CompassNav: Steering From Path Imitation To Decision Understanding In Navigation

LinFeng Li[1,2] Jian Zhao[2,†] Yuan Xie[1] Xin Tan[1,†] Xuelong Li[2]

[1]East China Normal University
[2]The Institute of Artificial Intelligence (TeleAI), China Telecom

Project page: https://linengcs.github.io/CompassNav

ABSTRACT

The dominant paradigm for training Large Vision-Language Models (LVLMs) in navigation relies on imitating expert trajectories. This approach reduces the complex navigation task to a sequence-to-sequence replication of a single correct path, fundamentally limiting the agent's ability to explore and generalize. In this work, we argue for and introduce a new paradigm: a shift from Path Imitation to Decision Understanding. The goal of this paradigm is to build agents that do not just follow, but truly understand how to navigate. We materialize this through two core contributions: first, we introduce Compass-Data-22k, a novel 22k-trajectory dataset.Its Reinforcement Fine-Tuning (RFT) subset provides a panoramic view of the decision landscape by annotating all feasible actions with A* geodesic distances. Second, we design a novel gap-aware hybrid reward function that dynamically adapts its feedback to decision certainty, shifting between decisive signals for optimal actions and nuanced scores to encourage exploration. Integrated into an SFT-then-RFT recipe, our CompassNav agent is trained not to memorize static routes, but to develop an internal "compass" that constantly intuits the direction to the goal by evaluating the relative quality of all possible moves. This approach enables our 7B agent to set a new state-of-the-art on Goal navigation benchmarks, outperforming even larger proprietary models, and achieve robust real-world goal navigation on a physical robot.

1 Introduction

The ultimate goal for embodied agents is to operate autonomously in complex, unseen environments. A cornerstone of this autonomy is the ability to explore freely and reason spatially to achieve goals without explicit, step-by-step guidance. To rigorously evaluate this crucial capability, we focus on goal-driven navigation, where the task is specified by a category (ObjectNav) (Chaplot et al., 2020) or a goal image (Image-Goal) (Krantz et al., 2022; Li et al., 2025).

This paradigm presents a fundamentally different and more challenging test of autonomy compared to Vision-and-Language Navigation (VLN) (Anderson et al., 2018). Whereas VLN provides agents with dense, semantic, turn-by-turn instructions, effectively constraining the problem to language grounding, goal-driven navigation offers only a sparse, high-level goal (e.g., "find a chair"). This absence of explicit instructions shifts the core difficulty *from* language grounding *to* genuine autonomous exploration and robust spatial reasoning under profound uncertainty. It is this sparse-signal, high-ambiguity setting—which mirrors real-world intelligent behavior—that exposes a critical flaw in the prevailing training paradigms for LVLM-based agents.

† Corresponding authors.

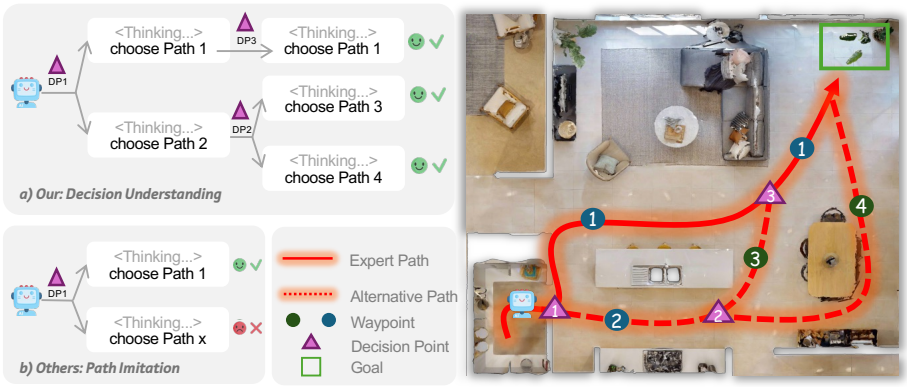


Figure 1: A visualization of our proposed paradigm shift from Path Imitation to Decision Understanding. (Bottom Left) The prevailing Path Imitation paradigm treats navigation as the replication of a single expert path (solid line), penalizing all other choices. (Top Left) In contrast, our Decision Understanding paradigm teaches the agent to evaluate the relative value of all alternative paths (dashed line), enabling flexible judgment at critical decision points. (Right) A concrete instantiation of these concepts in a navigation scenario.

