2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

ReThinkNav: Zero-Shot Vision-and-Language Navigation with Open-Source LLMs via Contextual Reasoning and Loop Recovery

Aolin Li[1], Yixian Yan[2], Hongkun Luo[2], Jiao Zhan[3], and Chi Guo[1,*]

***Abstract*— Zero-shot Vision-and-Language Navigation in Continuous Environments (VLN-CE) requires agents to follow natural language instructions and navigate without task-specific training. Prior works have demonstrated the potential of open-source large language models (LLMs) in zero-shot VLN-CE, yet two major limitations remain: (1) difficulty in accurately following instructions, and (2) susceptibility to loops in spatially confined or semantically similar regions. In this work, we introduce ReThinkNav, a framework designed to further advance open-source LLMs in zero-shot VLN-CE. ReThinkNav integrates contextual reasoning for enhanced instruction comprehension and progress estimation, enabling the LLM to accurately infer both the appropriate action and its rationale. In addition, a Loop Detection and Recovery (LDR) module detects loops and adjusts decisions accordingly. Experiments on the R2R-CE benchmark demonstrate excellent zero-shot performance, while real-world validation on the Unitree G1 humanoid robot confirms its practical applicability. The code is available at https://github.com/damonds27/ReThinkNav.**

I. INTRODUCTION

Vision-and-Language Navigation (VLN) requires agents to interpret natural language instructions and navigate within complex 3D environments [1]. This capability is crucial for household and industrial robots, enabling them to understand human instructions and accomplish daily tasks. Early VLN research primarily focuses on discrete environments and simplifies navigation to traversals over predefined viewpoint graphs [2]. This setting is simple and yields promising results but neglects continuous control and realistic dynamics. To bridge this gap, Vision-and-Language Navigation in Continuous Environments (VLN-CE) is proposed [3], enabling agents to perform fine-grained actions for real-world deployment with greater flexibility and realism.

Zero-shot VLN-CE accomplishes navigation tasks without task-specific training [4], [5], [6]. Existing works typically employ large language models (LLMs) as navigators to interpret instructions [7], [8]. For example, SmartWay [7] employs GPT-4o as the navigator with a backtracking-based strategy, demonstrating outstanding performance. However, it depends on closed-source LLMs, incurring token costs and data privacy risks. In contrast, Open-Nav [8] explores open-source LLMs for zero-shot VLN-CE, reducing costs and privacy risks. Meanwhile, it introduces a spatial-temporal chain-of-thought (CoT) framework [9], structuring navigation reasoning into three sequential stages: instruction comprehension, progress estimation, and decision-making. This design enables a modular and explainable decision-making process for improved navigation.

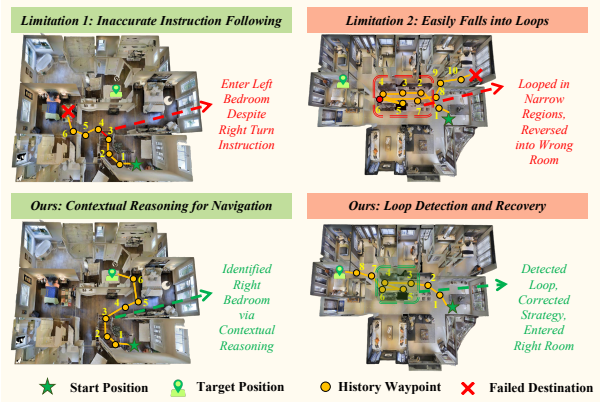

Fig. 1. Limitations of the existing method and the effectiveness of our method. Existing methods struggle with accurate instruction following. We introduce contextual reasoning for enhanced instruction comprehension and progress estimation. Moreover, agents easily fall into loops, causing navigation failure. Our LDR module identifies revisit behaviors and prompts the LLM to reflect on previous reasoning and select alternative actions.

Despite the favorable performance, Open-Nav still faces two key limitations: (1) **Inaccurate instruction following.** By treating actions in isolation without capturing their intent nor completion conditions, Open-Nav easily misinterprets instructions and executes them incorrectly, leading to task failure. In addition, its progress estimation relies solely on navigation history, which records the agent's reasoning and decisions at each step. Wrong decisions within the history severely interfere with progress estimation, leading to inaccurate subsequent actions. As illustrated on the left side of Figure 1, due to incorrect instruction comprehension and inaccurate progress estimation, the agent mistakenly enters the left room, resulting in task failure. (2) **Easily falls into loops.** In spatially confined or semantically similar regions, Open-Nav lacks escape mechanisms, causing agents to easily fail tasks due to getting stuck in loops.

To address these limitations, we present ReThinkNav, a novel framework to further explore open-source LLMs in zero-shot VLN-CE. It builds upon a modular meta-architecture [8] and consists of a waypoint predictor, a scene perception module, and an LLM navigator. We make the

[1]Aolin Li and Chi Guo are with the School of Robotics, Wuhan University, Wuhan 430072, China (e-mail: {liaolin, guochi}@whu.edu.cn).
[2]Yixian Yan and Hongkun Luo are with the School of Geodesy and Geomatics, Wuhan University, Wuhan 430072, China (e-mail: {yanyixian, luohongkun}@whu.edu.cn).
[3]Jiao Zhan is with the Hubei Luojia Laboratory, Wuhan 430072, China (e-mail: zhanjiao1994@whu.edu.cn).
This work was supported by the Major Science and Technology Project of Hubei Province (No. 2022AAA009), the Key Research and Development Program of Wuhan, and the Device Upgrade Project of the Digital Intelligence Education and Teaching Platform of Wuhan University.
*Corresponding author: Chi Guo.

following improvements for better navigation performance: First, we introduce a contextual reasoning mechanism for enhanced instruction comprehension and progress estimation. Instead of treating actions as isolated steps, our instruction comprehension leverages the correlations between actions within instructions for goal-oriented decomposition, thus facilitating better decision-making. Furthermore, beyond navigation history, the progress estimation integrates current scene semantics and motion information for precise next-action estimation. Second, we design an effective Loop Detection and Recovery (LDR) module for better loop escape. It identifies revisit behaviors and prompts the LLM to reflect on previous reasoning and select alternative actions, thus resolving loop issues. Consequently, with the dual effects of the contextual reasoning mechanism and LDR module, our ReThinkNav enables excellent zero-shot navigation.

