SEENAV-AGENT: ENHANCING VISION-LANGUAGE NAVIGATION WITH VISUAL PROMPT AND STEP-LEVEL POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing Vision-Language Navigation (VLN) agents based on Large Vision-Language Models (LVLMs) often suffer from errors of perception, reasoning, and planning, which significantly hinder their navigation performance. To address these limitations, a novel VLN agent framework, named SeeNav-Agent, is proposed. First, to reduce hallucinations of the visual module of the VLN agent, a dual-view Visual Prompt (VP) technique is introduced, which can also improve the agent's understanding of current spatial states. Subsequently, a novel step-level Reinforcement Fine-Tuning (RFT) method, Step Reward Group Policy Optimization (SRGPO), is designed for post-training. In SRGPO, we first define verifiable process rewards for navigation, and then perform efficient step-level advantage estimation by randomly grouping navigation steps. SRGPO provides dense reward signals for the RFT of the VLN agent and enhances its planning capability. Experiments on both the EmbodiedBench Navigation (Embench-Nav) and VLN-CE validate the effectiveness of the proposed method. By introducing the zero-shot VP module, the GPT-4.1 achieves a navigation success rate of 86.7%, surpassing the current best LVLM by approximately 20 percentage points (pp) on Embench-Nav and achieves an 11.1 pp improvement on VLN-CE compared to the base model. With VP+SRGPO, the Qwen2.5-VL-3B model reaches a navigation success rate of 72.3%, outperforming the best existing LVLM model by 5.6 pp on Embench-Nav and achieves 8.9 pp improvement on VLN-CE compared to the baseline method. Moreover, compared to GRPO and GiGPO, the proposed SRGPO demonstrates significant improvements in training stability, convergence efficiency, and generalization capability.

1 INTRODUCTION

In Vision-and-Language Navigation (VLN) Yin et al. (2024); Chen et al. (2021); An et al. (2023), an agent is required to continuously interact with the environment based on the natural language instructions and finally reach a target location. In each step, the environment provides the agent with visual state of the current scene. The agent needs to analyze the current view in combination with the instruction and output an action plan for navigation, which makes the VLN a typical Vision-Language-Action (VLA) problem. However, as Figure 1 shows, a VLA model often encounter errors at the levels of perception, reasoning and planning, which also limit its performance on the VLN task Yang et al. (2025). Visual hallucination is a typical kind of perception error, where the agent may incorrectly claim to see an object that is not actually in its field of view, or fail to recognize an object that is present. Reasoning errors mainly include incorrect understandings of spatial relationships and improper reflections on environmental feedback. Planning errors mainly include invalid or infeasible actions, such as moving towards obstacles.

Among the aforementioned error types, visual hallucination in LVLMs is a major culprit behind their suboptimal performance in VLA tasks Zhang et al. (2026). Some of the existing works attempt to add additional markers to the input images, such as obstacle masks or potential path indicators, to introduce more explicit visual cues and improve the perception ability of the agent Goetting et al. (2025); Wang et al. (2024b). Such technique is also known as visual prompt (VP). However, existing works lack research on effectively utilizing multi-view inputs through VP, and systematic

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.

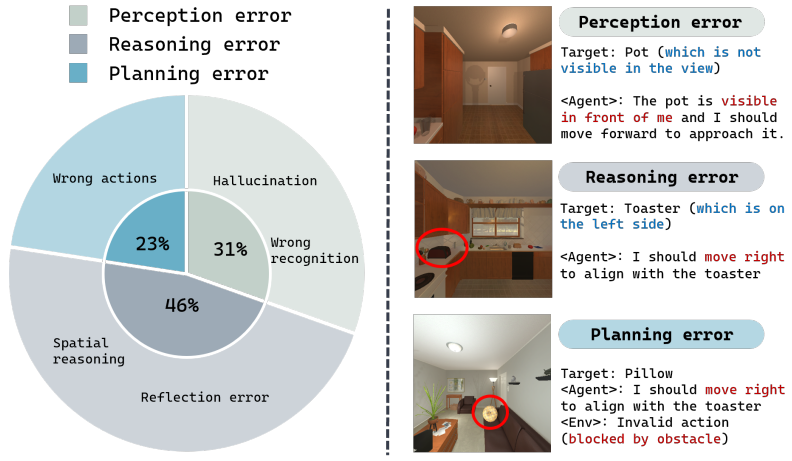

Figure 1: **The key motivations. Left:** Analysis for different error types from VLA models. **Right:** Examples of different error types in the navigation task.

analysis on how different VP types can complement each other is also missing. Besides, to mitigate errors in reasoning and planning, Supervised Fine-Tuning (SFT) and Reinforcement Fine-Tuning (RFT) are often applied to enhance VLN agents Liu et al. (2025); Qi et al. (2025); Gao et al. (2025). However, traditional RFT approaches only utilize the sparse outcome reward, while process-based RFT methods often calculate action advantage with respect to the identical anchor state Feng et al. (2025). When the anchor state is hard to define, the computation becomes inefficient and the scalability is limited. Overcoming the anchor-state limitation and developing an efficient, stable process-based RFT algorithm for navigation remains challenging.

To address above challenges, we introduce a novel framework named **SeeNav-Agent**. First, to reduce the hallucinations of the VLN agent and enhance its ability of spatial understanding, we introduce a dual-view visual prompt method. By introducing VP, the probability of perception and reasoning errors can be reduced. In addition, by using an action projection mechanism within VP, we transform the VLA task into a Visual Question Answering (VQA) task, thereby reducing the difficulty of the navigation. Subsequently, we further enhance the agent's reasoning and planning abilities through RFT. During RFT, a verifiable process reward for each navigation step is designed. We further propose the **S**tep **R**eward **G**roup **P**olicy **O**ptimization (**SRGPO**) algorithm, which utilizes the process reward signals by randomly grouping navigation steps and computing relative advantages.

The main contribution of this work is summarized as:

- We propose a dual-view visual prompt technique for navigation task, which significantly reduces the visual hallucinations and improves the spatial understanding capability of the VLN Agent with a zero-shot setting.
- We design a novel RFT algorithm, namely SRGPO, which efficiently integrates the step-level rewards of the navigation process, thereby further enhancing the agent's reasoning and planning abilities.
- Experiments on Embench-Nav and VLN-CE show that GPT4.1+VP surpasses the SOTA closed-source model by 20 pp (zero-shot) on Embench-Nav and by 11.1 pp on VLN-CE, and Qwen2.5-VL-3B-Instruct+VP+SRGPO exceeds the previous SOTA closed-source model by 5.6 pp on Embench-Nav and improves over the baseline by 8.9 pp on VLN-CE. Moreover, SRGPO shows superior convergence, stability, and generalization.

2 Related Work

Vision-and-Language Navigation (VLN). VLN is a type of multimodal embodied VLA task. Traditional VLN pipelines treat image perception, map node segmentation, reasoning and action planning as independent modules, and the agent usually moves in a discrete environment Cao et al.

