf{r}_t = \text{LN}\big(\mathbf{h}_t^r + \mathbf{z}_t^r + \mathbf{W}_i[\mathbf{h}_t^r \odot \mathbf{z}_t^r] + \mathbf{E}_{\text{prog}}(q_t)\big). \quad (9)$$

This design keeps HRP lightweight while forcing the prior to depend jointly on the frozen policy

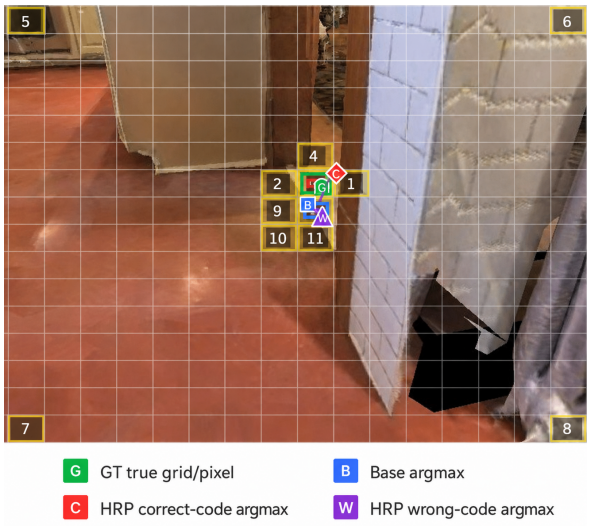

Figure 5: Goal correction mechanism when the base model selects a wrong target point. The ground-truth goal is discretized on a 16×16 grid (green). The base argmax can fall on an incorrect nearby cell (blue), while HRP with the correct route code can rerank the candidate set toward the correct spatial target (red). Using a wrong route code provides a negative condition (purple), isolating whether route memory, rather than an observation-only bias, drives the correction.

state, the retrieved route code, and progress along the demonstration.

HRP uses the same conditioned representation for action and goal prediction. For front-view decisions, a two-layer MLP maps $\mathbf{r}_t^f$ to four action logits over $\{\text{left}, \text{right}, \text{down}, \text{STOP}\}$. For actions labeled down, another two-layer MLP maps $\mathbf{r}_t^d$ to 256 logits over the 16×16 2D goal grid. As illustrated in Fig. 5, this goal prior is most useful when the base model selects an incorrect nearby target: HRP can rerank the same candidate set toward the demonstrated grid cell under the correct route code, while a wrong route code provides a controlled negative condition for the same observation and candidates.

A separate progress head predicts 128 progress-bin logits directly from $\mathbf{h}_t^f$. It is not used to score actions or goals during reranking. Instead, at inference time it selects which segment of the stored demonstration should provide the current route code, after which the action and goal priors score candidates using the retrieved code.

3.3 Code-Swap Training

Only HRP is optimized during training. A plain cross-entropy loss can allow HRP to ignore route codes and predict actions from the frozen policy state alone. We therefore train HRP with a code-swap contrast: for the same hidden state, target, and progress bin, the correct route code should assign higher probability to the target than an incorrect route code. The trainable HRP modules are the Route Conditioner, the four-way action head, the 16×16 goal-grid head, and the progress head.

The same contrast is used for the action and goal priors. Let $a \in \{\text{act}, \text{goal}\}$ denote the prior, $u_t^{\text{act}} = m_t$, $u_t^{\text{goal}} = g_t$, $\mathbf{h}_t^{\text{act}} = \mathbf{h}_t^f$, and $\mathbf{h}_t^{\text{goal}} = \mathbf{h}_t^d$. The code-swap gap is

$$\begin{aligned}\Delta_a = &\log P_{\text{HRP}}^a(u_t^a \mid \mathbf{h}_t^a, \mathbf{c}_t^+, q_t) \\ &- \log P_{\text{HRP}}^a(u_t^a \mid \mathbf{h}_t^a, \mathbf{c}_t^-, q_t).\end{aligned} \tag{10}$$

This gap is both a training signal and a diagnostic: a positive value means that HRP prefers the target under the correct route memory. The training loss combines positive-code action cross-entropy, an action margin penalty $[\delta - \Delta_{\text{act}}]_+$, positive-code goal-grid cross-entropy and goal margin penalty for down events, and progress-bin cross-entropy. STOP events participate in action and progress losses but are excluded from goal loss.