We conduct extensive experiments on the R2R-CE benchmark and achieve state-of-the-art (SOTA) performance compared to previous zero-shot VLN-CE methods. With Qwen3-32B as the navigator, it achieves a success rate (SR) of 40% and an SPL of 23.43%. With GPT-4o as the navigator, it achieves a success rate (SR) of 39% and an SPL of 26.08%. In addition, we deploy ReThinkNav on the Unitree G1 humanoid robot, validating its effectiveness in real-world environments. Ablation studies also demonstrate the effectiveness of the proposed improvements.

In summary, our main contributions are:

• We propose a novel framework, ReThinkNav, to further explore open-source LLMs in zero-shot VLN-CE.

• We propose a contextual reasoning mechanism for improved instruction comprehension and progress estimation.

• We propose an LDR module to identify and mitigate loop issues.

• We validate ReThinkNav in both simulated and real-world environments, achieving SOTA performance compared to prior zero-shot VLN-CE methods.

II. RELATED WORK

A. Vision-and-Language Navigation

Vision-and-Language Navigation (VLN) tasks require agents to follow natural language instructions to reach target locations. Early VLN research primarily focuses on discrete environments, such as the Matterport3D (MP3D) simulator [10], where agents navigate through predefined graphs representing viewpoints. This setup emphasizes high-level decisions, such as selecting the next viewpoint, while ignoring continuous control and realistic navigation dynamics. In contrast, recent works have shifted to continuous environments using simulators like Habitat [11], enabling fine-grained actions such as moving forward and rotating, better approximating real-world navigation [3]. On the algorithmic side, initial VLN methods typically employ encoder-decoder frameworks based on LSTM [12] and attention mechanisms. Subsequent efforts explore various strategies to improve performance, including enhanced representation learning [13], [14], reinforcement learning [15], data augmentation [16], [17], and external knowledge incorporation [18]. With the rise of Transformer architectures, specialized pre-training strategies tailored for VLN tasks have been proposed, along with methods that leverage topological and semantic mapping to capture global spatial context [19], [20]. Additionally, some recent works fine-tune LLMs, such as NAVILA [21] and NaVid-4D [22], to better suit VLN by leveraging massive navigation data. Unlike these methods, we address the task of VLN-CE using open-source LLMs as navigators in a zero-shot setting.

B. Large Language Models as VLN Navigators

Recent advances in large language models (LLMs) have motivated extensive research on their use as navigators for VLN tasks. In these methods, LLMs are directly responsible for interpreting instructions and selecting actions at each step. These methods generally fall into two categories: fine-tuning and zero-shot. The first relies on fine-tuning LLMs into VLN expert models. NaviLLM [23] is the first to fine-tune LLMs for embodied navigation, unifying diverse tasks into a generative framework. Subsequent works have also fine-tuned LLMs for navigation, achieving strong task performance but at the cost of substantial computational resources [21], [22], [24], [25], [26], [27].

The second category explores zero-shot LLMs for navigation without task-specific training. Early efforts focus on discrete environments. For instance, NavGPT [28] leverages GPT-4 for action selection. MapGPT [29] proposes map-guided prompting, while DiscussNav [30] introduces a multi-stage chain-of-thought framework combining instruction comprehension, progress estimation, and decision-making. More recently, zero-shot methods have extended to continuous environments. SmartWay [7] enhances robustness with a backtracking mechanism while still depending on closed-source LLMs. Open-Nav [8] explores open-source LLMs for zero-shot VLN-CE and proposes a spatial-temporal chain-of-thought (CoT) mechanism to guide their reasoning. However, Open-Nav still struggles with long-horizon reasoning and loops in spatially confined or semantically similar regions. In this work, we present ReThinkNav, a zero-shot VLN-CE framework that improves instruction following via contextual reasoning and enables loop-aware recovery in complex environments. Unlike SmartWay, whose backtracking relies on LLM semantic reasoning, our Loop Detection and Recovery (LDR) module detects loops via geometric pose consistency, enabling reliable recovery in semantically similar regions where semantic reasoning is prone to misjudgment.

III. PRELIMINARIES

In this work, we focus on Vision-and-Language Navigation in Continuous Environments (VLN-CE), where an agent must navigate in a continuous 3D environment $\mathcal{E}$ guided by natural language instructions. At time step t, the agent's position is denoted as $\mathbf{x}_t = (x_t, y_t, z_t) \in \mathcal{E}$. At each location, the agent obtains panoramic RGB-D images, divided into 12 evenly spaced directions $(0°, 30°, \ldots, 330°)$: $\mathcal{I} = \{(I_i^{\text{rgb}}, I_i^{\text{depth}}) \mid i = 1, \ldots, 12\}$, where $I_i^{\text{rgb}} \in \mathbb{R}^{H \times W \times 3}$ and $I_i^{\text{depth}} \in \mathbb{R}^{H \times W}$. The agent is provided with a natural

language instruction $\mathcal{L} = \{l_1, l_2, \ldots, l_n\}$, where each l_i is a token of the instruction. The objective is to navigate from a starting position $\mathbf{x}_{\text{start}} \in \mathcal{E}$ to a goal position $\mathbf{x}_{\text{goal}} \in \mathcal{E}$ by executing discrete low-level actions, aiming to minimize navigation errors and reach the target successfully.