(2024); Majumdar et al. (2020); Ku et al. (2020); Yin et al. (2025). Recent research based-on LVLMs has provided a unified solution for VLN tasks. The LVLMs-based VLN agents use powerful multimodal models as the backbone, and fuse perception, reasoning, and planning in an end-to-end manner for continuous navigation Zhang et al. (2025); Liu et al. (2025). These works either use closed-source models directly for navigation Goetting et al. (2025) or fine-tune open-source ones Qi et al. (2025); Gao et al. (2025).

Visual Prompt (VP) for VLN Agents. VP methods add visual cues such as bounding boxes, color blocks, and numerical marks to the input image so that LVLMs can better handle visual tasks Shtedritski et al. (2023); Yang et al. (2023); Nasiriany et al. (2024). Since LVLM pre-training corpora are dominated by VQA data, LVLMs are inherently adept at VQA, and VP allows planning-style VLA tasks like VLN to be reformulated as zero-shot VQA. PIVOT Nasiriany et al. (2024) projects candidate actions as numbered arrows and iteratively optimizes the action distribution via LVLM selection. VLMnav Goetting et al. (2025) annotates walkable-area masks and farthest-reachable arrows, and similarly lets the agent select an optimal action from projected candidates. However, the aforementioned works lack systemic research on how to effectively utilize multi-view inputs through VP and how to coordinate different VP modules for better performance.

Reinforcement Fine-Tuning (RFT) for VLN Agents. RFT is a common post-training method for VLN agents. Proximal Policy Optimization (PPO) Zhai et al. (2024) and Group Relative Policy Optimization (GRPO) Qi et al. (2025); Gao et al. (2025); Li et al. (2025) are widely used RFT frameworks, but classic GRPO relies only on sparse outcome rewards, which is inefficient for long-horizon navigation. Process rewards have been shown to effectively enhance RFT Wang et al. (2025; 2023); Chan et al. (2024). GiGPO Feng et al. (2025) incorporates step-level rewards by rolling out trajectories from the same initial state and grouping identical states. However, identical states are hard to define in continuous navigation, and achieving sufficient group sizes requires many rollouts, causing a heavy computational load. In this work, a robust and verifiable process reward for navigation tasks is designed for obtaining step-level reward signals. Meanwhile, we improve the GRPO framework and propose SRGPO, an RFT algorithm based on random step grouping, which efficiently addresses the issue in GiGPO that requires grouping of identical states.

3 METHODOLOGY

3.1 FRAMEWORK AND PROBLEM FORMULATION

In this section, we introduce a novel LVLM-based embodied navigation framework named SeeNav-Agent. As shown in Figure 2, SeeNav-Agent includes a specifically designed dual-view visual prompt technique for the input to reduce perception hallucinations and to enhance the spatial understanding capability of the agent. During RFT, to overcome the limitations of previous RFT algorithms like GRPO and GiGPO, we propose a novel algorithm named SRGPO that can efficiently utilize both episode-level and step-level navigation rewards in the advantage estimation stage.

We first formalize the LVLM-based VLN. The input state of the VLN agent at step t is $\mathbf{O}_t = \{\mathbf{I}, \mathbf{S}, \mathbf{V}_t, \mathbf{H}_t\}$, where $\mathbf{I}$ is the human instruction; $\mathbf{S}$ represents the action strategy given by humans, which is embedded in the input context; $\mathbf{V}_t$ denotes the environmental image observed by the agent at step t and $\mathbf{H}_t$ is the action interaction history before step t. In this work, we use an action interaction history with a time window as $\mathbf{H}_t = (\mathbf{a}_{t-T_H}, \mathbf{f}_{t-T_H}, \mathbf{a}_{t-T_H+1}, \mathbf{f}_{t-T_H+1}, \ldots, \mathbf{a}_{t-1}, \mathbf{f}_{t-1})$, where $\mathbf{a}_t$ refers to the action at step t, and $\mathbf{f}_t$ denotes the feedback from the environment for this action. Both $\mathbf{a}_t$ and $\mathbf{f}_t$ are presented in text form. T_H represents the time window for action history. The output of the agent in each step is $(\mathbf{c}_t, \mathbf{a}_t)$, where $\mathbf{c}_t$ represents the analysis and thinking process about the current state, and $\mathbf{a}_t \in \mathcal{A}$ is the action decision chosen from the action space $\mathcal{A}$. The LVLM-based VLN agent is parameterized as a policy network $\pi_\theta(\mathbf{a}_t|\mathbf{O}_t)$ with learnable parameters θ.

3.2 DUAL-VIEW VISUAL PROMPT FOR VLN AGENT

Most existing VLN works tend to use only a single perspective as the input for the agent, such as a front view (FV) or a bird's-eye view (BEV). However, due to the poor ability of LVLM itself to understand spatial information like depth of field, the spatial relationships of objects in the FV are not as clear as those in the BEV for LVLM. But the problem with relying solely on the BEV is

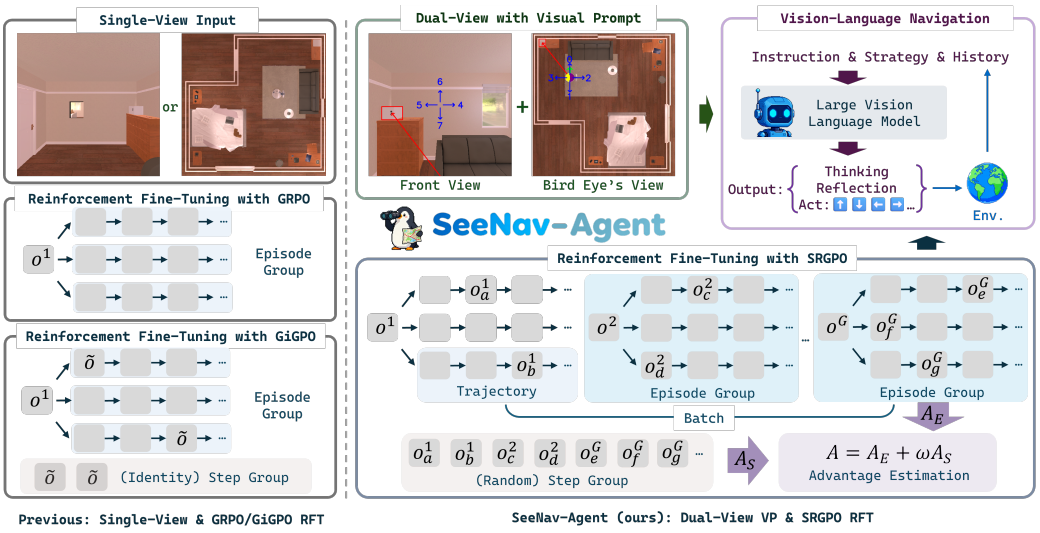

Figure 2: **Overview of SeeNav-Agent.** Different from previous VLN works tend to use single view image as input and use method like GRPO or GiGPO for RFT, our SeeNav-Agent designs a dual-view input with visual prompt to enhance the visual module in a zero-shot manner, and proposes SRGPO to efficiently introduce process reward signals by randomly grouping steps. The upper-middle schematic shows the dual-view visual prompt, with FV on the left and BEV on the right.

that the shapes of objects in it are often different from common representations, so an LVLM may fail to recognize a target object in the BEV. Therefore, we consider enabling the agent to see both FV and BEV simultaneously. This dual-view (DV) input can enrich the visual information, but it also increases the difficulty for the agent to process and understand the images. If utilized inappropriately, the performance of the agent may not improve but instead decline Yang et al. (2025). Consequently, we design a series of VP modules as follows to help the agent better understand the visual information. An example of these VP modules is shown in the upper-middle of Figure 2.

