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

Fast-SmartWay: Panoramic-Free End-to-End Zero-Shot Vision-and-Language Navigation

Xiangyu Shi[1], Zerui Li[1], Yanyuan Qiao[2§], Qi Wu[1†]

***Abstract*— Recent advances in Vision-and-Language Navigation in Continuous Environments (VLN-CE) have leveraged multimodal large language models (MLLMs) to achieve zero-shot navigation. However, existing methods often rely on panoramic observations and two-stage pipelines involving waypoint predictors, which introduce significant latency and limit real-world applicability. In this work, we propose Fast-SmartWay, an end-to-end zero-shot VLN-CE framework that eliminates the need for panoramic views and waypoint predictors. Our approach uses only three frontal RGB-D images combined with natural language instructions, enabling MLLMs to directly predict actions. To enhance decision robustness, we introduce an Uncertainty-Aware Reasoning module that integrates (i) a Disambiguation Module for avoiding local optima, and (ii) a Future-Past Bidirectional Reasoning mechanism for globally coherent planning. Experiments on both simulated and real-robot environments demonstrate that our method significantly reduces per-step latency while achieving competitive or superior performance compared to panoramic-view baselines. These results demonstrate the practicality and effectiveness of Fast-SmartWay for real-world zero-shot embodied navigation.**

I. INTRODUCTION

The Vision-and-Language Navigation (VLN) task [1] aims to enable embodied agents to navigate in environments based on natural language instructions [1], [2]. Early VLN approaches [3], [4], [5] typically relied on pre-defined navigation graphs, which constrained the agent's decision-making within a static environment representation and limited real-world applicability. To address these limitations and improve the deployability of VLN in realistic scenarios, a Vision-and-Language Navigation in Continuous Environment (VLN-CE) benchmark [6], [7] was proposed. Unlike previous tasks, VLN-CE removes the dependence on pre-constructed maps by introducing a waypoint predictor [7] to dynamically identify traversable regions. This mechanism empowers the agent to operate in previously unseen environments without prior knowledge of the layout, making the task more practical and generalizable. Traditional VLN-CE systems [8], [9], [10], [11] are usually trained on large-scale datasets, learning to map language instructions and visual observations into a sequence of navigation actions. While these trained agents may perform well in seen or similar environments, they often struggle to generalize to unfamiliar scenarios due to overfitting to specific visual or structural patterns. This challenge has led to an increasing interest in integrating large language models (LLMs) [12] and multimodal large models (MLLMs) [13] into both VLN and VLN-CE areas. For instance, NavGPT [14] is one of the first attempts to use GPT-4 [13] in zero-shot VLN by converting visual inputs into textual descriptions and feeding them into GPT-4 for reasoning and direction prediction. Open-Nav [15] first utilises open-source LLMs in a zero-shot VLN-CE setting by employing a waypoint predictor [7] to generate candidate navigable locations, converting each candidate's associated visual input into a textual description, and then feeding these descriptions into the LLMs for instruction interpretation and navigation decision making. In contrast, SmartWay [16] leverages GPT-4o's native multimodal capabilities, directly feeding images and text into the model for joint visual-linguistic reasoning.

Although recent zero-shot VLN-CE methods using LLMs

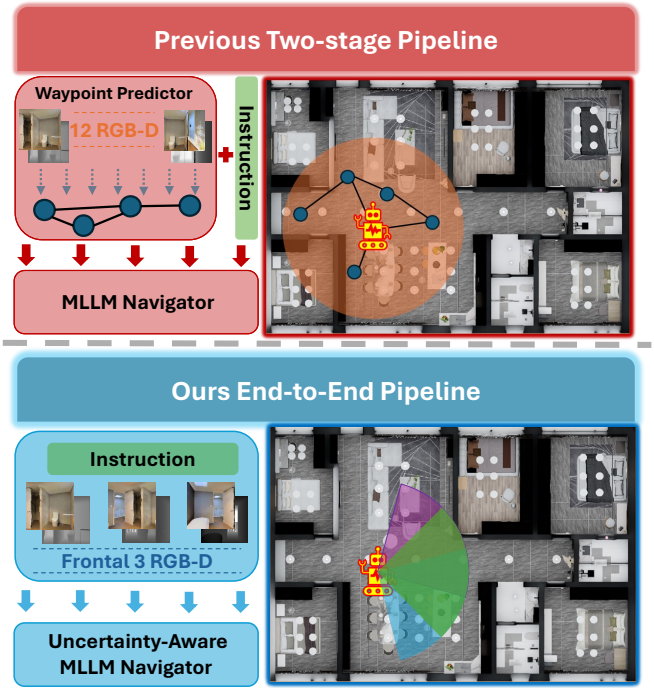

Fig. 1. Comparison between previous VLN-CE pipelines and our proposed end-to-end framework. **Top:** Previous two-stage methods first feed panoramic RGB-D observations (12 views) into a waypoint predictor to generate candidate waypoints. In the second stage, the candidate waypoints, together with the instruction, are provided to the navigator to select the final navigation action. **Bottom:** Our framework integrates the instruction with three frontal-view images in an Uncertainty-Aware Navigator, directly predicting navigation actions in a single end-to-end step.