IV. METHOD

The overall framework is illustrated in Figure 2. The Waypoint Prediction module first identifies candidate waypoints from panoramic RGB and depth observations. Next, the Scene Perception module analyzes candidate waypoints to detect objects and capture their spatial relationships, which are then transformed into textual descriptions. Given the instruction and the textualized descriptions, the LLM Navigator performs contextual reasoning: it leverages correlations between actions for instruction comprehension and integrates current scene semantics with motion information for progress estimation. Before making a decision, the Loop Detection and Recovery module identifies revisit behaviors and prompts the LLM to reflect on previous reasoning and select alternative actions, thus resolving loop issues.

A. Waypoint Prediction

Following prior works [7], [8], we leverage a pretrained waypoint predictor to facilitate the adaptation of LLM-based navigators to VLN-CE tasks. The waypoint predictor employs a transformer-based architecture to fuse RGB and depth features for navigation waypoint prediction. Specifically, two separate ResNet50 [31] networks extract features from the RGB and depth inputs, respectively. At each candidate viewpoint i, the extracted feature vectors are concatenated and projected through a linear layer to obtain a fused representation:

$$\mathbf{v}_i^{\text{rgbd}} = W_m\big(f_{\text{ResNet-RGB}}(I_i^{\text{rgb}}) \,\|\, f_{\text{ResNet-Depth}}(I_i^{\text{depth}})\big), \quad (1)$$

where $\|$ denotes concatenation and $W_m \in \mathbb{R}^{d\times 2d}$ is a learnable weight matrix for linear projection. These fused features $\mathbf{v}_i^{\text{rgbd}}$ are then passed through a two-layer transformer network that models spatial relationships among adjacent views. The self-attention mechanism is restricted to the neighboring viewpoints $\{i-1, i, i+1\}$ to enhance local spatial reasoning:

$$\tilde{\mathbf{v}}_i^{\text{rgbd}} = \text{Transformer}\big(\{\mathbf{v}_{i-1}^{\text{rgbd}}, \mathbf{v}_i^{\text{rgbd}}, \mathbf{v}_{i+1}^{\text{rgbd}}\}\big). \quad (2)$$

The transformer's output is fed into a multi-layer perceptron (MLP) to generate a heatmap of candidate waypoints. Non-maximum suppression (NMS) is applied to the heatmap to retain the most likely candidate locations:

$$H_{\text{refined}} = \text{NMS}\big(\text{MLP}(\tilde{\mathbf{v}}_i^{\text{rgbd}})\big). \quad (3)$$

The top K nearby waypoints $\Delta W = \{\Delta w_i\}_{i=1}^{K}$ are selected from the refined heatmap, with each waypoint $\Delta w_i = (\theta_i, d_i)$ representing the angle and distance relative to the agent. The number K is not fixed and can vary depending on the local navigable space around the agent.

B. Scene Perception

Textual descriptions of visual observations are essential for LLM-based zero-shot VLN-CE, as LLMs cannot directly process raw visual inputs. Following prior work [8], we employ the vision-language models RAM [32] and SpatialBot [33] for object detection and spatial relationships extraction. Specifically, RAM detects objects in RGB images and outputs object labels:

$$o_i = \text{RAM}(I^{\text{rgb}}), \quad i = 1, 2, \ldots, n, \quad (4)$$

where o_i is the i-th of n detected objects. SpatialBot processes both RGB and depth inputs to extract spatial relationships among objects, producing textual descriptions:

$$D_{\text{spatial}} = \text{SpatialBot}([I^{\text{rgb}}, I^{\text{depth}}]). \quad (5)$$

To further support navigation decisions, we construct direction descriptions S_d for each candidate direction d. Each direction is assigned a unique ID, and the corresponding S_d specifies the spatial movement resulting from choosing that direction. For example, ID 1 corresponds to the sector spanning the front to 30° left, where S_1 indicates a slight front-left movement, suitable for gently veering left while moving forward.

C. Contextual Reasoning for LLM Navigator

We design a contextual reasoning mechanism for enhanced instruction comprehension and progress estimation, enabling the LLM to accurately infer what actions to take and why.

Instruction Comprehension: To ensure accurate instruction following, the LLM is prompted to decompose instructions into goal-directed actions, each explicitly annotated with its intent and stopping condition. For example, consider the instruction *"Turn right at the top of the stairs, then enter the bedroom."* A naive split into *"turn right"* and *"enter bedroom"* may obscure the purpose of the turn. With a goal-oriented prompt, the LLM instead produces *"Turn right at the top of the stairs to face the bedroom. Move forward until you enter the bedroom."* Such decomposition preserves the correlations between actions, mitigating the limitations of treating actions in isolation and enabling more coherent navigation. Building upon this decomposition, the LLM also extracts landmarks, expressed through clear referential descriptions that specify both the entity and its spatial role in navigation. For example, *"the bedroom on the right (must enter this bedroom to proceed with the instruction)"* marks a critical landmark required for instruction completion.

Progress Estimation: To accurately estimate navigation progress, the LLM is provided navigation history, decomposed actions, landmarks, current observations, motion information, and the previous step's estimation. The current observations $\mathcal{O}_{\text{text}} = (D_{\text{spatial}}, \{o_i\}, S_d)$ are obtained from the Scene Perception module. The motion information is derived from candidate waypoint $\Delta w_i = (\theta_i, d_i)$. Leveraging these inputs, the LLM estimates navigation progress according to the following components:

- **Goal-Oriented Action Verification:** Integrate navigation history, current observations, and motion informa-