Bounding box. For navigation tasks such as "moving toward a target object", incorrect judgment of the target's existence is one of the most common visual hallucinations. The agent may claim to see an object that does not exist in the view, or ignore a target that is clearly within its field of view, leading to decision-making errors. We enhance the existence information of the target by drawing bounding boxes (BB) around it in both FV and BEV, and also add relevant prompts to reduce hallucinations in judging the target existence.

Navigation line. Inspired by the guidance trajectories in the minimaps of open-world games, we add navigation line (NL) arrows from the agent to the target. Since FV presents the first-person perspective of the agent, the starting point of the NL is set at the midpoint of the bottom edge. In BEV, however, the starting point of the NL is the agent's center point. NL can enhance the agent's understanding of the relative position and distance between itself and the target.

Agent marker. In navigation, enabling the agent to understand its position in space helps it plan the next action. Therefore, we use the agent marker (AM) represented by a circle and an arrow in the BEV to indicate the position and orientation of the agent. Additionally, since the agent often makes mistakes in understanding left-right relationships, we mark the left and right sides of the circle with different colors (yellow for left, purple for right, and a green arrow for the agent's front), which enhances its comprehension of the relative positional relationship between itself and the target.

Action projection. The main actions in navigation include movement in various directions and perspective rotation, which can be projected onto images. Through action projection (AP), the agent can partly observe the consequences of executing a specific action. Besides, it can also convert a planning problem into VQA of selecting the optimal action on the image, reducing the difficulty of navigation. We draw movement actions in the BEV and draw rotation actions in the FV with arrows and action IDs (0–7 in the Embench-Nav illustration), thus projecting the action space.

View alignment. In BEV, the agent can face to various directions due to the randomness of the BEV orientation, which may lead the agent to mistakenly understand the left-right and front-back relationships between itself and objects (such errors may still occur even with the AM). Therefore, we rotate the BEV according to the agent's orientation, ensuring that the agent always faces forward in the BEV, which means its viewing direction is consistent with FV.

Actually, VP does not require every cue from privileged state: only the target bounding box and the BEV need external perception. Navigation lines and action projections are derived from these plus the agent pose. In this work we directly use the environment's intrinsic information to obtain BEV and bounding boxes, which is a natural choice for simulation, games, and other environments where the full state is accessible. For settings that are closer to the real world, one can rely on additional auxiliary models or tools to obtain the required information. For example, works like BEVFormer Li et al. (2024) and CorrBEV Xue et al. (2025) focus on BEV modeling, and there is extensive research on vision-based object detection Zhang et al. (2022b;a); Wang et al. (2024a). Our proposed VP approach is a visual interface orthogonal to and decoupled from its signal source.

3.3 Step Reward Group Policy Optimization

While VP can mitigate perception error, further addressing reasoning and planning errors still requires post-training with RFT. Classic GRPO Shao et al. (2024) learns from the outcome reward, however, for agentic problems like VLN that require long-sequence interactions with the environment, GRPO suffers from sparse reward signals, and how to assign credit to each step becomes a challenging problem. Existing work like GiGPO Feng et al. (2025) proposes a step-level optimization method based on the identical anchor state. However, the definition of identical states is relatively strict, and for problems where the anchor state is hard to define, the computational burden will significantly increase. To solve this challenge, we propose SRGPO with verifiable process rewards, which leverages the unique properties of navigation tasks. The core insight of SRGPO is that if we can define a process reward that is almost independent of the state, a flexible grouping of steps and advantage estimation can be achieved. SRGPO overcomes the drawbacks of GRPO, which fails to consider process rewards, and the strict prerequisite of GiGPO, which requires grouping identical states, thereby achieving efficient utilization of process reward signals during the navigation process.

Verifiable process reward for navigation. For navigation, whether the final result is successful or not is a clear outcome reward. However, for each navigation step, the quality of each action can also be judged. For example, an action that brings the agent closer to the target, or pulls the target back into the FV, can be considered good. We denote the agent position at time step t as p_t, and the target object as g. The visible object set from FV at time step t is defined as $\mathcal{F}_t$. We define the verifiable process reward (VPR) R_t^s for navigation as follows:

$$
\begin{aligned}
R_t^s &= R_{t,\text{base}}^s - \lambda_{\text{valid}} \cdot R_{t,\text{valid}}^s, \\
R_{t,\text{base}}^s &= \begin{cases} 1, & \text{if } \operatorname{dist}(p_{t+1}, g) < \operatorname{dist}(p_t, g) \text{ or } \mathbf{1}_{\{g \in \mathcal{F}_{t+1}\}} > \mathbf{1}_{\{g \in \mathcal{F}_t\}}, \\ 0, & \text{otherwise}, \end{cases} \\
R_{t,\text{valid}}^s &= \mathbf{1}_{\{\mathbf{a}_t \notin \mathcal{A}_t^{\text{feasible}}\}}.
\end{aligned} \tag{1}
$$

where $\operatorname{dist}(p_t, g)$ is the distance between the agent and the target at step t, and $g \in \mathcal{F}_t$ means the target is in the FV at step t. Since $\mathbf{a}_t$ is selected from $\mathbf{O}_t$ and executed to reach p_{t+1}, R_t^s evaluates the transition caused by $\mathbf{a}_t$. $R_{t,\text{valid}}^s$ describes the penalty for invalid actions, where $\mathcal{A}_t^{\text{feasible}} \subseteq \mathcal{A}$ is the set of actions that can be successfully executed at step t (*e.g.*, not blocked by obstacles), as reported by the environment feedback $\mathbf{f}_t$. The design of the process reward relies solely on the distance changes between the agent and the target for two consecutive steps, as well as the changes in the visibility of the target in the FV. The key insight of this reward design is that it is independent of the specific environmental state: a reward of 1 always denotes verifiable local progress toward the target and 0 the absence of it. Therefore, the quality of actions can be compared under any two different states, which is the prerequisite for the design of SRGPO.