At present, embodied navigation training paradigms (Liu et al., 2025a; Qi et al., 2025; Gao et al., 2025) are limited to imitation learning, specifically imitating a predefined trajectory (Krantz et al., 2020; Ku et al., 2020). This approach reduces complex navigation tasks to replicating a single standard solution, failing to teach the agent to comprehend the richness of the decision space or the semantic equivalence of different viable paths. In the real world, there are often multiple effective paths to a target. Rigidly imitating one path stifles exploration and harms generalization. To address this, we propose a paradigm shift from *Path Imitation* to *Decision Understanding*. Our framework is designed not to provide an 'answer key' to be memorized, but to instill a robust internal compass that guides genuine decision-making. We visualize this core conceptual shift in Figure 6.

Realizing this new paradigm with accessible, open-source models, however, faces significant hurdles. While powerful closed-source models (Achiam et al., 2023; Team et al., 2024) exist, their cost and latency are prohibitive. Conversely, open-source LVLMs (Bai et al., 2025; Wang et al., 2024; Dubey et al., 2024) lack the specialized data and alignment techniques required. Our work forges a complete and reproducible pathway to empower these open-source models, moving beyond simple imitation.

To build an agent with such an internal compass and operationalize this paradigm, we make two primary contributions. First, we construct and release Compass-Data-22k, a new 22k-trajectory navigation dataset specifically designed for decision understanding. Generated via a novel pipeline, it provides a panoramic view of the decision space by recording A* geodesic distances for *all* feasible actions at every step in its Reinforcement Fine-Tuning (RFT) subset. Second, we design a novel gap-aware hybrid reward function to tackle the critical bottleneck of reward design (Kwon et al., 2023), as standard binary rewards (Rafailov et al., 2024; Ouyang et al., 2022) are ill-suited for navigation's nuances. Our reward acts as the core training mechanism, dynamically assessing decision certainty at each step. It is both *relative*—offering nuanced scores in ambiguous situations to encourage exploration—and *decisive*, providing a strong, focused reward signal when a clearly superior path exists.

Finally, to ensure our agent is prepared to develop this sophisticated reasoning capability, we address the "cold-start" problem (Zhang et al., 2025b) with a preparatory Supervised Fine-Tuning (SFT) stage. This entire SFT-then-RFT recipe provides an integrated solution that operationalizes our new paradigm, enabling agents to learn how to weigh options and decide, rather than just follow. Our core technical contributions are:

- **The Compass-Data-22k Dataset for Decision Understanding.** We introduce a novel 22k-trajectory dataset designed to embody our new paradigm. It comprises an 11k-sample set of distilled reasoning traces for Supervised Fine-Tuning (SFT) and a corresponding 11k-sample set for Reinforcement Fine-Tuning (RFT), which shifts supervision from single-path imitation to a panoramic view by densely annotating all feasible actions with A* geodesic distances.

- **A Gap-Aware Hybrid Reward for Nuanced Policy Alignment.** We design a novel reward function that acts as a sophisticated training signal by dynamically adapting its feedback to the decision's certainty. It robustly handles ambiguity with relative scores while providing decisive signals in clear-cut scenarios, thereby cultivating genuine decision-making skills by fully leveraging the dense annotations in the RFT subset of Compass-Data-22k.

2 Related Work

Approaches to Goal-Driven Navigation. Goal-driven navigation has traditionally been tackled by modular pipelines that separate perception, mapping, and planning (Yokoyama et al., 2024a), like CogNav (Cao et al., 2024) and UniGoal (Yin et al., 2025), which construct complex, explicit world representations (e.g., cognitive maps) to achieve strong results. However, these multi-stage systems aren't suitable for flexible human-computer interaction and suffer from significant engineering complexity. While end-to-end LVLMs offer a more unified alternative, current approaches have diverged into two main streams. One stream leverages powerful, closed-source models for zero-shot navigation (Zhang et al., 2025a; Goetting et al., 2024), but this method faces prohibitive barriers in cost and latency. Consequently, another stream has emerged to fine-tune open-source LVLMs on existing trajectory data from tasks like R2R and RxR (Liu et al., 2025a; Qi et al., 2025; Gao et al., 2025), aiming to bridge the capability gap. However, the performance of these fine-tuned models has remained modest, a result of limited data availability, incomplete alignment techniques, and, most critically, a singular and restrictive training paradigm based on imitation.