[1]Xiangyu Shi, Zerui Li, and Qi Wu are with the Australian Institute for Machine Learning, the University of Adelaide. Xiangyu Shi (xiangyu.shi@adelaide.edu.au)
[2]Yanyuan Qiao is with the CREATE Lab, Swiss Federal Institute of Technology Lausanne (EPFL)
§ Project Lead: Yanyuan Qiao (yanyuan.qiao@epfl.ch)
† Corresponding author: Qi Wu (qi.wu01@adelaide.edu.au)
This work utilizes GPT as the reasoning backbone of the proposed method.

and MLLMs have shown promising results, they still face key challenges in real-world deployment. Most of these methods rely on panoramic observations, which require either rotating the robot 360 degrees to capture 12 views or using specialized panoramic cameras. This process is slow, introduces latency in dynamic environments, and is often incompatible with compact robots that only have a front-facing camera. In addition, many existing methods [15], [16] rely on waypoint predictors that select navigable points based only on RGB-D inputs, without considering the semantic alignment between the current environment and the instruction. As a result, the selected candidates may not align with the intended goal, leading to suboptimal decisions. This reliance on discrete candidates also prevents the agent from generating smooth and continuous trajectories, further limiting their practicality in real-world navigation tasks.

To address these limitations, we propose an End-to-End VLN-CE framework named **Fast-SmartWay**, shown in Fig. 1, that significantly reduces the processing time for each action step. Unlike prior methods that rely on panoramic observations, typically requiring 12 images captured by rotating 30 degrees each in traditional VLN-CE settings, we leverage only the front three-view images as visual input. These images, together with the language instruction, are directly fed into a multimodal large model (MLLM), which predicts the turning angle and forward distance in a single forward pass, enabling fully end-to-end navigation.

In addition, we propose a **Spatial-Semantic Textual Description Generation** module that replaces traditional waypoint predictors by transforming spatial layouts and semantic cues into textual prompts. These prompts, combined with the language instruction, are directly fed into the MLLM to predict navigation actions, eliminating the need for explicit candidate selection. This design tightly integrates perception and decision-making, enhancing both efficiency and generalisation.

Furthermore, we introduce an **Uncertainty-Aware Reasoning** module to enhance the robot's robustness and decision quality in challenging environments. It consists of two components: (1) a Disambiguation module, which dynamically evaluates the diversity and uncertainty of path choices to help the robot avoid dead-ends and local optima, and (2) inspired by NavBench's [17] Local Observation ability, we propose a Future-Past Bidirectional Reasoning (FPBR) mechanism, which leverages both historical analysis and future predictions to guide the MLLM in evaluating the rationality of its current decisions, improving the robot's adaptability and success rate in dynamic environments without requiring additional training.

Our main contributions are as follows:

- We present an **End-to-End VLN-CE Framework** that eliminates the need for panoramic observations and waypoint predictors by leveraging only frontal views, enabling more efficient end-to-end navigation.
- We propose an Uncertainty-Aware Reasoning module, which integrates a Disambiguation module for avoiding dead-ends and a FPBR mechanism for enhancing planning consistency and robustness.
- We conduct extensive experiments on standard VLN-CE benchmarks, demonstrating the effectiveness and practicality of our method in zero-shot settings.

II. Related Work

A. Vision-and-Language Navigation

Vision-and-Language Navigation (VLN) tasks require agents to follow instructions to reach target locations in unseen environments [1], [19]. Early benchmarks are built on discrete environments with predefined navigation graphs [20], [21], [22]. To improve realism, Krantz et al. [6] extended VLN to continuous environments (VLN-CE), enabling agents to navigate in more realistic, map-free settings. To support this, subsequent works [7] introduced waypoint predictors to estimate traversable candidates based on the agent's current observation. These predictors have since become a core component in many VLN-CE systems [8], [9]. However, training such systems remains data-intensive and environment-specific, leading to growing interest in zero-shot VLN approaches that leverage the generalization capabilities of large foundation models.

B. Zero-Shot VLN

To reduce reliance on task-specific data and improve generalization to unseen environments, recent studies explore zero-shot VLN approaches by leveraging the powerful reasoning capabilities of large language models (LLMs) and multimodal large models (MLLMs). These methods aim to bypass the need for environment-specific training by prompting pretrained models with visual and linguistic inputs. NavGPT [14] is an early attempt that converts visual observations into textual descriptions and feeds them, along with the instruction, into GPT-4 for navigation reasoning. DiscussNav [23] further adopts a multi-expert framework, assigning the VLN task to GPT4 for execution. OpenNav [15] extends this idea to the VLN-CE setting by combining a learned waypoint predictor with open-source LLMs, which evaluate each candidate location based on its textual description. More recently, SmartWay [16] utilizes GPT-4o's multimodal ability, enabling visual-linguistic reasoning with a backtrack mechanism. Nevertheless, these methods often rely on panoramic inputs and incur significant processing latency, motivating the exploration of frontal-view VLN settings.

C. Ego-Centric VLN

Most existing VLN and VLN-CE methods rely on panoramic observations obtained through 360° rotations or specialized sensors. However, such configurations are often impractical for real-world robots, which typically operate with only a forward-facing camera. Recent efforts, such as Navid [24], have explored ego-centric settings by leveraging sequences of past front-view observations combined with the latest frame for decision-making. Similarly, Wang et al. [25] propose building a semantic traversability map to identify navigable regions and guide agents along feasible paths.