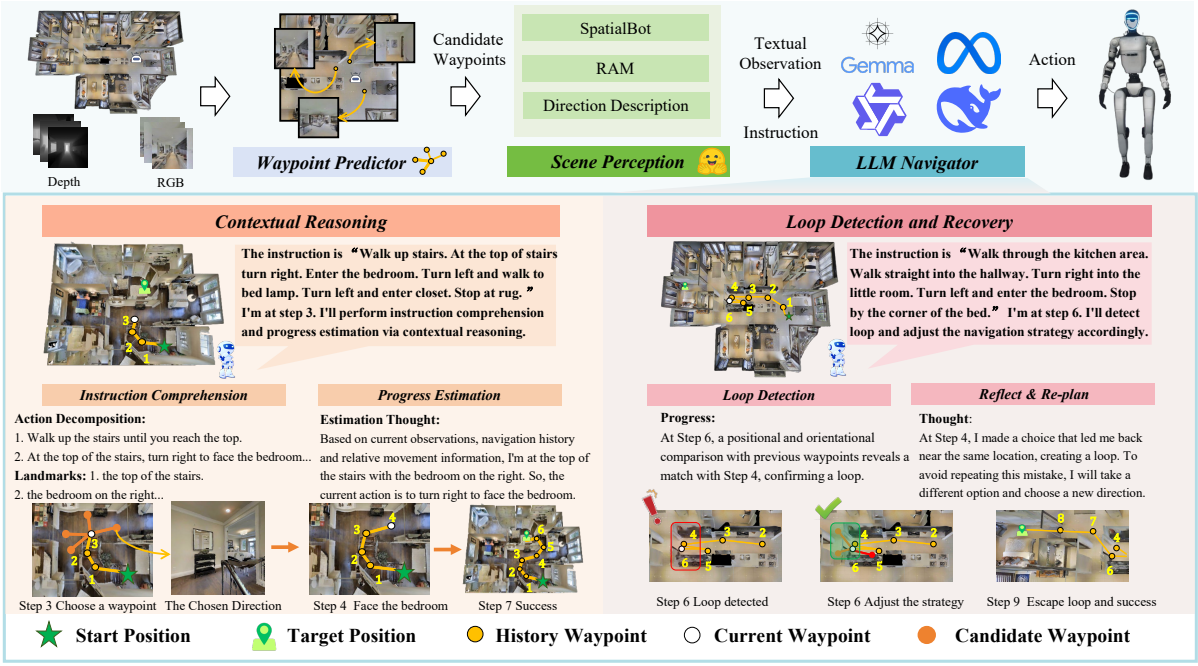

Fig. 2. Overview of the framework. The Waypoint Prediction module identifies candidate waypoints from panoramic RGB-D input. The Scene Perception module converts detected objects and spatial relationships into textual descriptions. Given the instruction and these descriptions, the LLM Navigator performs contextual reasoning. The Loop Detection and Recovery module identifies revisit behaviors and prompts the LLM to reflect on previous reasoning and select alternative actions.

tion to determine which actions have been successfully completed. An action is marked complete only when its intended goal is achieved (e.g.,"enter the bedroom" requires the agent to be visually inside the bedroom).

- **Temporal and Step-wise Tracking:** Maintain strict temporal ordering, ensuring that later actions are considered complete only after prior ones are verified. For multi-step actions, such as ascending a staircase, progress is monitored incrementally using current observations and motion information.
- **Dynamic Re-Evaluation:** Previously completed actions are rechecked from the current observations to ensure their completion status remains accurate.
- **Global Completion Estimation:** Evaluate whether all actions have been completed to determine instruction termination.

D. Loop Detection and Recovery Module

The LDR module first performs loop detection before decision-making. If no loop is detected, the LLM selects candidate waypoints based on their semantic alignment with the instruction and spatial feasibility. If a loop is detected, the system retrieves the corresponding navigation history, prompting the LLM to reflect on prior reasoning and select alternative actions.

Loop Detection. Candidate waypoints are first predicted by the waypoint predictor. Each waypoint $\Delta w_i = (\theta_i, d_i)$ provides the relative angle θ_i and distance d_i from the agent, which are used to record the navigation trajectory and subsequently detect loops. A loop is flagged when the agent's current pose (x_t, θ_t) satisfies

$$\exists\, k < t \colon \|x_t - x_k\|_2 < d_{\text{th}} \ \wedge\ |\theta_t - \theta_k| < \theta_{\text{th}} \qquad (6)$$

where d_{th} and θ_{th} denote the positional and orientation thresholds, respectively. We set $d_{\text{th}} = 0.25$ m and $\theta_{\text{th}} = 30°$. These thresholds correspond to the agent's *minimum forward step* and the *panorama angular interval*. We conduct ablation experiments to evaluate the navigation performance of the LDR module under different threshold settings. The results indicate that $d_{\text{th}} = 0.25$ m and $\theta_{\text{th}} = 30°$ yield the best performance, supporting our choice of these thresholds, which are then fixed for the experiments.

Decision Making under Non-Loop Condition. When no loop is detected, the LLM makes navigation decisions by selecting candidate waypoints based on their semantic alignment with the instruction and spatial feasibility. The LLM considers the candidate waypoints, the instruction with decomposed actions and landmarks, navigation history, previous step estimation, and current observations. For each candidate waypoint, the LLM assigns a confidence score reflecting its consistency with semantic and spatial cues. If a loop is later detected, these scores help the LLM recognize which action caused it and choose alternatives.

Decision Making under Loop Condition. When a loop is detected, the module first identifies the corresponding historical step and retrieves the associated image and LLM reasoning from the navigation history. Next, the module obtains the historical image feature $\mathbf{f}_h$ and each candidate image feature $\mathbf{f}_c^j$ using Contrastive Language-Image Pre-training (CLIP) [34]. The semantic similarity between the

historical and candidate image features is then computed as:

$$s_j = \frac{\mathbf{f}_h \cdot \mathbf{f}_c^j}{\|\mathbf{f}_h\| \, \|\mathbf{f}_c^j\|}, \quad j = 1, 2, \ldots, M. \tag{7}$$

The candidate with the highest similarity is considered the one that would most likely reproduce the loop. Based on this analysis, the LLM reflects on prior reasoning and selects alternative actions, thereby resolving loop issues.