Episode-level advantages. We first introduce the episode-level advantage of vanilla GRPO. We roll out N trajectories $\{\boldsymbol{\tau}_i\}_{i=1}^N$ using the agent policy $\pi_{\theta_{\text{old}}}$ for one task, where each trajectory $\boldsymbol{\tau}_i = \{(\mathbf{O}_{i,0}, \mathbf{a}_{i,0}), (\mathbf{O}_{i,1}, \mathbf{a}_{i,1}), \ldots, (\mathbf{O}_{i,T_i-1}, \mathbf{a}_{i,T_i-1})\}$. The outcome reward for each trajectory is defined as $R(\boldsymbol{\tau}_i)$. For navigation, the reward is 1 if the agent completes the task and 0 otherwise.

Then, the episode-level group and the corresponding normalized advantage are:

$$G_E = \{(\boldsymbol{\tau}_1, R(\boldsymbol{\tau}_1)), \ldots, (\boldsymbol{\tau}_N, R(\boldsymbol{\tau}_N))\}, \tag{2}$$

$$A_E(\boldsymbol{\tau}_i) = \frac{R(\boldsymbol{\tau}_i) - \text{mean}(\{R(\boldsymbol{\tau}_j)\}_{j=1}^N)}{\text{std}(\{R(\boldsymbol{\tau}_j)\}_{j=1}^N)}. \tag{3}$$

Step-level advantages. The episode-level reward signals are relatively sparse, while RFT tends to benefit from denser reward signals. Therefore, we use verifiable process reward signals to calculate step-level advantages. Similar to GiGPO, calculating step-level advantages requires grouping steps first. However, unlike GiGPO, since the verifiable process rewards designed in Equation (1) do not depend on specific states, we do not need to define the anchor states which describe state identity as in GiGPO. The core idea is that, through the definition of verifiable process rewards, we discard the prerequisite that "to compare the quality of two actions, it must be done under the same state". Therefore, when calculating step-level advantages, we can change the concept of groups in the episode-level advantage calculation and instead randomly sample steps from the *entire batch* to form step groups. Assuming one batch contains B tasks, we have $N \times B$ trajectories, and the step-level groups with size N_S can be defined as:

$$G_S = \left\{(\mathbf{c}_t^{(i)}, \mathbf{a}_t^{(i)}, R_t^{s(i)})\right\}_{N_S}, \quad t \in \{0, \ldots, T_i - 1\},\ i \in \{1, \ldots, N \times B\}, \tag{4}$$

where i and t are randomly sampled from their valid range, and T_i represents the episode length of the i-th trajectory. The step-level advantage for a step within the group is then:

$$A_S(\mathbf{c}_t^{(i)}, \mathbf{a}_t^{(i)}) = \frac{R_t^{s(i)} - \text{mean}_{G_S}(R^s)}{\text{std}_{G_S}(R^s)}, \tag{5}$$

where $\text{mean}_{G_S}(R^s)$ and $\text{std}_{G_S}(R^s)$ are the mean and standard deviation of process rewards in G_S.

SRGPO training objective. The two levels of advantages are combined into a unified advantage for both step-level and trajectory-level credit assignment. With balance coefficient ω, the combined advantage and the SRGPO objective are:

$$A_{i,t} = A_E(\boldsymbol{\tau}_i) + \omega \cdot A_S(\mathbf{c}_t^{(i)}, \mathbf{a}_t^{(i)}), \tag{6}$$

$$\begin{aligned}\mathcal{J}_{\text{SRGPO}}(\theta) = \mathbb{E}_{\{\boldsymbol{\tau}_i\}_{i=1}^N \sim \pi_{\theta_{\text{old}}}} \Bigg[\frac{1}{N}\sum_{i=1}^{N} \frac{1}{T_i}\sum_{t=0}^{T_i-1} \Big(\min\big(\rho_{i,t}(\theta) A_{i,t}, \text{clip}(\rho_{i,t}(\theta), 1 \pm \epsilon) A_{i,t}\big) \\ - \beta\, \mathbb{D}_{\text{KL}}\big(\pi_\theta(\cdot|\mathbf{O}_{i,t}) \,\|\, \pi_{\text{ref}}(\cdot|\mathbf{O}_{i,t})\big)\Big)\Bigg],\end{aligned} \tag{7}$$

where $\rho_{i,t}(\theta) = \frac{\pi_\theta(\mathbf{c}_t^{(i)}, \mathbf{a}_t^{(i)}|\mathbf{O}_{i,t})}{\pi_{\theta_{\text{old}}}(\mathbf{c}_t^{(i)}, \mathbf{a}_t^{(i)}|\mathbf{O}_{i,t})}$ is the importance sampling ratio and β is the KL penalty coefficient. Both the importance ratio and the KL penalty are computed over the complete generated response of each step, *i.e.*, the reasoning tokens $\mathbf{c}_t$ together with the action tokens $\mathbf{a}_t$, and the same advantage $A_{i,t}$ is shared by all tokens of this response.

4 Experiments

4.1 Experimental Setup

Benchmark. We evaluate SeeNav-Agent on Embench-Nav Yang et al. (2025) and VLN-CE Krantz et al. (2020). Embench-Nav contains 60 AI2-THOR Kolve et al. (2017) in-room tasks; VLN-CE is a Habitat-Sim Puig et al. (2023) continuous indoor benchmark. Action spaces, scene parameters, and data splits are given in Appendix A.

Baselines and metrics. We compare against (i) closed-source LVLMs, (ii) open-source LVLMs, and (iii) post-training methods (SFT, GRPO, GiGPO), using Qwen2.5-VL-3B-Instruct Bai et al. (2025) as the backbone for RFT. The main metric is success rate (SR); on VLN-CE we also report SPL, NE, and a comparison with VLN-R1 Qi et al. (2025).

Implementation details. Hyper-parameters are shared across both benchmarks: $\lambda_{\text{valid}} = 0.1$, $\omega = 0.5$ in Equation (6), $N = 4$, $N_S = 16$, and $T_H = 5$. VP is kept throughout RFT, and dual-view inputs concatenate FV (left) and BEV (right). Other setup details are in the appendix.