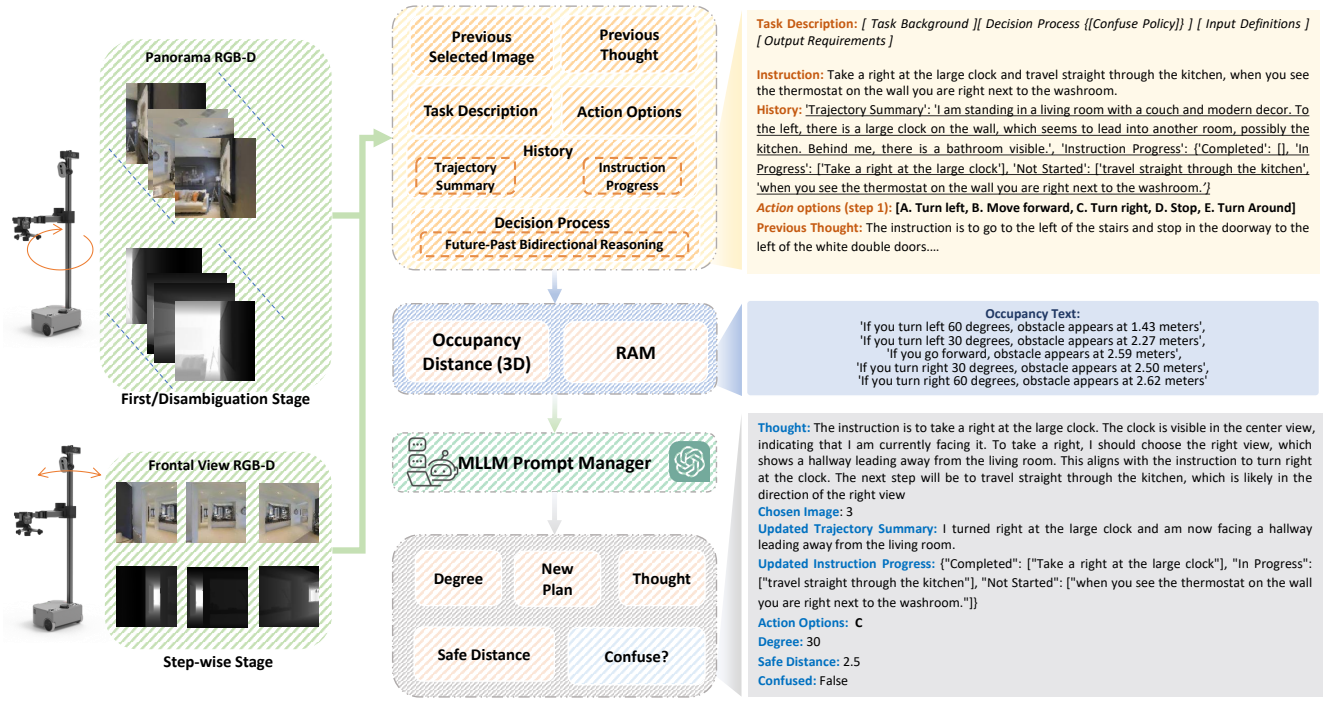

Fig. 2. Overall workflow of our proposed zero-shot VLN-CE framework. The navigation process begins with panoramic RGB-D observations at the initial or disambiguation stage, and uses three frontal RGB-D views during the step-wise stage. The system first constructs structured prompts, then extracts spatial and semantic descriptions, and finally sends both together to the Multimodal Large Language Model (MLLM). Within the Decision Process, the MLLM performs step-wise reasoning and incorporates Future-Past Bidirectional Reasoning (FPBR) to ensure globally consistent planning. It also determines whether the robot is confused based on the instruction and context. If so, the Disambiguation Module is triggered to collect a panoramic observation and replan. The robot used in the figure is a Hello Robot [18] equipped with an Intel RealSense camera mounted at a height of 125 cm.

Nevertheless, ego-centric perception can cause agents to remain on incorrect trajectories without realizing deviations from the instruction, sometimes following a wrong route until the end. To address this challenge, our work employs three forward-facing views and introduces an uncertainty-aware reasoning module, enhancing robustness perception without additional training.

III. Problem Formulation

In this work, we tackle the problem of *Vision-and-Language Navigation in Continuous Environments* (VLN-CE), where an agent needs to navigate through a 3D environment $\mathbf{E}$ following the natural language instruction $L = \{l_1, l_2, \ldots, l_n\}$.

a) Classical VLN-CE Setting.: In prior VLN-CE methods, the agent at each time step t is assumed to have access to a full panoramic observation. This panorama consists of 12 RGB-D images captured at evenly spaced headings (e.g., $0°, 30°, \ldots, 330°$), yielding: $I_t = \{(I_i^{rgb}, I_i^{depth}) \mid i = 1, \ldots, 12\}$, where each $I_i^{rgb} \in \mathbb{R}^{H \times W \times 3}$ and $I_i^{depth} \in \mathbb{R}^{H \times W}$. However, such 360-degree perception sensing is often challenging in real-world scenarios due to hardware limitations and time constraints for multi-view scanning.

b) Our Setting: To better reflect realistic constraints, we propose a more practical setting where the agent only performs one panoramic scan at the initial step $t = 0$, yielding the initial input: $I_0 = \{(I_i^{rgb}, I_i^{depth}) \mid i = 1, \ldots, 12\}$. This scan provides a coarse environment understanding only before navigation begins. For all subsequent steps $t > 0$, the agent receives only frontal observation consisting of 3 RGB-D views captured at heading angles $(330°, 0°, 30°)$, corresponding to the left, front, and right directions: $I_t = \{(I_i^{rgb}, I_i^{depth}) \mid i = 1, 2, 3\}$.

At each step, given the current partial observation I_t and instruction L, the agent selects a low-level action (i.e., movement direction and distance) to progressively follow the instruction and reach the goal location $\mathbf{x}_{\text{goal}}$.

IV. Method

As illustrated in Fig. 2, our method begins with an end-to-end, zero-shot VLN-CE pipeline. Unlike prior zero-shot approaches [15], [16] that rely on waypoint predictors [7] to generate candidate turning angles and movement distances from panoramic views, our method utilizes a Multimodal Large Language Model (MLLM) to directly infer the robot's next action, including both turning angle and forward distance based on the given instruction and the frontal view.

While this end-to-end formulation simplifies the navigation pipeline by eliminating intermediate heuristics, it may still encounter ambiguous or conflicting situations during navigation, especially in complex indoor environments. To enhance the robustness of our MLLM-based inference, we introduce an *Uncertainty-Aware Reasoning* mechanism. This mechanism enables the robot to (i) identify and recover from uncertain cases via a Disambiguation Module, and

(ii) incorporate both past trajectories and anticipated future subgoals to enable globally consistent path planning.