3.4 Route-Conditioned Reranking

At deployment, no online learning is performed. The frozen VQ route tokenizer first encodes the successful video into $\mathbf{C}^{\text{demo}}$. At step t, the progress head predicts $\hat{q}_t$, which is linearly mapped to a route-memory index $\hat{k}_t = \text{round}(\hat{q}_t(|\mathbf{C}^{\text{demo}}| - 1)/127)$; the retrieved code is $\hat{\mathbf{c}}_t = \mathbf{C}^{\text{demo}}_{\hat{k}_t}$. Since each entry in $\hat{\mathbf{c}}_t$ points to a learned codebook vector, the retrieved code carries the route-transition semantics learned during VQ reconstruction, rather than serving as a purely symbolic time marker.

The frozen policy scores candidate actions $\mathcal{Y}_{\text{act}} = \{\text{left}, \text{right}, \text{down}, \text{STOP}\}$ with teacher-forced mean token log-probabilities $s_{\text{base}}(y)$. HRP scores the parsed action under the retrieved route code, and the final action is chosen by the combined base and HRP score in Eq. (1) with action weight β_{act}. If the selected action is down, the base policy proposes 2D pixel-goal candidates for the forward move, and HRP reranks them analogously with $P_{\text{HRP}}^{\text{goal}}$. The final goal candidate is passed to the original parser and low-level controller, so route memory affects behavior only through the HRP prior over parsed base-policy outputs.

4 Experiments

We evaluate ROUTEGUIDE with a mechanism-first protocol: logged candidate reranking with correct-

code and wrong-code controls, followed by closed-loop Habitat navigation on unstable val-unseen episodes.

4.1 Experimental Setup

All experiments use InternVLA-N1 (Wei et al., 2025a) as the frozen navigation backbone. HRP is trained on VLN-CE training events and evaluated on val-unseen data, while the vision tower, language head, navigation backbone, and low-level controller remain unchanged. We evaluate ROUTEGUIDE in two stages. The logged reranking stage isolates the inference score in Eq. (1) on held-out val-unseen samples by comparing the base ranking with reranking under the retrieved correct route code and a wrong route code. The closed-loop Habitat stage then tests whether this reranking behavior transfers to navigation, using the same base policy, a text-hint baseline, and ROUTEGUIDE under repeated execution.

For the Habitat stage, we construct a focused unstable subset from val-unseen episodes. We first take the five longest episodes from each val-unseen scene and execute each candidate episode five times with the base policy. We keep episodes whose base success rate is between 0.2 and 0.6 inclusive, yielding ten episodes where the base policy can sometimes succeed but is not repeatable. The lower bound guarantees at least one successful run, from which we obtain the successful video used by ROUTEGUIDE and by the text-hint baseline; the upper bound filters out episodes that are already stable under repeated base execution. We then execute each selected episode five times with the base policy, the text-hint baseline, and ROUTEGUIDE. The text-hint baseline appends a GPT-5.5-generated route description from the full successful video to the navigation input. Logged reranking reports base, correct-code, and wrong-code action accuracy plus flip rates, while navigation reports SR, OS, SPL, NE, and Repeat@5, where an episode counts only if all five executions succeed.

4.2 Logged Rerank Evaluation

We rerank the same 50 held-out val-unseen logged samples with the inference rule: base log-probability plus a weighted HRP prior. For each sample, we compare the base ranking, reranking with the retrieved correct route code, and reranking with a wrong route code. $\beta_{\text{act}} = 0$ recovers the base ranking; larger values increase route-prior influence.

As shown in Table 1, the base policy obtains 86.00% action accuracy at $\beta_{\text{act}} = 0$, which also corresponds to no beneficial or harmful flips because the HRP prior is disabled. Adding the correct route code with $\beta_{\text{act}} = 0.5$ raises accuracy to 92.00%, while the wrong-code condition remains at the base 86.00%; this setting produces 8.00% beneficial flips and only 2.00% harmful flips. Increasing the weight to $\beta_{\text{act}} \in \{1.0, 2.0\}$ preserves the same 92.00% correct-code accuracy, but wrong-code accuracy decreases to 76.00% and 68.00%, respectively, indicating that an overly strong route prior can override the base evidence when the retrieved code is mismatched. We therefore use $\beta_{\text{act}} = 0.5$ for Habitat navigation because it captures the full correct-code gain without increasing sensitivity to an incorrect route code.

4.3 Habitat Navigation on Unstable Episodes

Habitat rollouts test whether logged reranking carries over to closed-loop navigation and whether natural-language hints from the same video are sufficient.