V. EXPERIMENTS

A. Experiment Setup

Simulated Environments. We conduct our experiments on the val-unseen split of R2R-CE [35] using the Habitat simulator [11]. Following the practice of prior works [7], [8], all methods are evaluated on the same 100 episodes. This ensures a fair comparison while maintaining representative coverage of the environment.

Real-world Environments. As shown in Figure 3, we evaluate navigation performance across four real-world environments with diverse layouts: meeting room, office, laboratory, and court. For each environment, we annotate 20 instructions of varying complexity, ranging from simple goal-directed movements to multi-step navigation. This setup enables us to systematically assess ReThinkNav's adaptability in real-world environments.

Lab Meeting Room Office Court

Fig. 3. The layout of the real-world environments

Implementation Details. For simulated experiments, we use the waypoint predictor in [35]. In real-world experiments, we employ a Unitree G1 humanoid robot equipped with a Realsense D435i camera. The robot perceives the environment solely from camera observations, without pre-built maps. Robot pose for loop detection is obtained from the Unitree G1's built-in odometry. Since the G1 humanoid robot cannot capture panoramic images, we omit the waypoint predictor and instead let the LLM select one action per step from the discrete set {*move forward 0.25 m*, *turn left 30°*, *turn right 30°*, *Done*}. For the LLM navigator, we deploy open-source LLMs using the Ollama framework, including Qwen3-32B, Qwen3-14B, Qwen3-8B, Gemma3-27B, Phi3-14B, and Phi4-14B. Furthermore, we evaluate GPT-4o and Qwen3-Max via API, analyzing their navigation performance. All simulated and real-world experiments are conducted on an NVIDIA RTX 4090 GPU (48 GB VRAM, workstation configuration). Prompt templates are provided in the open-source code.

Evaluation Metrics. We evaluate navigation performance using standard VLN metrics: Navigation Error (NE), the distance between the agent's final position and the goal; Success Rate (SR), the percentage of episodes with NE below 3 meters; Oracle Success Rate (OSR), defined as SR under an oracle stop policy; Success weighted by Path Length (SPL), which normalizes SR by trajectory length to reflect efficiency; and Trajectory Length (TL). An episode is considered successful if the agent stops within 3 meters of the goal in the VLN-CE environment. For fair comparison, real-world evaluations use a 2-meter success threshold, consistent with prior works [7], [8].

B. Comparison on Simulated Environment

TABLE I

COMPARISON ON SIMULATED ENVIRONMENT R2R-CE DATASET.

Method	TL	NE ↓	OSR ↑	SR ↑	SPL ↑
Supervised Learning					
CMA [35]	11.08	6.92	45	37	32.17
RecBERT [35]	11.06	5.8	57	48	43.22
BEVBert [36]	13.63	5.13	64	60	53.41
ETPNav [20]	11.08	5.15	58	52	52.18
No Training					
Random	8.15	8.63	12	2	1.50
LXMERT [35]	15.79	10.48	22	2	1.87
MapGPT-CE-GPT4o [29]	12.63	8.16	21	7	5.04
DiscussNav-GPT4 [30]	6.27	7.77	15	11	10.51
Open-Nav-Llama3.1 [8]	8.07	7.25	23	16	12.90
Open-Nav-GPT4 [8]	7.68	6.70	23	19	16.10
SmartWay-GPT4o [7]	13.09	7.01	**51**	29	22.46
ReThinkNav-Qwen3 (Ours)	16.80	7.28	49	**40**	23.43
ReThinkNav-GPT4o (Ours)	14.20	**6.56**	47	39	**26.08**

Quantitative Evaluation. Table I compares our proposed ReThinkNav with several advanced methods on the R2R-CE dataset. To ensure a fair comparison, all methods are evaluated with the waypoint predictor pretrained in [35], except SmartWay, which employs its own trained predictor [7]. The compared methods are divided into two categories. The first group includes supervised methods, such as CMA [35], RecBERT [35], BEVBert [36], and ETPNav [20], which rely on task-specific training. The second group consists of zero-shot methods, including a random waypoint selector (Random), LXMERT [35], MapGPT [29], DiscussNav [30], Open-Nav [8], and SmartWay [7].

With Qwen3-32B as the navigator, our ReThinkNav (i.e., ReThinkNav-Qwen3) achieves a SOTA SR of 40% and the second-highest SPL of 23.43% among zero-shot methods, demonstrating strong overall performance. Its OSR of 49% is also competitive, while the NE of 7.28 meters suggests a slightly larger stopping distance from the target compared with SmartWay. In addition, we evaluate ReThinkNav with GPT-4o-2024-08-06 as the navigator (i.e., ReThinkNav-GPT4o). In this setting, ReThinkNav attains an improved SPL of 26.08% and a reduced NE of 6.56 meters, while maintaining a strong SR of 39% and an OSR of 47%. In conclusion, these impressive quantitative comparisons underscore the superiority of our proposed method.

Qualitative Evaluation. The success cases in Figure 4 demonstrate the reliability of our method. In case (a), the agent successfully follows a complex instruction through contextual reasoning. In case (b), the agent initially falls into a loop, but the LDR mechanism detects it and enables successful completion. The failure cases in Figure 5 reveal remaining challenges. In case (a), the agent starts in a hallway and initially chooses the wrong direction, moving further away from the target, which partly explains why our

overall NE is higher than that of SmartWay. In case (b), the presence of multiple sinks along the path introduces semantic ambiguity, causing the agent to stop prematurely.