A. End-to-End Pipeline

In prior zero-shot VLN-CE methods [15], [16], the waypoint predictor is adopted as an intermediate module to process a panoramic RGB-D input, which is represented by 12 distinct images captured at 30-degree intervals. The module subsequently generates a discrete set of waypoint candidates, each of which is presented as an image for the navigator to make a selection. However, this two-stage design introduces a structural disconnection between vision and language. The waypoint predictor is typically trained only by RGB-D, without an instruction understanding module. As a result, the generated candidates may be visually plausible but semantically irrelevant to the goal, leading to suboptimal choices.

In contrast, under our frontal view setting, instead of retraining a waypoint predictor to operate on restricted-view observations, we completely remove the waypoint predictor and propose the End-to-End Navigator that asks MLLM to directly generate the next action.

Spatial-Semantic Textual Description Generation: To capture spatial information from the robot's depth observations, we construct a partial panoramic point cloud using three depth images $\{I_i^{depth} \mid i \in \{L,F,R\}\}$, corresponding to views taken at 30° to the left, forward, and 30° to the right of the robot's heading. We extract a centre crop from each image to focus on the most reliable region, thereby reducing peripheral distortion and improving depth quality. To focus on nearby obstacles on the ground, we retain only the bottom half of each depth map during projection. Each cropped depth image is then projected into a 3D space using the pinhole camera model. The resulting local point clouds are finally transformed into the world frame using the robot's pose and the relative yaw of each view.

Let the resolution of each cropped depth image be (h,w), where h denotes the image height and w the width, where $h = w = 256$. After the projection, each depth map yields a set of 3D points arranged in a $(h/2,w)$ grid, corresponding to the bottom half of the image. We then compute the ground-plane (horizontal) distance of each 3D point to the robot:

$$D = \sqrt{x^2 + z^2},$$

where (x,z) are the 2D coordinates of a 3D point in the robot's local frame.

To robustly estimate the distance to the nearest obstacle in each direction, we compute the minimum ground-plane distance within each column:

$$D_j^{(i)} = \min_{k \in \{1,\ldots,h/2\}} \sqrt{x_{k,j}^2 + z_{k,j}^2}.$$

where $i \in \{L,F,R\}$ indexes the view direction, and $j \in \{1,\ldots,w\}$ indexes the image column. This results in three 1D vectors $\{D^{(L)}, D^{(F)}, D^{(R)}\}$, each of length w, encoding the closest ground-level obstacles in the left, front, and right views, respectively.

To make the distances interpretable for the MLLM, we concatenate the $\{D^{(L)}, D^{(R)}\}$, covering a 120° horizontal field of view (hfov) (−60° to +60°), where each individual $D^{(i)}$ contributes a 60° sub-field. This range is discretized into $N = 5$ directional bins for five directions: turning left 60°, turning left 30°, going forward, turning right 30°, and turning right 60°. For each bin $b \in \{1,\ldots,5\}$, we compute the mean distance $\bar{d}_b$ of all 3D points within its angular range and generate a textual description in the form *"If you [direction], ..."*, where the direction is determined by the bin's orientation.

To generate natural language descriptions, we compare each $\bar{d}_b$ against two thresholds, d_{close} and d_{mid}, and select the textual description based on the following rule:

$$\ell_b = \begin{cases} \text{'If you [direction], there is a very close obstacle'}, & \text{if } \bar{d}_b < d_{\text{close}} \\ \text{'If you [direction], obstacle appears at } \bar{d}_b \text{ meters'}, & \text{if } d_{\text{close}} \le \bar{d}_b < d_{\text{mid}} \\ \text{'If you [direction], path is clear for moving forward in } \bar{d}_b \text{ meters'}, & \text{otherwise} \end{cases}$$

We denote the resulting set of spatial descriptions as $\mathscr{L}_{\text{spatial}}^{(5)} = \{\ell_1, \ell_2, \ell_3, \ell_4, \ell_5\}$, where each ℓ_i corresponds to one of the 5 discretized orientations $-60°, -30°, 0°, +30°, +60°$.

The spatial description generation is reused not only during regular navigation steps, but also during two special situations: (i) the first navigation step, where the robot performs a 360° rotation to understand its full surroundings, and (ii) the Disambiguation stage, which is triggered when the robot encounters uncertain or conflicting signals. In both cases, the robot performs a 360° turn and captures 12 depth images (rotate 30° each) with $w = 256$. In this case, we compute distances only from the central half of each image (columns $[64, 192]$) to reduce redundant obstacles. For each view $i \in \{1,\ldots,12\}$, we calculate the mean distance $\bar{d}_i$ of the selected segment and generate a natural language statement based on its global orientation, such as *"go forward"*, *"turn left* 90°*"*, *"turn right* 90°*"*, or *"turn around"*. Directions falling within the left half of the panoramic view are described as "turn left," while those exceeding 180° are normalised and expressed as "turn right" with the equivalent angle. Descriptions again indicate whether the path contains a very close obstacle, a mid-range obstacle with distance annotation, or a clear corridor of length. This ensures that the robot's full surrounding context is expressed in textual form at the beginning of navigation. We obtain $\mathscr{L}_{\text{spatial}}^{(12)} = \{\ell_1, \ell_2, \ldots, \ell_{12}\}$, where each ℓ_i describes the spatial information at orientation $30° \times (i-1)$.