Table 1: **Main Results.** RFT cases are trained with i.d. settings

Method	Embench-Nav SR↑	VLN-CE(R2R) SR↑	VLN-CE(R2R) SPL↑	VLN-CE(R2R) NE↓
Close-Source LVLMs				
Claude-3.5-Sonnet	0.667			
GPT4.1	0.650	0.141	0.117	8.8
Gemini-2.0-flash	0.633			
Qwen-VL-Max	0.500			
Open-Source LVLMs				
InternVL3-78B	0.667			
Qwen2.5-VL-72B-Ins	0.467			
Qwen2.5-VL-7B-Ins	0.200			
Qwen2.5-VL-3B-Ins	0.167	0.112	0.083	9.7
Post-Train baselines (backbone: Qwen2.5-VL-3B-Ins)				
VP+SFT	0.367			
VP+GRPO	0.408±0.008	0.230±0.067	0.191±0.072	8.6±2.4
VP+GiGPO (vanilla)	0.294±0.028	0.226±0.070	0.190±0.067	8.6±2.6
VP+GiGPO (w. VPR)	0.572±0.080	0.281±0.042	0.220±0.041	7.8±1.9
VLN-R1 *(Qwen2-VL-7B)* Qi et al. (2025)	-	0.302	0.218	7.0
SeeNav-Agent (ours)				
VP-only *(Qwen2.5-VL-3B)*	0.283	0.184	0.165	7.8
SRGPO-only *(Qwen2.5-VL-3B)*	0.382±0.090	0.246±0.102	0.211±0.081	8.3±2.5
VP+SRGPO *(Qwen2.5-VL-3B)*	**0.723±0.008** (↑0.556)	**0.391±0.021**(↑0.279)	**0.327±0.020**(↑0.244)	**5.1±1.8**(↓4.6)
GPT4.1+VP *(GPT4.1)*	**0.867 (↑0.217)**	0.252(↑0.111)	0.208(↑0.091)	7.0(↓1.8)

Table 2: **Ablation Study for Visual Prompt on Embench-Nav.** All combinations are based on GPT4.1 and comb.(A) is the FV only baseline.

Comb.	DV	BB	AP	AM	NL	VA	SR
(A)							0.650
(B)	✓					✓	0.450
(C)	✓		✓	✓		✓	0.650
(D)	✓	✓	✓	✓	✓		0.583
(E)	✓	✓		✓	✓	✓	0.683
(F)	✓	✓	✓	✓		✓	0.800
(G)	✓	✓	✓		✓	✓	0.800
(H)	✓	✓	✓	✓	✓	✓	**0.867**

Table 3: **Grouping Strategy Comparison on Embench-Nav.** Both localized alternatives underperform batch-level random grouping.

Grouping strategy	Scope	SR↑
Random step grouping (SRGPO)	batch-level	0.723±0.008
Within-task grouping	task-level	0.633±0.031
Within-trajectory grouping	trajectory-level	0.583±0.012

4.2 Main Results

The main results are shown in Table 1. Compared with the close-source models, our VP technique improves the SR of GPT4.1 by 21.7 pp and surpasses the best Claude-3.5-Sonnet by 20.0 pp on Embench-Nav, and achieves an 11.1 pp improvement of SR on VLN-CE. Compared with the open-source models, our method achieves a 55.6 pp improvement on the base model Qwen2.5-VL-3B-Ins and surpasses the best InternVL3-78B by 5.6 pp on Embench-Nav, and beats the baseline VLN-R1 by 8.9 pp of SR on VLN-CE. Moreover, our SRGPO significantly outperforms post-training baselines including SFT, GRPO and GiGPO, confirming the effectiveness of SeeNav-Agent.

4.3 Ablation Study for Visual Prompt

We conducted ablation experiments on each VP modules on the Embench-Nav benchmark and the results are shown in Table 2. The results show that by introducing VP, a 21.7 pp improvement in navigation success rate can be achieved with a zero-shot manner. From the ablation results, it can be suggested that simply introducing dual-view (DV) input and view alignment (VA) cannot improve the success rate, instead, it may even lead to a decline in the agent's performance. The reason is that it is difficult for the agent to accurately understand and align the spatial information brought by the two different views. Furthermore, the results also indicate that each module in the VP recipe works collectively. It is necessary to include all VP modules simultaneously to achieve the optimal navigation success rate. Among these modules, bounding-boxes (BB), action projection (AP), and view alignment (VA) are relatively more important, which means removing any one of them will lead to a significant drop in the success rate of the navigation task.

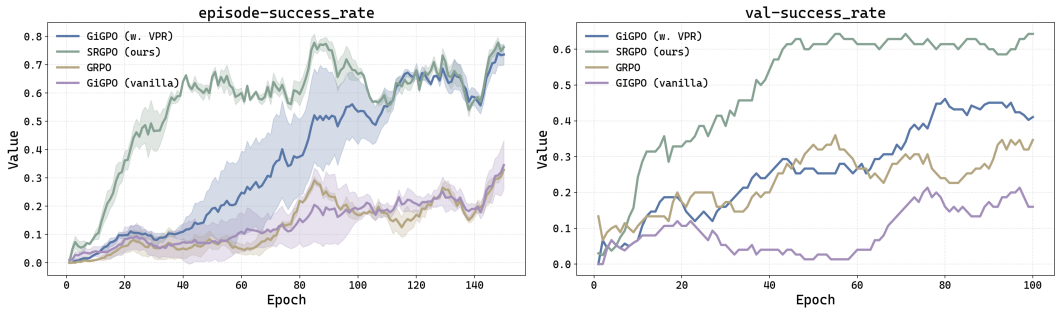

Figure 3: **i.d and o.o.d Curves on Embench-Nav. Left:** i.d training curves, where the solid line and the shaded region represent the mean and standard deviation obtained from multiple training runs. **right:** o.o.d. validation curves, where all algorithms are trained on o.o.d. scenes and tested on base scenes of the Embench-Nav.

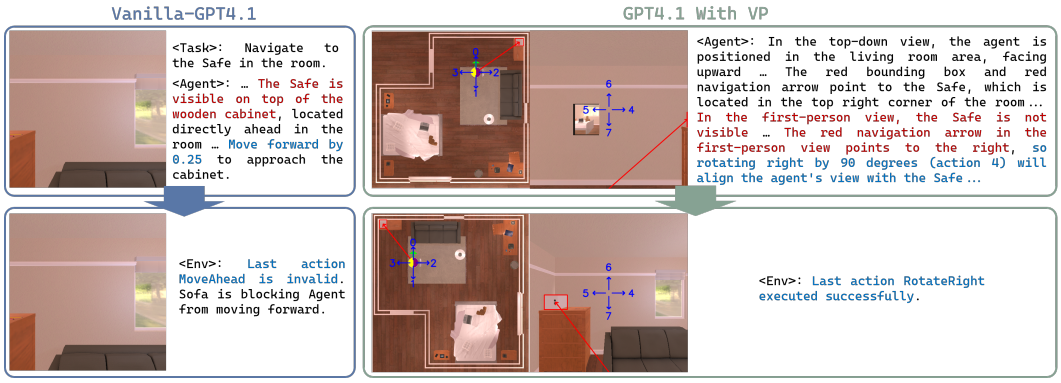

Figure 4: **Case Comparison of vanilla-GPT 4.1 and GPT4.1 with VP on Embench-Nav.** The bottom part shows the image and text feedback from the environment after taking actions. This case clearly shows that the dual-view VP technique reduces visual hallucination and strengthens the VLN agent's spatial understanding.