As reported in Table 2, the text-hint baseline does not improve closed-loop success despite using the same successful video as input context. It obtains the same 16/50 successful rollouts as the base policy, leaving SR at 32.0%, OS at 42.0%, and R@5 at 0/10, with only marginal efficiency changes in SPL (0.213 to 0.215) and NE (8.21 to 8.18). In contrast, ROUTEGUIDE converts the same demonstration signal into action-level reranking and increases successful rollouts from 16/50 to 20/50. This corresponds to an 8.0-point SR gain, from 32.0% to 40.0%, together with higher OS (42.0% to 46.0%), higher SPL (0.213 to 0.275), lower NE (8.21 to 7.69), and the first repeatable episode under the strict R@5 criterion (0/10 to 1/10). At the episode level, four episodes improve in SR, four remain unchanged, and two regress, indicating that the route prior improves repeatability on part of the unstable subset while still leaving failure cases when the retrieved latent code biases the rollout toward an unhelpful local action.

5 Conclusion

We presented ROUTEGUIDE, which turns one successful egocentric video into discrete route memory for repeatable VLN without updating the frozen base policy. The method makes a narrow deployment claim: route-specific evidence should im-

Table 1: Offline logged action reranking on 50 held-out val-unseen samples. $\beta_{\text{act}} = 0.5$ improves correct-code accuracy without reducing wrong-code accuracy.

β_{act}	Base Action Acc↑	HRP Correct Action Acc↑	HRP Wrong Action Acc↑	Beneficial Flip↑	Harmful Flip↓
0.0	86.00	86.00	86.00	0.00	0.00
0.5	86.00	92.00	86.00	8.00	2.00
1.0	86.00	92.00	76.00	8.00	2.00
2.0	86.00	92.00	68.00	8.00	2.00

Table 2: Habitat navigation on the focused VLN-CE val-unseen subset. Each method is run five times on each of the ten unstable episodes; Repeat@5 counts episodes with five successful executions.

Method	Rollouts		Success			Efficiency		Repeat
	Ep.	Succ.	SR↑	ΔSR	OS↑	SPL↑	NE↓	R@5↑
Base	10	16/50	32.0	–	42.0	0.213	8.21	0/10
+ text hint	10	16/50	32.0	+0.0	42.0	0.215	8.18	0/10
ROUTEGUIDE	10	20/50	**40.0**	**+8.0**	**46.0**	**0.275**	**7.69**	**1/10**

prove later executions beyond a one-off success. This is why we pair closed-loop navigation with logged reranking under correct-code and wrong-code controls; gains without this separation would not support the proposed mechanism. On the focused unstable subset, ROUTEGUIDE improves aggregate navigation metrics and produces one fully repeatable episode under Repeat@5.

6 Limitations

ROUTEGUIDE is evaluated on a focused unstable subset rather than on a full-scale VLN benchmark, so broader reliability claims require larger evaluations across more scenes, instructions, and base-policy failure modes. The current HRP module can only rescore candidates that the frozen base policy already proposes; if the correct action or goal is absent from the candidate set, route memory cannot recover it. The method also assumes that the current rollout remains close enough to the demonstrated route for progress-bin retrieval to select a useful segment of the successful video. Compared with text hints, latent route memory preserves local visual-transition cues that language summaries can omit, but its effect depends on retrieving the right segment of the successful video. Improving progress estimation and route-memory indexing accuracy is therefore a key direction for making the prior more reliable in longer or more divergent rollouts.

References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683.

Qiaojun Bu and 1 others. 2025. Univla: Learning to act anywhere with task-centric latent actions. In *Robotics: Science and Systems*. ArXiv preprint arXiv:2505.06111.

Guangyan Chen, Meiling Wang, Qi Shao, Zichen Zhou, Weixin Mao, Te Cui, Minzhao Zhu, Yinan Deng, Luojie Yang, Zhanqi Zhang, Yi Yang, Hua Chen, and Yufeng Yue. 2025. See once, then act: Vision-language-action model with task learning from one-shot video demonstrations. *arXiv preprint arXiv:2512.07582*.

Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. 2021. History aware multimodal transformer for vision-and-language navigation. In *Advances in Neural Information Processing Systems*, volume 34, pages 5834–5847.

Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. 2022. Think global, act local: Dual-scale graph transformer for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16537–16547.

Hao-Tien Lewis Chiang and 1 others. 2024. Mobility vla: Multimodal instruction navigation with long-context vlms and topological graphs. *arXiv preprint arXiv:2407.07775*.