In addition to spatial descriptions, we incorporate semantic information from RGB observations. Specifically, we utilise the RAM model [26] to identify a set of explicitly detected semantic objects from each RGB image. For the frontal view setting, this yields three object lists $\mathscr{O}^{(3)} = \{\mathscr{O}_L, \mathscr{O}_F, \mathscr{O}_R\}$, corresponding to the left, front, and right RGB images, $I_{1\ldots3}^{rgb}$. During the panoramic setting with 12 views, we obtain 12 object lists $\mathscr{O}^{(12)} = \{\mathscr{O}_1, \mathscr{O}_2, \ldots, \mathscr{O}_{12}\}$, where each $\mathscr{O}_i$ denotes the detected objects from the i-th RGB image in the sequence $I_{1\ldots12}^{rgb}$.

Multimodal Prompt Construction: At the beginning of a navigation episode, the robot's orientation is unknown and

no actions have yet been executed. The input at the initial step consists of: (1) the instruction text $\mathscr{I}$. (2) semantic descriptions $\mathscr{O}^{(12)}$ and spatial descriptions $\mathscr{L}^{(12)}_{\text{spatial}}$, which together provide a comprehensive understanding of the scene to the MLLM. (3) the task description $\mathscr{T}$, which defines the high-level objective and operational constraints of the navigation task. (4) the 12 panoramic RGB images $I^{rgb}_{1...12}$. Given this multimodal input, the model reasons about the scene and the instruction to determine the most promising initial direction and movement. It produces the following structured outputs:

- **Thought:** A natural language explanation of how the instruction and scene observations were interpreted to choose the initial direction.
- **Selected Image (1–12):** The index of the RGB image that best represents the direction the robot should initially face and begin moving toward.
- **Safe Distance:** A forward distance (in meters) the robot can safely travel in the selected direction, constrained by nearby obstacles.
- **Trajectory Summary:** A natural language overview describing the surrounding environment based on the panoramic observation.
- **Instruction Progress:** Ask $\mathscr{I}$ split into subgoals with status: first subgoal `In Progress`, the rest `Not Started`, and none `Completed`.

Step-wise Multimodal Reasoning: After the initial orientation is determined, the robot continues navigation with frontal view. The inputs include: (1) three RGB images from different views, namely left ($-30°$), center ($0°$), and right ($+30°$). (2) textual navigation instruction $\mathscr{I}$, semantic descriptions $\mathscr{O}^{(3)}$ and spatial descriptions $\mathscr{L}^{(5)}_{\text{spatial}}$, and a list of valid high-level action options. (3) contextual information from the previous step, including the Observed objects $\mathscr{O}$, `Previous Selected Image` (`Selected Image` from last step), `Instruction Progress`, `Trajectory Summary`, and the `Previous Thought` that supported the last action.

As shown in Fig. 3, once MLLM receives input, it follows a structured reasoning process to select the next action. This process involves analysing candidate views, predicting future observations, analysing last action (More details list in Sec. IV-B.2), evaluating stop conditions, and estimating safe moving parameters.

The final output of each step is represented in a structured JSON format containing the reasoning trace, the selected view, the chosen action option, action parameters, and updated navigation history. This explicit format encourages the MLLM to maintain coherent reasoning and ensures interoperability of the decision-making process throughout the navigation episode.

- **Thought**: Step-by-step reasoning trace of the current decision.
- **Selected Image**: The selected image index from the limited front views (1, 2, or 3).
- **Action Options**: A symbolic label (e.g., 'A') chosen from the available action candidates.
- **Degree**: Rotation degree if applicable (e.g., 30, 60), otherwise `null`.
- **Safe Distance**: A forward distance (in meters) the robot can safely travel in the selected direction, constrained by nearby obstacles.
- **Confuse**: A Boolean flag indicating whether the Disambiguation Module is triggered (`true` / `false`).
- **Updated History**: Structured record of navigation progress, including:
  - **Trajectory Summary**: a textual summary of the current position and past movements.
  - **Instruction Progress**: the updated subgoal completion status (`Completed` / `In Progress` / `Not Started`).

To avoid getting stuck, the robot compares the current spatial descriptions $\mathscr{L}$ with the previous ones, and performs a slight rightward shift if they are unchanged.

B. Uncertainty-Aware Reasoning

In real-world navigation, the robot often encounters challenges like vague instructions, sensor noise, or conflicting visual and language inputs. These uncertainties can lead to poor decisions or cause the robot to get stuck in local areas. To enhance robustness and decision quality under such uncertainty, we propose an **Uncertainty-Aware Reasoning** module, which equips the robot with the ability to detect uncertainty and reason over both future and past context.

This module consists of two complementary mechanisms:

- **Disambiguation Module**: explicitly detects ambiguous and triggers re-evaluation to avoid poor local decisions.
- **Future-Past Bidirectional Reasoning (FPBR)**: encourages the robot to reason forward (simulate outcomes) and backward (reflect on past decisions) to ensure globally coherent navigation.

1) Disambiguation Module: Navigation instructions may be ambiguous, under-specified, or visually contradictory with the current scene. To handle such cases, we propose a Disambiguation Module that performs an explicit uncertainty check. After the initial step, the robot invokes the MLLM to determine whether the robot is "confused". The binary decision `Confuse: true` is triggered if any of the following conditions are met: (1) the instruction is ambiguous, (2) the goal is unclear, or (3) the visual cues conflict with the expected progression.

When confusion is detected, the robot performs a full 360° rotation to collect panoramic observations, consisting of 12 RGB images $I^{\text{rgb}}_{1...12}$, semantic descriptions $\mathscr{O}^{(12)}$, and spatial descriptions $\mathscr{L}^{(12)}_{\text{spatial}}$, same as the initial observation. It also incorporates contextual inputs updated from the previous step: a `Trajectory Summary` and `Instruction Progress`, which respectively describe the robot's navigation history and the completed portions of the instruction.