Figure 4 shows the differences in the thinking and acting processes of GPT4.1 before and after introducing VP modules when the agent loses sight of the target in FV. In this scenario, the agent is instructed to move to the *Safe*. The reasoning process of the vanilla GPT4.1 model exhibits hallucination, mistakenly believing that the *Safe* is still within its field of view. Consequently, it decides to move forward and fails due to an obstacle. In contrast, after applying VP, the agent can correctly detect that the target is out of sight. Furthermore, with the information from NL, the agent identifies that the target should be on its right side. Finally, among the projected actions, the agent selects action 4 for view rotation and brings the target back into its field of view.

4.4 Effectiveness of SRGPO

The i.d. training curves on Embench-Nav of GRPO, GiGPO and SRGPO are shown in the left part of Figure 3. We conduct three runs for each method. For GiGPO, we test both the step reward from the original paper "VP+GiGPO(vanilla)" and the verifiable step reward proposed in this work "VP+GiGPO(w. VPR)". As can be seen from the figure, "VP+SRGPO" significantly outperforms "VP+GRPO" and "VP+GiGPO(vanilla)" in both convergence speed and final training success rate. Besides, it also demonstrates the best training stability, as evidenced by the very small standard deviation of its training curves. However, when GiGPO adopts the verifiable step reward, it can converge to a success rate close to that of SRGPO, but its convergence speed is still much slower. The reason is that although GiGPO is capable of grouping steps, its grouping efficiency is lower than that of SRGPO due to the constraint of identical states. In other words, the number of steps in each group is relatively small. After the introduction of the verifiable step reward, GiGPO becomes

similar to SRGPO with a small number of steps per group. Although "VP+GiGPO(w. VPR)" can achieve a similar success rate to "VP+SRGPO" during training, "VP+SRGPO" demonstrates much faster convergence and better stability.

Furthermore, to compare the navigation performance of agents after RFT, we evaluate the SR on the Embench-Nav test scenarios using the last-epoch models. As shown in Table 1, the testing navigation SR of "VP+SRGPO" is 15.1 pp higher than that of "VP+GiGPO(w. VPR)", and 31.5 pp higher than that of "VP+GRPO", indicating that the agent trained with SRGPO possesses superior generalization performance. On the more complex VLN-CE environment, our method also outperforms GRPO and GiGPO, and compared to VLN-R1, which uses historical frames in its input, our method achieves significant improvements in SR, SPL, and NE, validating the scalability of SRGPO.

In addition, the experimental results also show that although SFT alone can effectively improve the navigation ability of the agent, it may negatively impact the performance of SRGPO. This phenomenon may be caused by the reduction in policy diversity after SFT, which in turn restricts the agent's capacity for exploration during the RFT process.

We also compared different grouping strategies for SRGPO. As reported in Table 3, two more localized alternatives (within-task and within-trajectory grouping) both underperform batch-level random grouping. The reason is that overly easy or hard tasks yield little intra-group reward variance, whereas batch-level sampling preserves diversity by mixing steps from tasks of different difficulty.

4.5 Out-Of-Domain Generalization Capability

To further evaluate the generalization ability of the model trained with SRGPO, we switched the training scenes of Embench-Nav to o.o.d. setting. During the training process, the navigation success rates of models trained with different algorithms on the test set change as illustrated in the right part of Figure 3. From the figure, it can be seen that the testing curve of SRGPO is significantly higher than those of the other three algorithms, indicating that even when trained in o.o.d. scenes, the model trained by SRGPO demonstrates stronger navigation capabilities than the other models. This indicates that SRGPO enables the model to acquire more general navigation knowledge, confirming its superior generalization.

5 Conclusion

In this work, we propose SeeNav-Agent, a novel LVLM-based embodied navigation framework that includes a zero-shot dual-view visual prompt technique for the input side and an efficient RFT algorithm SRGPO for post-training. With the dual-view VP modules, the visual hallucinations can be significantly reduced and the spatial understanding capability of the VLN Agent can be improved. Through systematic analysis and ablation experiments, we identified the optimal VP recipe, which enables the best alignment and utilization of information from dual-view inputs. Based on the defined verifiable process reward for navigation that is independent of the state, the proposed SRGPO algorithm can randomly group steps for advantage estimation and thereby efficiently utilize the step reward signals. Compared with GRPO and GiGPO, SRGPO not only addresses their inherent limitations, but also demonstrates significant advantages in training stability, convergence efficiency, and generalization ability. On Embench-Nav, "GPT4.1+VP" achieves SR surpassing the previous SOTA close-source model by 20 pp, while "VP+SRGPO" exceeds the previous SOTA model by 5.6 pp. On VLN-CE, our method exceeds VLN-R1 by 8.9 pp in SR and outperforms it on SPL and NE, showing the superiority and scalability of SeeNav-Agent.

A limitation of our method is that the process reward proposed in this work is not applicable to complex navigation tasks that require traversing between floors. For such problems, designing more ingenious and general rewards is a worthwhile direction for future research. Although this work focuses on the navigation task, we may also consider extending the proposed SRGPO to other tasks where a state-independent process reward can be well-defined.

AI use statement

We used generative AI tools to identify related literature, suggest the structure of the paper, draft and polish parts of the text, and assist in implementing parts of the experimental code. All AI-assisted

code was tested by the authors, and all references and AI-assisted text were verified 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

This work does not raise any ethical concerns. All experiments are conducted entirely in simulation using publicly available benchmarks (Embench-Nav built on AI2-THOR, and VLN-CE built on Habitat-Sim) and publicly available models or APIs. No human subjects, personally identifiable information, or private data are involved, and we are not aware of any potential harmful applications or conflicts of interest arising from this work.

Reproducibility statement

We have made efforts to ensure the reproducibility of our results. The dual-view visual prompt and the SRGPO algorithm are described in Sections 3.2 and 3.3, including the process reward, the episode-level and step-level advantages, and the training objective (Equations (1), (3) and (5) to (7)). The backbone model (Qwen2.5-VL-3B-Instruct), baselines, evaluation metrics, and all hyper-parameters are specified in Section 4.1. Appendix A provides the environment parameters, action spaces, and data splits for Embench-Nav and VLN-CE, the scene randomization protocol for in-distribution and out-of-distribution training, the compute resources and training time (Appendix A.3), and the complete language prompt template together with an example of the interaction history (Tables 4 and 5). Results of our method are reported as mean and standard deviation over multiple training runs.

References

Dong An, Yuankai Qi, Yangguang Li, Yan Huang, Liang Wang, Tieniu Tan, and Jing Shao. Bevbert: Multimodal map pre-training for language-guided navigation. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Yihan Cao, Jiazhao Zhang, Zhinan Yu, Shuzhen Liu, Zheng Qin, Qin Zou, Bo Du, and Kai Xu. Cognav: Cognitive process modeling for object goal navigation with llms. *arXiv preprint arXiv:2412.10439*, 2024.