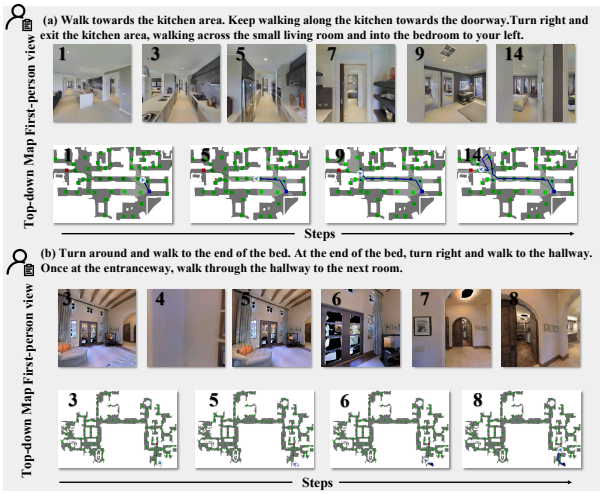

Fig. 4. Success cases of our method in simulation environments

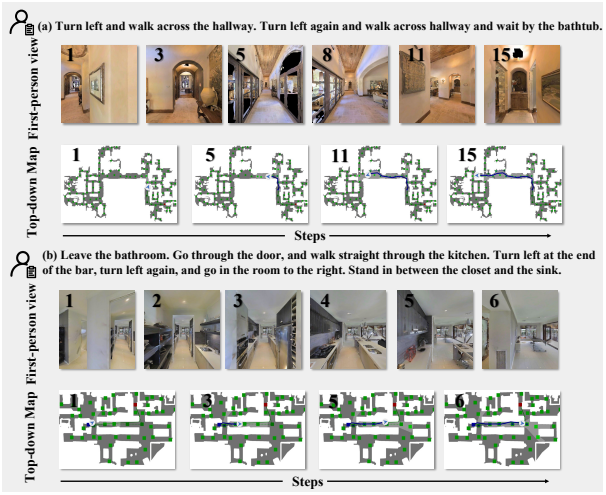

Fig. 5. Failure cases of our method in simulation environments

C. Comparison on Real-world Environment

TABLE II

COMPARISON ON REAL-WORLD ENVIRONMENTS. ALL METHODS USE QWEN3-32B AS THE NAVIGATOR.

Method	Court (Easy)		Meeting Room		Lab		Office (Hard)		All	
	NE↓	SR↑	NE↓	SR↑	NE↓	SR↑	NE↓	SR↑	NE↓	SR↑
Open-Nav	2.42	40	2.53	35	2.65	35	2.94	10	2.64	30
Ours (w/o LDR)	2.33	40	2.40	40	2.48	40	2.88	10	2.52	32.5
Ours (w/ LDR)	2.31	45	2.38	40	2.42	40	2.46	25	**2.39**	**37.5**

For real-world evaluations, we adopt Open-Nav [8] as the baseline. As shown in Table II, ReThinkNav consistently outperforms Open-Nav across all metrics, demonstrating its superior performance. In addition, we conduct an ablation study on the proposed LDR module. The results indicate that incorporating the LDR module reduces navigation error rates and improves success rates, particularly in complex scenes such as the office. As illustrated in Figure 6, in case (a), navigating to the office requires passing through a corridor, where the agent is prone to loops, thereby increasing task difficulty. The supplementary video presents the navigation progress, including representative success and failure cases, from which it can be observed that the system latency remains within an acceptable range.

Fig. 6. Success cases of our method in real-world environments

D. Ablation Study

Contributions of Different Components. We conduct ablation experiments with Qwen3-32B as the navigator to assess the contributions of three components: Direction Descriptions (S_d), Contextual Reasoning (CR), and Loop Detection and Recovery (LDR). Although S_d is a relatively minor adjustment, we include its ablation for completeness. Following prior works [20], [35], [37], we set the maximum navigation steps to 20, instead of Open-Nav's default 6 or 8 determined by the number of decomposed actions. This ensures that the agent has sufficient steps to complete instructions and that the ablation fairly reflects the contribution of each component. In addition, a termination signal is triggered once progress estimation indicates instruction completion.

Table III presents 9 frameworks with different configurations, with Row 1 being the baseline Open-Nav. These results reveal several key insights: (1) Increasing the maximum navigation steps (Row 2) and introducing the *Done* signal improve performance across all metrics. (2) Adding S_d alone (Row 3) enhances SPL, confirming that direction descriptions improve path efficiency. (3) Incorporating CR (Row 4) yields clear gains in SR and OSR, indicating the effectiveness of the improved instruction comprehension and progress estimation. (4) LDR (Row 5) increases SR and reduces NE by detecting and recovering from loops, thereby mitigating repeated errors. (5) Combining any two modules (Rows 6–8) further improves performance, particularly in SR and SPL. (6) Integrating all three modules (Row 9) achieves the best overall results, with the highest SR (40%), OSR (49%), and SPL (23.43%), demonstrating their complementarity.

In conclusion, the above experimental results highlight the effectiveness of our proposed improvements, demonstrating

TABLE III

ABLATION STUDY OF THE PROPOSED MODULES AND SETTINGS ON NAVIGATION PERFORMANCE. SYMBOLS: - = ABSENT/ORIGINAL, ✓= PRESENT/ENHANCED.

#	MaxStep	Done	S_d	CR	LDR	Metrics				
						TL	NE↓	OSR↑	SR↑	SPL↑
1	6/8	-	-	-	-	9.19	7.49	24	20	14.58
2	20	✓	-	-	-	12.53	6.94	31	25	16.86
3	20	✓	✓	-	-	12.95	7.65	33	25	17.60
4	20	✓	-	✓	-	16.08	6.91	37	30	19.42
5	20	✓	-	-	✓	13.90	7.05	36	28	19.05
6	20	✓	✓	-	✓	12.16	7.66	38	28	20.97
7	20	✓	-	✓	✓	17.05	**6.79**	43	33	20.84
8	20	✓	✓	✓	-	16.63	7.64	45	35	21.36
9	20	✓	✓	✓	✓	16.80	7.28	**49**	**40**	**23.43**

their potential to improve navigation performance in zero-shot VLN-CE.