We construct a structured prompt to guide the MLLM through reasoning over this information. The prompt instructs the model to:

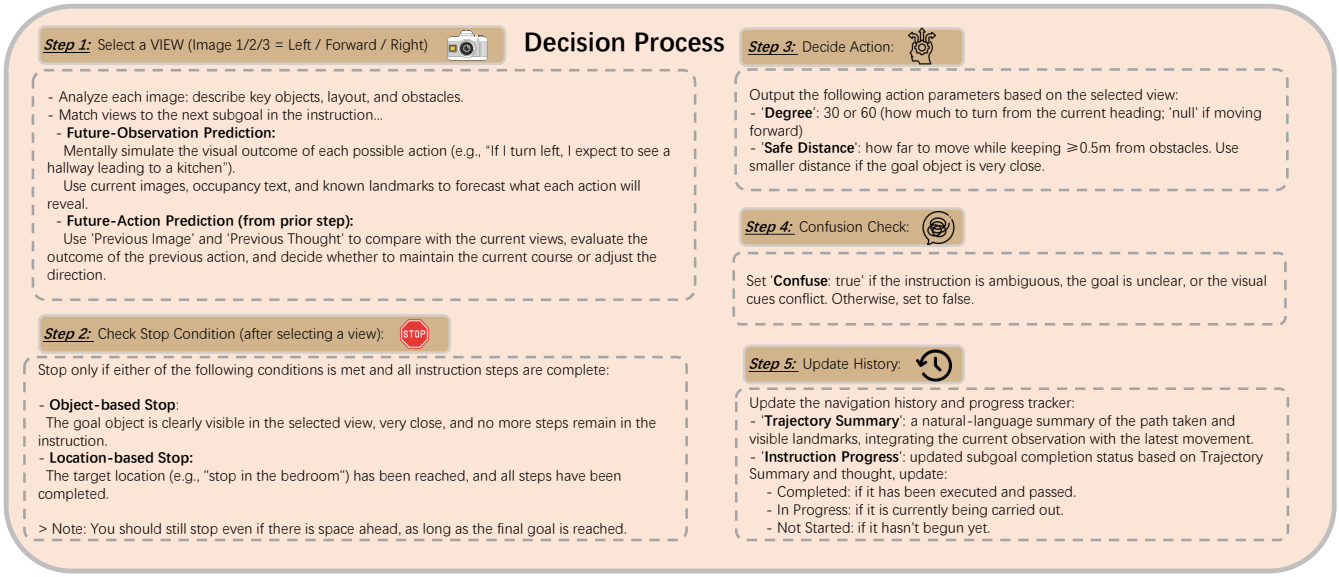

Fig. 3. Overview of the structured Decision Process in the MLLM-based Step-wise navigation pipeline. Upon receiving inputs, the model evaluates candidate views, predicts future observations, analyses the previous action, determines stop conditions, and estimates safe moving distance with selected direction.

1) Identify completed steps in the instruction.
2) Detect any misalignment between the current heading and the instruction intent.
3) Recommend a re-orientation direction and safe distance to resume navigation.

The MLLM returns a structured JSON object:

- **Thought**: Step-by-step reasoning trace of the current decision.
- **Selected Image**: The index of the RGB image that best represents the direction the robot should face and move toward.
- **Safe Distance**: A forward distance (in meters) the robot can safely travel in the selected direction, constrained by nearby obstacles.
- **Updated History**: Structured record of navigation progress, including:
  - **Trajectory Summary**: a textual summary of the current position and past movements.
  - **Instruction Progress**: the updated subgoal completion status (`Completed` / `In Progress` / `Not Started`).

2) Future-Past Bidirectional Reasoning: To maintain temporal consistency and improve long-horizon planning, we introduce a Future-Past Bidirectional Reasoning (FPBR) mechanism by formulating a structured prompt that explicitly guides the agent reasoning process. Inspired by Local Observation-Action Reasoning [17], this component prompts the model to reason about possible future states while reflecting on past actions.

Future Prediction: The robot is prompted to mentally simulate the visual consequences of alternative navigational actions (e.g., "If I turn left, I expect to see a hallway leading to a kitchen"). This simulation is conditioned on current RGB and semantic inputs, surrounding spatial descriptions, and any landmarks extracted from the instruction. The model evaluates the plausibility of each action by comparing expected visual outcomes with the goal context.

Past Recall: The model also incorporates the `Previous Selected Image` and `Previous Thought` to recall its previous decision. It compares the current observation at step t with what it predicted at step $t-1$. If a mismatch is detected (e.g., expected to see the "chicken" but sees a "bed"), the model may revise its policy accordingly.

This bidirectional reasoning promotes both short-term correction and long-term instruction alignment. By grounding each decision in both simulated futures and reflective pasts, the robot avoids cascading errors and achieves more coherent navigation trajectories.

V. Experiments

A. Experiment Setup

Implementation Details For zero-shot navigation, the MLLM-based navigator is deployed via the GPT-4o-2024-08-06 API, which is the same as baseline [16]. The occupancy distance thresholds are set as $d_{\text{close}} = 0.5$ and $d_{\text{mid}} = 4$. Due to the randomness in MLLM outputs, each experiment is run four times, and we report the average.

Environments in the Simulator and Real World Our simulator implementation is based on the Habitat simulator [29]. Following the evaluation protocol of Open-Nav [15], we assess our navigator with the same 100 episodes to ensure direct comparability with prior work.

For real-world validation, we deploy our system on the Hello Robot platform [18] to evaluate per-step latency and deployment efficiency, while navigation performance is also measured exclusively on the robot. The Hello Robot is equipped with an Intel RealSense D435if RGB-D camera, and all onboard inference is executed on a workstation with the NVIDIA RTX 4090 GPU during operation.