Reward Design for Reinforcement Fine-Tuning. The efficacy of Reinforcement Fine-Tuning (RFT) (Sun et al., 2025) in navigation is critically dependent on reward design. While general reward strategies range from sparse to dense, sparse signals (Yang et al., 2024; Liu et al., 2025b) are intractable for navigation's long horizons. However, common dense rewards are also ill-suited. Simple heuristics like Euclidean distance are geometrically naive and fail around obstacles, while standard preference-based methods (Rafailov et al., 2024) use binary signals that cannot capture the crucial nuances—such as preference magnitude and ambiguity—inherent in navigation's multi-option decision space.

Reflecting these challenges, current RFT applications in navigation have been limited to a restrictive imitation paradigm. For instance, Nav-R1 (Liu et al., 2025a) calculates fidelity against a *single reference trajectory*, while others use simple binary rewards for the expert's action (Qi et al., 2025; Gao et al., 2025). This feedback style reinforces path replication, not genuine understanding.

This imitation-style feedback, whether trajectory-based or binary, merely reinforces replication of an answer key rather than fostering true decision-making. To enable our proposed decision understanding paradigm, a reward mechanism that acts as a teacher is essential. This requires a shift to a dense signal over *all* possible actions, evaluating the relative merit of each choice. Our gap-aware hybrid reward, which leverages a panoramic view of the decision space provided by A* annotations, is designed precisely for this purpose, representing a fundamental departure from prior reward designs.

3 The Compass-Data Dataset: Generation and Formulation

We frame goal-driven navigation as a sequential decision-making problem. At each step, an agent observes its current view, is presented with a set of feasible candidate actions, and must select one to execute (further details are in Appendix B). This section details the generation and formulation of our novel dataset, Compass-Data, which is specifically designed to shift the training paradigm from path imitation to decision understanding, as illustrated in Figure 2.

3.1 Compass-Data-RFT: Dense Annotation for Decision Understanding

A core prerequisite for teaching decision-making is a dataset that provides dense, fine-grained supervision. Existing trajectory datasets from VLN (Anderson et al., 2018), such as R2R (Krantz et al., 2020) and RxR (Ku et al., 2020), are unsuitable for this purpose due to two fundamental mismatches. First, a goal-suitability mismatch: VLN trajectories are generated to satisfy language instructions, meaning their terminal viewpoints are often arbitrary and not framed around a specific object. Second, and more critically, a supervision-granularity mismatch: these datasets provide only

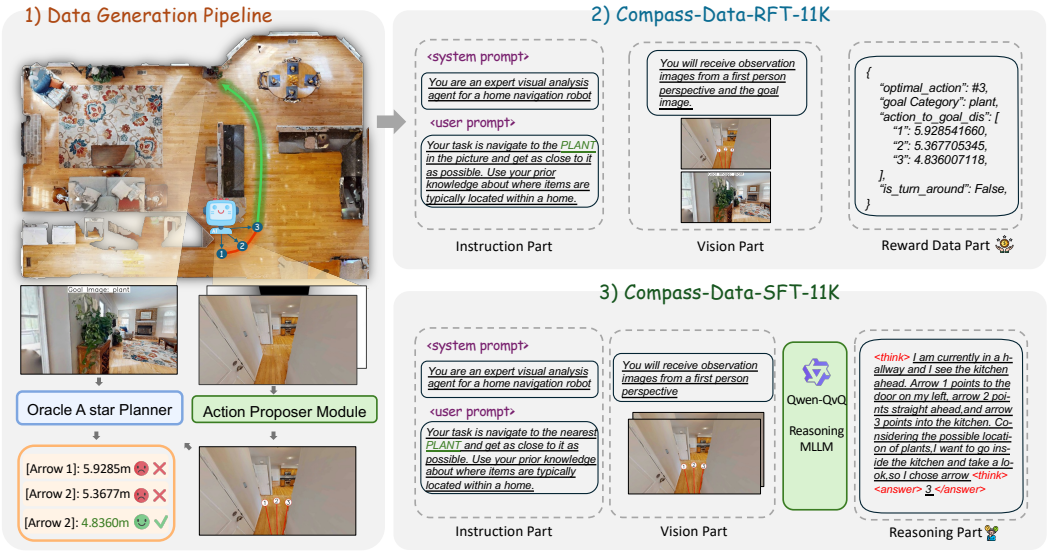


Figure 2: **Overview of our data generation and formulation pipeline.** (1) Data Generation: We use an A* planner in `habitat-sim` to densely annotate all feasible actions with their distance to the goal. (2) Dataset Formulation: Trajectories are formatted into two types. SFT data (bottom right) contains a teacher's reasoning trace for imitation. RFT data (top right) contains the full vector of action distances for reward modeling.

a single ground-truth path, lacking the dense, per-step annotations for alternative actions. Simply imitating a lone optimal trajectory fails to teach the model the relative quality of alternative choices. To learn effective decision-making, an agent must understand not only the best action but also *why* other options are better or worse, which requires a dense preference signal over all possible actions at each step.