Fine-Grained Ablation on Contextual Reasoning. With Qwen3-32B as the navigator, we further analyze the contributions of the Contextual Reasoning (CR) framework by ablating its two subcomponents: Instruction Comprehension (IC) and Progress Estimation (PE). The baseline (Row 1) employs naive designs for both. As shown in Table IV, instruction comprehension alone (Row 2) improves SPL by +1.04% by reducing misaligned actions, while progress estimation alone (Row 3) yields a larger gain (+2.46% SPL) by mitigating premature termination. The full CR module (Row 4) achieves the best balance, attaining the highest SR (30%) and SPL (19.42%), highlighting the synergy between accurate instruction comprehension and progress estimation.

TABLE IV

ABLATION STUDY ON CONTEXTUAL REASONING COMPONENTS

#	IC	PE	TL	NE↓	OSR↑	SR↑	SPL↑
1	-	-	12.53	6.94	31	25	16.86
2	✓	-	13.59	7.05	35	27	17.90
3	-	✓	17.00	7.21	37	27	19.32
4	✓	✓	16.08	**6.91**	**37**	**30**	**19.42**

Analysis of LDR Module under Different Thresholds. Figure 7 evaluates the LDR module under varying distance and angle thresholds in simulation. Incorporating the LDR module can improve SR and SPL compared to a baseline without loop recovery under appropriate threshold settings. For instance, a distance threshold of 0.25 meters with angle thresholds between 30° and 45° achieves an SR of 40% and an SPL of 23.43%, while a looser distance threshold (0.75 meters) leads to lower efficiency (SPL 18.57%). Overall, these results demonstrate that the LDR module maintains robust performance across a practical range of threshold settings, while overly loose thresholds can degrade efficiency.

Comparison of Different LLMs in Navigation. Table V shows that the Qwen3 series consistently outperforms other models, indicating stronger suitability for navigation reasoning. Within the series, performance does not strictly increase with model size: Qwen3-8B already surpasses Qwen3-14B on some metrics, while Qwen3-32B achieves the best overall

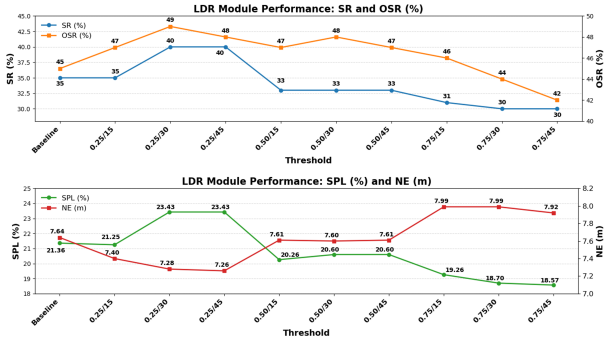

Fig. 7. Analysis of LDR module under different thresholds

balance with the highest SR and SPL. To investigate whether scaling beyond 32B provides further gains, we evaluate Qwen3-Max via API access. Although Qwen3-Max attains the lowest NE of 6.75 meters and the highest OSR of 55%, its SPL falls below that of Qwen3-14B, reflecting over-deliberation and hesitation to terminate upon reaching the goal. These results confirm that ReThinkNav's improvements stem primarily from contextual reasoning and loop recovery rather than model size alone.

TABLE V

PERFORMANCE OF DIFFERENT LLMS ON NAVIGATION

Method	TL	NE↓	OSR↑	SR↑	SPL↑
Gemma3-27B	13.01	8.05	26	20	12.51
Phi3-14B	12.22	8.09	26	21	14.28
Phi4-14B	13.84	6.85	30	25	13.73
Qwen3-8B	15.74	6.96	44	26	14.39
Qwen3-14B	18.69	7.59	37	24	16.79
Qwen3-32B	16.80	7.28	49	**40**	**23.43**
Qwen3-Max	20.67	**6.75**	**55**	35	15.98

VI. CONCLUSIONS

In this paper, we propose a novel framework, ReThinkNav, to further explore open-source LLMs for zero-shot VLN-CE. ReThinkNav integrates contextual reasoning for enhanced instruction comprehension and progress estimation, enabling the LLM to accurately infer both the appropriate action and its rationale. To address loop issues, we propose a Loop Detection and Recovery (LDR) module, which identifies revisit behaviors and prompts the LLM to reflect on previous reasoning and select alternative actions. We conduct extensive experiments on the R2R-CE benchmark and achieve state-of-the-art performance compared to previous zero-shot VLN-CE methods. Real-world deployment on a Unitree G1 humanoid robot further confirms its practical applicability. In future work, we plan to extend our study to the VLN-PE platform [38], which allows evaluation under more realistic robot locomotion, enabling the investigation of failures arising from robot control and navigation dynamics.

REFERENCES

[1] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. 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*, 2018, pp. 3674–3683.

[2] D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L.-P. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, and T. Darrell, "Speaker-follower models for vision-and-language navigation," *Advances in neural information processing systems*, vol. 31, 2018.

[3] J. Krantz, E. Wijmans, A. Majumdar, D. Batra, and S. Lee, "Beyond the nav-graph: Vision-and-language navigation in continuous environments," in *European Conference on Computer Vision*. Springer, 2020, pp. 104–120.

[4] Y. Long, W. Cai, H. Wang, G. Zhan, and H. Dong, "Instructnav: Zero-shot system for generic instruction navigation in unexplored environment," in *Conference on Robot Learning*. PMLR, 2025, pp. 2049–2060.

[5] K. Chen, D. An, Y. Huang, R. Xu, Y. Su, Y. Ling, I. Reid, and L. Wang, "Constraint-aware zero-shot vision-language navigation in continuous environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[6] S. Wu, R. Liu, Z. Xie, and Z. Pang, "Map-semnav: Advancing zero-shot continuous vision-and-language navigation through visual semantics and map integration," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 7577–7584.

[7] X. Shi, Z. Li, W. Lyu, J. Xia, F. Dayoub, Y. Qiao, and Q. Wu, "Smartway: Enhanced waypoint prediction and backtracking for zero-shot vision-and-language navigation," *arXiv preprint arXiv:2503.10069*, 2025.