Alex J Chan, Hao Sun, Samuel Holt, and Mihaela Van Der Schaar. Dense reward for free in reinforcement learning from human feedback. *arXiv preprint arXiv:2402.00782*, 2024.

Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. History aware multimodal transformer for vision-and-language navigation. In *NeurIPS*, 2021.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025.

Chen Gao, Liankai Jin, Xingyu Peng, Jiazhao Zhang, Yue Deng, Annan Li, He Wang, and Si Liu. Octonav: Towards generalist embodied navigation. *arXiv preprint arXiv:2506.09839*, 2025.

Dylan Goetting, Himanshu Gaurav Singh, and Antonio Loquercio. End-to-end navigation with vision-language models: Transforming spatial reasoning into question-answering. In *Proceedings of the International Conference on Neuro-symbolic Systems*, pp. 22–35, 2025.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

Jacob Krantz, Erik Wijmans, Arjun Majundar, Dhruv Batra, and Stefan Lee. Beyond the navgraph: Vision and language navigation in continuous environments. In *European Conference on Computer Vision (ECCV)*, 2020.

Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-Across-Room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2020.

LinFeng Li, Jian Zhao, Yuan Xie, Xin Tan, and Xuelong Li. Compassnav: Steering from path imitation to decision understanding in navigation. *arXiv preprint arXiv:2510.10154*, 2025.

Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Qiao Yu, and Jifeng Dai. Bevformer: learning bird's-eye-view representation from lidar-camera via spatiotemporal transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(3):2020–2036, 2024.

Qingxiang Liu, Ting Huang, Zeyu Zhang, and Hao Tang. Nav-r1: Reasoning and navigation in embodied scenes. *arXiv preprint arXiv:2509.10884*, 2025.

Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. Improving vision-and-language navigation with image-text pairs from the web. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

Soroush Nasiriany, Fei Xia, Wenhao Yu, Ted Xiao, Jacky Liang, Ishita Dasgupta, Annie Xie, Danny Driess, Ayzaan Wahid, Zhuo Xu, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. *arXiv preprint arXiv:2402.07872*, 2024.

Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander William Clegg, Michal Hlavac, So Yeon Min, et al. Habitat 3.0: A co-habitat for humans, avatars and robots. *arXiv preprint arXiv:2310.13724*, 2023.

Zhangyang Qi, Zhixiong Zhang, Yizhou Yu, Jiaqi Wang, and Hengshuang Zhao. Vln-r1: Vision-language navigation via reinforcement fine-tuning. *arXiv:2506.17221*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Aleksandar Shtedritski, Christian Rupprecht, and Andrea Vedaldi. What does clip know about a red circle? visual prompt engineering for vlms. *arXiv preprint arXiv:2304.06712*, 2023.

Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Yolov10: Real-time end-to-end object detection. *arXiv preprint arXiv:2405.14458*, 2024a.

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

Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. Spa-rl: Reinforcing llm agents via stepwise progress attribution. *arXiv preprint arXiv:2505.20732*, 2025.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2023.

Ziteng Xue, Mingzhe Guo, Heng Fan, Shihui Zhang, and Zhipeng Zhang. Corrbev: Multi-view 3d object detection by correlation learning with multi-modal prototypes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27413–27423, June 2025.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*, 2023.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, et al. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *arXiv preprint arXiv:2502.09560*, 2025.

Hang Yin, Xiuwei Xu, Zhenyu Wu, Jie Zhou, and Jiwen Lu. Sg-nav: Online 3d scene graph prompting for llm-based zero-shot object navigation. *arXiv preprint arXiv:2410.08189*, 2024.

Hang Yin, Xiuwei Xu, Linqing Zhao, Ziwei Wang, Jie Zhou, and Jiwen Lu. Unigoal: Towards universal zero-shot goal-oriented navigation. *arXiv preprint arXiv:2503.10630*, 2025.

Simon Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Peter Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, et al. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. *Advances in neural information processing systems*, 37:110935–110971, 2024.

Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022a.

Haotian* Zhang, Pengchuan* Zhang, Xiaowei Hu, Yen-Chun Chen, Liunian Harold Li, Xiyang Dai, Lijuan Wang, Lu Yuan, Jenq-Neng Hwang, and Jianfeng Gao. Glipv2: Unifying localization and vision-language understanding. *arXiv preprint arXiv:2206.05836*, 2022b.

Jianke Zhang, Xiaoyu Chen, Qiuyue Wang, Mingsheng Li, Yanjiang Guo, Yucheng Hu, Jiajun Zhang, Shuai Bai, Junyang Lin, and Jianyu Chen. Vlm4vla: Revisiting vision-language-models in vision-language-action models. *arXiv preprint arXiv:2601.03309*, 2026.

Lingfeng Zhang, Yuecheng Liu, Zhanguang Zhang, Matin Aghaei, Yaochen Hu, Hongjian Gu, Mohammad Ali Alomrani, David Gamaliel Arcos Bravo, Raika Karimi, Atia Hamidizadeh, et al. Mem2ego: Empowering vision-language models with global-to-ego memory for long-horizon embodied navigation. *arXiv preprint arXiv:2502.14254*, 2025.

A Experimental Details

A.1 Parameters for EmbodiedBench-Navigation Environment

The parameters for the EmbodiedBench-Navigation environment are set as follows, which are aligned with the original parameters used in the original EmbodiedBench paper Yang et al. (2025). These parameters are used in both training and testing environments.

- **Visible Distance:** 10m — The maximum visible distance for objects perceived by the agent.
- **Image Width:** 500px — The pixel width of the single-view image.
- **Image Height:** 500px — The pixel height of the single-view image.
- **Field of View:** 100° — The field of view width visible from the agent's first-person perspective.
- **Move Magnitude:** 0.25m — The distance the agent moves one step forward, backward, left, or right.
- **Rotate Degree:** 90° — The number of degrees the agent turns its view left or right each time.
- **Tilt Degree:** 30° — The number of degrees the agent turns its view up or down each time.
- **Min Distance:** 2.5m — The minimum distance between the agent's starting position and the target.
- **Max Distance:** 3m — The maximum distance between the agent's starting position and the target.
- **Success Threshold:** 1m — The distance threshold for determining whether the agent has successfully reached the vicinity of the target.
- **Action Space:** eight low-level actions with IDs: Move 0-forward / 1-backward / 2-right / 3-left by Δx meters; Rotate to the 4-right / 5-left by $\Delta\theta$ degrees; Tilt the camera 6-upward / 7-downward by $\Delta\phi$ degrees. Each task is defined by a unique initial robot pose, target object, and language instruction.

A.2 Parameters for VLN-CE Environment

The action space and parameters for the VLN-CE environment are set as follows. These parameters are used in both training and testing environments.