To create this dense signal, we built a data generation pipeline to produce the RFT-focused part of our dataset, Compass-Data-RFT-11k. As shown in Figure 2 (Panel 1), the pipeline uses an Action Proposer Module to generate feasible candidate actions at each timestep, represented as high-level tuples (r, θ) and rendered as arrows on the agent's view. An oracle A* planner then computes the geodesic distance for each candidate. While the agent primarily follows the optimal path, our pipeline incorporates a crucial backtracking mechanism: at ambiguous decision points, where multiple actions are nearly optimal, the agent explores and records these alternative viable paths. Crucially, we record the computed distances for all candidate actions at every step along all explored trajectories. This dual approach yields a collection of diverse, goal-centric trajectories, where each step is annotated with a panoramic, fine-grained supervisory signal, capturing not just one but multiple effective routes to the goal.

After generation, this densely annotated data is structured for the Reward Fine-Tuning stage. To maintain a consistent structure throughout our framework, we establish a standard data format. As shown in Figure 2 (top right), each RFT data sample consists of a standard input (the instructional prompt and the agent's current visual observation) paired with a specialized target for reward modeling. This target object contains the optimal action's ID and, most importantly, the complete vector of A* distances for all candidate actions at that step. This vector provides the fine-grained, graded preference signal that is essential for our gap-aware hybrid reward function and the GRPO framework.

3.2 Compass-Data-SFT: Knowledge Distillation for Policy Initialization

Initializing RFT directly from a base LVLM is often inefficient due to the "cold-start" problem (Zhang et al., 2025b). A base model's initial policy is typically poor, leading to suboptimal actions and sparse reward signals that hinder learning. To address this, we introduce a preparatory Supervised Fine-Tuning (SFT) stage using our Compass-Data-SFT-11k set, which is designed to instill a foundational "reason-then-act" capability.

Instead of constructing post-hoc reasoning for a predefined path, we adopt a more ecologically valid knowledge distillation strategy. We task a powerful teacher model, Qwen-QvQ (Team, 2024), to genuinely perform the ObjectNav task in `habitat-sim`. By recording the complete, step-by-step reasoning and action choices only from its successful episodes, we form an SFT dataset that reflects emergent and effective exploratory strategies.

SFT Data Structure. As shown in Figure 2 (bottom right), each SFT training instance shares the same input structure as the RFT data:(a) an instructional prompt and (b) the agent's current visual observation. Its distinction lies in the target output, which is a single string containing the teacher's full cognitive process and decision, formatted as `<think>...reasoning...</think><answer>k</answer>`. This format explicitly trains the model to externalize its reasoning before committing to an action, establishing the foundational 'reason-then-act' behavior. We intentionally exclude historical frames from the input to ensure compatibility with external memory modules. This design choice is deliberate, as our empirical validation showed that current methods for integrating historical context directly into the training process, such as using text summaries or concatenated images, consistently led to a degradation in performance(Details can be found in Appendix E.2).

4 CompassNav Framework

CompassNav is a two-stage fine-tuning recipe designed to first initialize a robust policy and then align it with environmental objectives. We adopt this SFT-then-RFT structure to create a synergistic learning process. The initial SFT stage efficiently overcomes the "cold-start" problem by instilling a strong policy prior through imitation. Subsequently, the RFT stage leverages this competent initial policy to align the agent towards genuine decision understanding. The entire process is visually summarized in Figure 3.

4.1 Stage 1: Policy Initialization via Supervised Fine-Tuning

While our ultimate goal is to move beyond imitation, a foundational reasoning capability is a prerequisite. Stage 1 thus employs a targeted form of imitation (SFT) to instill this 'reason-then-act' structure by learning from a teacher model's reasoning process. At each timestep t, the model is trained to generate a two-part response that mimics the teacher's output: a chain-of-thought rationale followed by a final action choice, formatted as `<think>` ... `</think>` `<answer>`k `</answer>`.