Mingfei Han, Liang Ma, Kamila Zhumakhanova, Ekaterina Radionova, Jingyi Zhang, Xiaojun Chang, Xiaodan Liang, and Ivan Laptev. 2025. Roomtour3d: Geometry-aware video-instruction tuning for embodied navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Haodong Hong and 1 others. 2025. General scene adaptation for vision-and-language navigation. *arXiv preprint arXiv:2501.17403*.

Yves Inglin, Jonas Frey, Changan Chen, and Marco Hutter. 2026. Less is more: Scalable visual navigation from limited data. *arXiv preprint arXiv:2601.17815*.

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

Jacob Krantz and 1 others. 2023. Iterative vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16534–16544.

Hao Li and 1 others. 2026. Cmmr-vln: Vision-and-language navigation via continual multimodal memory retrieval. *arXiv preprint arXiv:2603.07997*.

Maxime Oquab and 1 others. 2024. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*. ArXiv preprint arXiv:2304.07193.

Sandeep Routray, Hengkai Pan, Unnat Jain, Shikhar Bahl, and Deepak Pathak. 2026. Vipra: Video prediction for robot actions. In *International Conference on Learning Representations*. ArXiv preprint arXiv:2511.07732.

Mingchen Song, Xiang Deng, Guoqiang Zhong, Qi Lv, Jia Wan, Yinchuan Li, Jianye Hao, and Weili Guan. 2025. Few-shot vision-language action-incremental policy learning. *arXiv preprint arXiv:2504.15517*.

Xinshuai Song, Weixing Chen, Yang Liu, Weikai Chen, Guanbin Li, and Liang Lin. 2024. Towards long-horizon vision-language navigation: Platform, benchmark and method. *arXiv preprint arXiv:2412.09082*.

Beichen Wang, Juexiao Zhang, Shuwen Dong, Irving Fang, and Chen Feng. 2024. Vlm see, robot do: Human demo video to robot action plan via vision language model. *arXiv preprint arXiv:2410.08792*.

Meng Wei, Chenyang Wan, Jiaqi Peng, Xiqian Yu, Yuqiang Yang, Delin Feng, Wenzhe Cai, Chenming Zhu, Tai Wang, Jiangmiao Pang, and Xihui Liu. 2025a. Ground slow, move fast: A dual-system foundation model for gener-

alizable vision-and-language navigation. *arXiv preprint arXiv:2512.08186*.

Meng Wei and 1 others. 2025b. Streamvln: Streaming vision-and-language navigation via slowfast context modeling. *arXiv preprint arXiv:2507.05240*.

Yifan Xu and 1 others. 2025. Dream to recall: Imagination-guided experience retrieval for memory-persistent vision-and-language navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. ArXiv preprint arXiv:2510.08553.

Lihe Yang and 1 others. 2024. Depth anything v2. *Advances in Neural Information Processing Systems*. ArXiv preprint arXiv:2406.09414.

Seonghyeon Ye and 1 others. 2025. Latent action pretraining from videos. In *International Conference on Learning Representations*. ArXiv preprint arXiv:2410.11758.

Ming-Ming Yu, Yi Chen, Börje F. Karlsson, and Wenjun Wu. 2025. Ranger: A monocular zero-shot semantic navigation framework through visual contextual adaptation. *arXiv preprint arXiv:2512.24212*. To appear in ICRA 2026.

Arthur Zhang, Xiangyun Meng, Luca Calliari, Dong-Ki Kim, Shayegan Omidshafiei, Joydeep Biswas, Ali Agha, and Amirreza Shaban. 2025. Ventura: Adapting image diffusion models for unified task conditioned navigation. *arXiv preprint arXiv:2510.01388*.

Jiazhao Zhang, Kunyu Wang, Shaoan Wang, Minghan Li, Haoran Liu, Songlin Wei, Zhongyuan Wang, Zhizheng Zhang, and He Wang. 2024. Uni-navid: A video-based vision-language-action model for unifying embodied navigation tasks. *arXiv preprint arXiv:2412.06224*.

Qi Zheng and 1 others. 2024. Esceme: Vision-and-language navigation with episodic scene memory. *International Journal of Computer Vision*. ArXiv preprint arXiv:2303.01032.

Huayi Zhou and Kui Jia. 2025. Vlbiman: Vision-language anchored one-shot demonstration enables generalizable bimanual robotic manipulation. *arXiv preprint arXiv:2509.21723*.