- **Action Space:** Stop(0); Move forward(1); Turn left(2); Turn right(3).
- **Dataset Split:** 10,819 training samples and 1,839 val-unseen evaluation samples; scenes are based on real indoor environment scans.
- **Image Width:** 500px — The pixel width of the single-view image.
- **Image Height:** 500px — The pixel height of the single-view image.
- **Field of View:** 90° — The field of view width visible from the agent's first-person perspective.
- **Move Magnitude:** 0.5m — The distance the agent moves one step forward, backward, left, or right.
- **Rotate Degree:** 30° — The number of degrees the agent turns its view left or right each time.
- **Success Threshold:** 3m — The distance threshold for determining whether the agent has successfully reached the vicinity of the target.

A.3 Training Protocol and Compute

For in-distribution (i.d.) training on Embench-Nav, we randomize the 60 base scenes by randomly placing movable objects, randomly setting the agent's initial position and orientation, and randomly selecting navigation goals. For out-of-distribution (o.o.d.) training, we select 60 new scenes from the AI2-THOR scene library that are not used in Embench-Nav, and apply the same randomization. Dual-view inputs are constructed by concatenating FV (left) and BEV (right) into a single image. All experiments were run on 8 H20 GPUs (96 GB VRAM each); a single RFT run took about 48/96 hours on Embench-Nav/VLN-CE.

A.4 Example of Language Prompts on Embench-Nav

The language prompts used for the SeeNav-Agent with visual prompts for Embench-Nav in this work is shown in Table 5. In the prompt, <image>is a placeholder for the input image, {user_instruction} is the navigation command given by human, like "Navigate to the Bread in the room and be as close as possible to it", and {history} represents the interaction history between the agent and the environment, which is a list of natural language. An example of the interaction history for Embench-Nav is shown in Table 4. The maximum time window for the interaction history is set to 5 in this work, which means that the {history} part is limited to recording interaction information for the past five steps at most. When VP or certain modules of VP are not used in the experiments, we remove the prompts related to the corresponding modules from the original prompt.

Table 4: **An Example of the Interaction History.**

The action history:

<action info>
Step 0, action id 0, Move forward by 0.25.
<env feedback>
Last action MoveAhead executed successfully.

<action info>
Step 1, action id 0, Move forward by 0.25.
<env feedback>
Last action MoveAhead executed successfully.

<action info>
Step 2, action id 0, Move forward by 0.25.
<env feedback>
Last action MoveAhead executed successfully.

<action info>
Step 3, action id 2, Move rightward by 0.25.
<env feedback>
Last action MoveRight is invalid. Cube.668 is blocking Agent 0 from moving by (0.0000, 0.0000, 0.2500).

<action info>
Step 4, action id 3, Move leftward by 0.25.
<env feedback>
Last action MoveLeft is invalid. Cube.690 is blocking Agent 0 from moving by (0.0000, 0.0000, -0.2500).

Table 5: **Language Prompt Template for SeeNav-Agent with Visual Prompts.**

<image> ## You are a robot operating in a home. You can do various tasks and output a sequence of actions to accomplish a given task with images of your status.
Your input is a concatenation of your top-down view and the first-person view image, with the top-down image on the left and the first-person view on the right.
The colored circle in the top-down view represents your current position, with the YELLOW side indicating your LEFT and the PURPLE side indicating your RIGHT. The GREEN arrow shows your current camera orientation. In the first person view and the overhead view, the red bounding box in both views highlights the object you need to navigate to. The candidate actions are shown in the overhead view and the first person view images, and each action is represented by a blue arrow, with the corresponding action ID at the end of the arrow.

Now the human instruction is: {user_instruction}.

The available action id (0 to 7) and action names are: action id 0: Move forward by 0.25,
action id 1: Move backward by 0.25,
action id 2: Move rightward by 0.25,
action id 3: Move leftward by 0.25,
action id 4: Rotate to the right by 90 degrees,
action id 5: Rotate to the left by 90 degrees,
action id 6: Tilt the camera upward by 30 degrees,

action id 7: Tilt the camera downward by 30 degrees.

Among these actions, actions 0-3 will only be annotated in the top-down view, while actions 4-7 will only be annotated in the first-person view.

{history}

Now your task is to determine which arrow's action will help you reach the navigation target with a red bounding box (if you encounter any obstacles, prioritize bypassing the obstacles currently blocking your way), and choose the optimal action id.

*** Strategy ***
To achieve the task, 1. Reason about the current visual state and your final goal, and 2. Reflect on the effect of previous actions. 3. Summarize how you learn from the Strategy. Aim for one action per step. At last, output the action id from the available actions to execute.
1. The action arrows on the images indicate the forward direction of the agent after performing the corresponding actions (actions 0-3), or the direction of the agent's view rotation (actions 4-7). You can use this information to help decide which action to take.
2. Strategy2: When determining the relative left-right position between the target and the agent, do not simply look at whether the target is on the left or right side of the top-down view. Instead, you need to take the agent's orientation into account (for example, when the agent is facing downward in the image, objects that appear to be on the left side in the image are actually on the agent's right side).
3. Strategy3: The red bounding box marks your navigation target. Please pay special attention to whether there is a corresponding red bounding box and a red nevigation arrow in your FIRST-PERSON VIEW. Avoid mistakenly judging an existing box as absent, or assuming a non-existent box is present.
4. Strategy4: There is a red navigation arrow in the top-down view and the first-person view that point from the agent to the target, you can use this arrow to assist the current navigation task. If the red navigation arrow is not visible in the first-person view, it means that the target is not visible in the first-person view.
5. Strategy5: When you choose an action, please pay attention to the relationship between the action arrows in the top-down view and the red navigation arrow. Your movement direction should help shorten the red navigation arrow and align the green arrow with the red arrow.
6. Strategy6: If an invalid action has occurred in the action history, please do not perform this action again unless you have already performed a rotation operation.
7. Strategy7: Based on the information in the images, determine which arrow's action will best help you reach the navigation target with a red bounding box (if you encounter any obstacles, prioritize bypassing the obstacles currently blocking your way), and choose the optimal action.
8. Strategy8: You should use rotation (action 4-5) or camera tilt (action 6-7) sparingly, only when you lose track of the target object and it's NOT IN YOUR VIEW. If so, plan nothing but ONE ROTATION OR TILT at a step until that object appears in your view. After the target object appears, start navigation and avoid using rotation until you lose sight of the target again.
You are supposed to output in JSON.

The output json format should be {'visual_state_description': str, 'reasoning_and_reflection': str, "language_plan': str, 'executable_plan': List[{'action_id': int, 'action_name': str}]}
The fields in above JSON follows the purpose below:
1. visual_state_description is for description of current state of the first-person view and the top-down view image.
2. reasoning_and_reflection is for summarizing the history of interactions and any available environmental feedback. Additionally, provide reasoning as to why the last action or plan failed and did not finish the task, and providing reasoning process of why you choose the current action.
3. language_plan is for describing the best action you choose from all the action arrows, which is started by the step number and the action name.
4. executable_plan is the best action you choose that having an action ID and a name.
5. keep your plan efficient and concise.