A key practical challenge is ensuring the model's action choice 'k' is always a valid action from the set of available candidates $\mathcal{A}_t$. To enforce this, we employ masked multiple-choice decoding. This is achieved by applying a masked softmax over the decoder's output logits $\mathbf{z}$ for the answer token, restricting the vocabulary to only the set of valid candidate indices $\mathcal{V}_t$:

$$\pi_\theta(j \mid x_t) = \frac{\exp(z_j)}{\sum_{j' \in \mathcal{V}_t} \exp(z_{j'})} \quad \text{for } j \in \mathcal{V}_t \tag{1}$$

This simple and effective technique ensures that all generated outputs are executable, which is crucial for the stability of the subsequent RFT stage.

The SFT loss is a standard cross-entropy loss over the entire teacher-generated sequence, including both the reasoning tokens and the final action token:

$$\mathcal{L}_{\text{SFT}}(\theta) = \mathbb{E}_{(x_t, y_t)} \left[\sum_{u=1}^{|y_t|} -\log p_\theta\big(y_{t,u} \mid x_t, y_{t,<u}\big) \right] \tag{2}$$

where y_t represents the full target sequence `<think>...k</answer>`.

4.2 Stage 2: Policy Alignment with Gap-Aware hybrid Reward Tuning

In Stage 2, we align the SFT-initialized policy with environmental objectives using the Group-wise Reward Policy Optimization (GRPO) (Shao et al., 2024) framework, as shown in Figure 3 (Stage 2). The learning signal is provided by our novel gap-aware hybrid reward function.

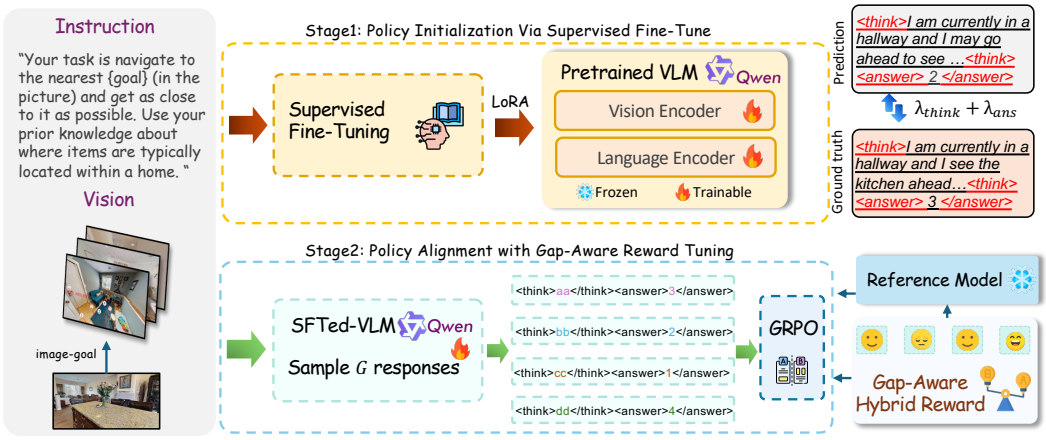


Figure 3: **The CompassNav two-stage training pipeline.** In Stage 1 (SFT), a pretrained VLM is fine-tuned to imitate a teacher's "reason-then-act" output. In Stage 2 (RFT), the SFT-tuned policy generates multiple responses, which are scored by our Gap-Aware Hybrid Reward function.

GRPO Sampling and Reward Assignment. For a given input prompt x_t, we use the policy π_θ to generate a group of G distinct output sequences, $\{y_1, y_2, \ldots, y_G\}$. For each generated sequence y_j, we parse it to extract the chosen action i_j. The reward for this sequence, $r(y_j)$, is then determined by the quality of this chosen action, evaluated using our gap-aware hybrid reward function and the pre-computed A* distances for all candidates.

Gap-Aware Hybrid Reward Function. decisive signal in high-certainty scenarios while remaining nuanced to encourage exploration in low-certainty ones. This is achieved by composing the reward from two key components: a continuous base score and a certainty-modulated dynamic bonus.

The base score, $s_t^{(i_j)}$, provides a continuous evaluation for all available options, is calculated using a softmax function over the vector of distances to the goal, d_t. Actions with shorter distances receive a higher score, reflecting their relative quality:

$$s_t^{(i_j)} = \frac{\exp(-d_t^{(i_j)}/\tau)}{\sum_{k \in \mathcal{A}_t} \exp(-d_t^{(k)}/\tau)}, \tag{3}$$

where τ is a temperature hyperparameter that controls the sharpness of the distribution.