TABLE I

COMPARISON OF REAL-WORLD NAVIGATION EFFICIENCY AND PERFORMANCE ON HELLO ROBOT

Method	View Type	Perception Time (s)	Inference Time (s)	Total Time (s)	SR↑	NE↓
RecBERT [7]	Panoramic	22.40	0.02	22.42	20	3.92
SmartWay [16]	Panoramic	22.40	6.85	29.25	32	3.01
Ours	Frontal View	5.13	7.26	12.39	**36**	**2.78**

TABLE II

COMPARISON ON SIMULATED ENVIRONMENT IN R2R-CE DATASET

Method	View Type	TL	NE↓	nDTW↑	SR↑	SPL↑
Supervised						
CMA[7]	Panoramic	11.08	6.92	50.77	37	32.17
RecBERT[7]	Panoramic	11.06	5.80	54.81	48	43.22
BEVBert[8]	Panoramic	13.63	5.13	61.40	60	53.41
ETPNav[9]	Panoramic	11.08	5.15	61.15	52	52.18
RecBERT[7]	Frontal View	7.89	7.78	40.78	6	5.80
Navid [24]	Frontal View	9.98	7.24	-	28	21.40
Zero-Shot						
Random	Panoramic	8.15	8.63	34.08	2	1.50
LXMERT[27]	Panoramic	15.79	10.48	18.73	2	1.87
MapGPT-CE-GPT4o [28]	Panoramic	12.63	8.16	27.38	7	5.04
DiscussNav-GPT4[23]	Panoramic	6.27	7.77	42.87	11	10.51
Open-Nav-Llama3.1[15]	Panoramic	8.07	7.25	44.99	16	12.90
Open-Nav-GPT4[15]	Panoramic	7.68	6.70	45.79	19	16.10
Smartway[16]	Panoramic	16.01 ± 0.85	6.81 ± 0.38	41.77 ± 2.42	**29.00 ± 2.94**	22.08 ± 2.92
Random	Frontal View	4.81 ± 0.19	9.24 ± 0.07	33.22 ± 0.22	1.25 ± 0.96	1.18 ± 0.88
Smartway[16]	Frontal View	10.91 ± 0.49	8.50 ± 0.28	41.19 ± 0.69	7.25 ± 3.77	6.32 ± 3.16
Ours	Frontal View	12.56 ± 0.71	7.72 ± 0.42	**51.83 ± 1.54**	27.75 ± 2.22	**24.95 ± 2.70**

To assess the system's generalisation in real-world settings, following the experimental setup from SmartWay [16] and DiscussNav [23], we collect 25 diverse trajectories across multiple rooms, including open-vocabulary target landmarks (e.g., green bin). These trajectories vary in instruction complexity and spatial layout, enabling a comprehensive evaluation of our zero-shot navigation policy under realistic conditions.

Evaluation Metrics We evaluate our navigator against several learning-based and zero-shot VLN methods using standard VLN metrics [7], including Success Rate (SR), normalized Dynamic Time Warping (nDTW), Success weighted by Path Length (SPL), Trajectory Length (TL), and Navigation Error (NE). Following prior works [15], [16], we consider a navigation to be successful if the robot stops within 3 meters of the goal in the simulator, and within 2 meters in real-world robot experiments.

B. Results in Real-world Environments

Table I compares our method with two existing baselines: a supervised approach (RecBERT [7]) and a zero-shot method (SmartWay [16]). Both baselines rely on panoramic inputs and operate on the Hello Robot platform [18]. RecBERT achieves the lowest inference time (0.02s), but its overall performance is limited, with a success rate (SR) of only 20 and a navigation error (NE) of 3.92. SmartWay improves these metrics (SR = 32, NE = 3.01), but its inference time increases to 6.85s, leading to the highest overall latency of 29.25s. Here, the perception time refers to the time spent by the real-world robot on collecting visual observations, while the inference time is the average model prediction time per step.

In contrast, our method leverages only the frontal-view images, significantly reducing the perception time to 5.13s. Although its inference time is slightly higher (7.26s), the total latency is reduced to 12.39s, only 42.4% of SmartWay's total time. More critically, our method achieves the best navigation performance with the highest success rate (SR = 36) and the lowest navigation error (NE = 2.78), demonstrating its strong effectiveness under real-world constraints.

These results highlight that our approach, relying solely on front-view inputs, accelerates deployment and achieves strong navigation performance, demonstrating its practicality and scalability for real-world robotic navigation tasks.

C. Results in Simulator

As shown in Table II, our method achieves strong zero-shot navigation performance on the R2R-CE benchmark, despite relying solely on frontal views rather than full panoramic inputs. Among panoramic-view baselines, Open-Nav-GPT4 [15] and SmartWay [16] perform well with SR of 19 and 29, and SPL of 16.10 and 22.08, respectively. In

TABLE III

ABLATION STUDY ON THE IMPACT OF THE DISAMBIGUATION MODULE AND FUTURE-PAST BIDIRECTIONAL REASONING (FPBR).

Method	TL	NE↓	nDTW↑	SR↑	SPL↑
Ours (w/o Disambiguation and FPBR)	11.69 ± 0.63	7.64 ± 0.47	50.83 ± 1.83	19.75 ± 4.19	17.42 ± 2.49
+ Disambiguation only	12.21 ± 0.45	7.64 ± 0.49	**52.06 ± 1.42**	24.25 ± 1.26	21.14 ± 1.77
Full Model (with Disambiguation + FPBR)	12.56 ± 0.71	7.72 ± 0.42	51.83 ± 1.54	**27.75 ± 2.22**	**24.95 ± 2.70**

comparison, our method achieves a competitive SR of 27.75 and surpasses both baselines in SPL (24.95) and nDTW of 51.83 (vs. 41.77 for SmartWay and 45.79 for Open-Nav-GPT4). For fair comparison, we reproduce frontal-view baselines using our own implementation.