[8] Y. Qiao, W. Lyu, H. Wang, Z. Wang, Z. Li, Y. Zhang, M. Tan, and Q. Wu, "Open-nav: Exploring zero-shot vision-and-language navigation in continuous environment with open-source llms," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 6710–6717.

[9] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[10] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niebner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," in *2017 International Conference on 3D Vision (3DV)*. IEEE Computer Society, 2017, pp. 667–676.

[11] X. Puig, E. Undersander, A. Szot, M. D. Cote, T.-Y. Yang, R. Partsey, R. Desai, A. W. Clegg, M. Hlavac, S. Y. Min, *et al.*, "Habitat 3.0: A co-habitat for humans, avatars and robots," *arXiv preprint arXiv:2310.13724*, 2023.

[12] A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[13] D. An, Y. Qi, Y. Huang, Q. Wu, L. Wang, and T. Tan, "Neighbor-view enhanced model for vision and language navigation," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 5101–5109.

[14] P. Kong, R. Liu, Z. Xie, and Z. Pang, "Vln-khvr: Knowledge-and-history aware visual representation for continuous vision-and-language navigation," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 5236–5243.

[15] X. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang, and L. Zhang, "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 6629–6638.

[16] J. Li, H. Tan, and M. Bansal, "Envedit: Environment editing for vision-and-language navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 15 407–15 417.

[17] Z. Wang, J. Li, Y. Hong, Y. Wang, Q. Wu, M. Bansal, S. Gould, H. Tan, and Y. Qiao, "Scaling data generation in vision-and-language navigation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 12 009–12 020.

[18] X. Li, Z. Wang, J. Yang, Y. Wang, and S. Jiang, "Kerm: Knowledge enhanced reasoning for vision-and-language navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2583–2592.

[19] S. Chen, P.-L. Guhur, M. Tapaswi, C. Schmid, and I. Laptev, "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*, 2022, pp. 16 537–16 547.

[20] D. An, H. Wang, W. Wang, Z. Wang, Y. Huang, K. He, and L. Wang, "Etpnav: Evolving topological planning for vision-language navigation in continuous environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[21] A.-C. Cheng, Y. Ji, Z. Yang, X. Zou, J. Kautz, E. Biyik, H. Yin, S. Liu, and X. Wang, "Navila: Legged robot vision-language-action model for navigation," in *RSS*, 2025.

[22] H. Liu, W. Wan, X. Yu, M. Li, J. Zhang, B. Zhao, Z. Chen, Z. Wang, Z. Zhang, and H. 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)*, 2025, pp. 10 607–10 615.

[23] D. Zheng, S. Huang, L. Zhao, Y. Zhong, and L. Wang, "Towards learning a generalist model for embodied navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 13 624–13 634.

[24] J. Zhang, K. Wang, S. Wang, M. Li, H. Liu, S. Wei, Z. Wang, Z. Zhang, and H. Wang, "Uni-navid: A video-based vision-language-action model for unifying embodied navigation tasks," *Robotics: Science and Systems*, 2025.

[25] M. Wei, C. Wan, X. Yu, T. Wang, Y. Yang, X. Mao, C. Zhu, W. Cai, H. Wang, Y. Chen, *et al.*, "Streamvln: Streaming vision-and-language navigation via slowfast context modeling," *arXiv preprint arXiv:2507.05240*, 2025.

[26] Z. Yu, Y. Long, Z. Yang, C. Zeng, H. Fan, J. Zhang, and H. Dong, "Correctnav: Self-correction flywheel empowers vision-language-action navigation model," *arXiv preprint arXiv:2508.10416*, 2025.

[27] Z. Qi, Z. Zhang, Y. Yu, J. Wang, and H. Zhao, "Vln-r1: Vision-language navigation via reinforcement fine-tuning," *arXiv preprint arXiv:2506.17221*, 2025.

[28] G. Zhou, Y. Hong, and Q. Wu, "Navgpt: Explicit reasoning in vision-and-language navigation with large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 7, 2024, pp. 7641–7649.

[29] J. Chen, B. Lin, R. Xu, Z. Chai, X. Liang, and K.-Y. Wong, "Mapgpt: Map-guided prompting with adaptive path planning for vision-and-language navigation," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 9796–9810.

[30] Y. Long, X. Li, W. Cai, and H. Dong, "Discuss before moving: Visual language navigation via multi-expert discussions," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 17 380–17 387.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[32] Y. Zhang, X. Huang, J. Ma, Z. Li, Z. Luo, Y. Xie, Y. Qin, T. Luo, Y. Li, S. Liu, *et al.*, "Recognize anything: A strong image tagging model," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 1724–1732.

[33] W. Cai, I. Ponomarenko, J. Yuan, X. Li, W. Yang, H. Dong, and B. Zhao, "Spatialbot: Precise spatial understanding with vision language models," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 9490–9498.

[34] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PmLR, 2021, pp. 8748–8763.

[35] Y. Hong, Z. Wang, Q. Wu, and S. 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*, 2022, pp. 15 439–15 449.

[36] D. An, Y. Qi, Y. Li, Y. Huang, L. Wang, T. Tan, and J. Shao, "Bevbert: Multimodal map pre-training for language-guided navigation," *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[37] Z. Wang, X. Li, J. Yang, Y. Liu, and S. Jiang, "Gridmm: Grid memory map for vision-and-language navigation," in *Proceedings of the IEEE/CVF International conference on computer vision*, 2023, pp. 15 625–15 636.

[38] L. Wang, X. Xia, H. Zhao, H. Wang, T. Wang, Y. Chen, C. Liu, Q. Chen, and J. Pang, "Rethinking the embodied gap in vision-and-language navigation: A holistic study of physical and visual disparities," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.