Overall, the results show that even with less visual input, our model performs competitively with panoramic-based methods. These findings demonstrate that our frontal-view approach offers a strong balance of efficiency and effectiveness, making it well-suited for real-world deployment.

D. Ablation Study

Table III shows the ablation results on the R2R-CE dataset. When both the Disambiguation module and Future-Past Bidirectional Reasoning (FPBR) are removed, performance drops significantly (SR: 19.75, SPL: 17.42). Adding only the Disambiguation module boosts SR to 24.25 and SPL to 21.14. With both modules included, the full model achieves the best results (SR: 27.75, SPL: 24.95), confirming that each component contributes to stronger and more reliable navigation.

VI. CONCLUSION

We presented Fast-SmartWay, a zero-shot VLN-CE framework that removes the need for panoramic view and waypoint predictors. Instead, it uses only three frontal views and a multimodal large language model to directly predict actions. To improve robustness and planning consistency, we introduced an Uncertainty-Aware Reasoning module that combines disambiguation and future-past bidirectional reasoning. Experiments in both simulation and real-world settings show that Fast-SmartWay achieves performance comparable to state-of-the-art (SOTA) panoramic view zero-shot methods, while offering much faster processing time. Our results demonstrate the potential of efficient end-to-end multimodal reasoning for scalable and practical navigation systems in real-world scenarios.

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 *CVPR*, 2018, pp. 3674–3683.

[2] 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 *ICCV*, 2023, pp. 12 009–12 020.

[3] Y. Hong, Q. Wu, Y. Qi, C. Rodriguez-Opazo, and S. Gould, "Vln bert: A recurrent vision-and-language bert for navigation," in *CVPR*, 2021, pp. 1643–1653.

[4] S. Chen, P.-L. Guhur, C. Schmid, and I. Laptev, "History aware multimodal transformer for vision-and-language navigation," *NeurIPS*, vol. 34, pp. 5834–5847, 2021.

[5] Y. Qiao, Y. Qi, Y. Hong, Z. Yu, P. Wang, and Q. Wu, "Hop: History-and-order aware pre-training for vision-and-language navigation," in *CVPR*, 2022, pp. 15 418–15 427.

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

[7] 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 *CVPR*, 2022, pp. 15 439–15 449.

[8] D. An, Y. Qi, Y. Li, Y. Huang, L. Wang, T. Tan, and J. Shao, "Bevbert: Multimodal map pre-training for language-guided navigation," *arXiv preprint arXiv:2212.04385*, 2022.

[9] 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," *TPAMI*, 2024.

[10] Z. Li, G. Zhou, H. Hong, Y. Shao, W. Lyu, Y. Qiao, and Q. Wu, "Ground-level viewpoint vision-and-language navigation in continuous environments," in *ICRA*, 2025.

[11] J. Krantz and S. Lee, "Sim-2-sim transfer for vision-and-language navigation in continuous environments," in *ECCV*, 2022, pp. 588–603.

[12] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[13] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[14] G. Zhou, Y. Hong, and Q. Wu, "Navgpt: Explicit reasoning in vision-and-language navigation with large language models," in *AAAI*, vol. 38, no. 7, 2024, pp. 7641–7649.

[15] 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 *ICRA*, 2025.

[16] 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," in *IROS*, 2025.

[17] Y. Qiao, H. Hong, W. Lyu, D. An, S. Zhang, Y. Xie, X. Wang, and Q. Wu, "Navbench: Probing multimodal large language models for embodied navigation," *arXiv preprint arXiv:2506.01031*, 2025.

[18] C. C. Kemp, A. Edsinger, H. M. Clever, and B. Matulevich, "The design of stretch: A compact, lightweight mobile manipulator for indoor human environments," in *ICRA*. IEEE, 2022, pp. 3150–3157.

[19] Y. Zhang, Z. Ma, J. Li, Y. Qiao, Z. Wang, J. Chai, Q. Wu, M. Bansal, and P. Kordjamshidi, "Vision-and-language navigation today and tomorrow: A survey in the era of foundation models," *arXiv preprint arXiv:2407.07035*, 2024.

[20] Y. Qi, Q. Wu, P. Anderson, X. Wang, W. Y. Wang, C. Shen, and A. v. d. Hengel, "Reverie: Remote embodied visual referring expression in real indoor environments," in *CVPR*, 2020, pp. 9982–9991.

[21] A. Ku, P. Anderson, R. Patel, E. Ie, and J. Baldridge, "Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding," in *EMNLP*, 2020, pp. 4392–4412.

[22] J. Thomason, M. Murray, M. Cakmak, and L. Zettlemoyer, "Vision-and-dialog navigation," in *CoRL*, 2020, pp. 394–406.

[23] Y. Long, X. Li, W. Cai, and H. Dong, "Discuss before moving: Visual language navigation via multi-expert discussions," *ICRA*, 2024.

[24] J. Zhang, K. Wang, R. Xu, G. Zhou, Y. Hong, X. Fang, Q. Wu, Z. Zhang, and H. Wang, "Navid: Video-based vlm plans the next step for vision-and-language navigation," *arXiv preprint arXiv:2402.15852*, 2024.

[25] Z. Wang, X. Li, J. Yang, Y. Liu, and S. Jiang, "Sim-to-real transfer via 3d feature fields for vision-and-language navigation," *arXiv preprint arXiv:2406.09798*, 2024.

[26] 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," *arXiv preprint arXiv:2306.03514*, 2023.

[27] H. Tan and M. Bansal, "Lxmert: Learning cross-modality encoder representations from transformers," in *EMNLP-IJCNLP*, 2019, pp. 5100–5111.

[28] 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 *ACL*, 2024, pp. 9796–9810.

[29] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik *et al.*, "Habitat: A platform for embodied ai research," in *ICCV*, 2019, pp. 9339–9347.