The core of our adaptive mechanism lies in the dynamic bonus, which is governed by a "certainty" factor, g_t. This factor dynamically assesses whether the current situation is decisive or ambiguous by measuring the normalized gap between the best ($d_t^{(1)}$) and second-best ($d_t^{(2)}$) options:

$$g_t = \text{clip}\left(\frac{d_t^{(2)} - d_t^{(1)}}{|d_t^{(1)}| + \epsilon}, 0, 1 \right). \tag{4}$$

This normalization allows the signal to prioritize broad exploration when far from the goal and promote precision when near it. A high g_t indicates a clear, high-certainty choice, while a low g_t signifies an ambiguous one.

The final hybrid reward, $r_t^{(i_j)}$, combines the base score with the bonus, which is modulated by this certainty factor and is triggered only for the optimal action i^*:

$$r(y_j) \equiv r_t^{(i_j)} = s_t^{(i_j)} + \beta_{\max} \cdot g_t \cdot \mathbb{1}[i_j = i^*], \tag{5}$$

where $i^* = \arg\min_k d_t^{(k)}$ is the optimal action, $\mathbb{1}[\cdot]$ is the indicator function, and $\beta_{\max}$ is the preset maximum bonus ratio. The final reward value is then clipped to the range $[0, 1]$.

To illustrate its superiority, we analyze its behavior against two common baselines—a sparse Binary reward (Rafailov et al., 2023) and a linear normalized Min-Max reward (Patro and Sahu, 2015)—across three key navigation scenarios in Figure 4.

As shown in Figure 4, our Gap-Aware hybrid reward function demonstrates superior behavior in diverse scenarios. In the Decisive Case (A), it creates a large margin between the best and second-best

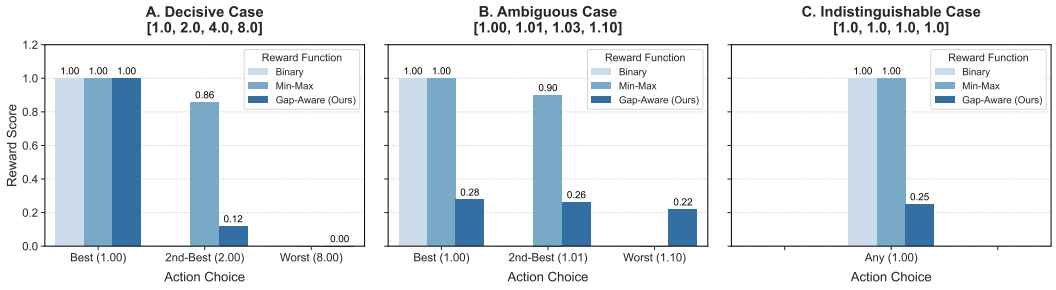


Figure 4: A comparative analysis of our Gap-Aware hybrid Reward against Binary and Min-Max schemes across three representative navigation scenarios. Each scenario shows the reward assigned for choosing the best, second-best, and worst actions.

actions (1.00 vs. 0.12), providing a strong and unambiguous learning signal where baselines fail to differentiate. In the Ambiguous Case (B), it correctly assigns similar, non-extreme scores to closely-valued options, encouraging exploration rather than arbitrarily penalizing a viable choice. Finally, in the Indistinguishable Case (C), while other methods output a misleadingly perfect score of 1.0, our function provides an honest, low score of 0.25. This prevents the agent from receiving an illusory high reward for a random guess, fostering a more robust and faithful policy.

Objective Function. The GRPO objective maximizes the expected reward over the generated group. After normalizing the rewards to produce advantages $A(y_j)$, the loss function to be minimized is:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}_{x_t,\{y_j\}}\left[\sum_{j=1}^{G} A(y_j)\log \pi_\theta(y_j|x_t)\right] + \beta_{\text{KL}} \cdot \mathbb{E}_{x_t}\left[\text{KL}(\pi_\theta(\cdot|x_t)\|\pi_{\text{SFT}}(\cdot|x_t))\right]. \quad (6)$$

Here, π_{SFT} is the frozen reference policy from Stage 1 (the "Reference Model" in Figure 3), and the KL term regularizes the policy update. This objective encourages the policy to generate sequences leading to high-reward actions, as evaluated by our fine-grained reward model.

5 Experiments

5.1 Experimental Setup

Datasets and Tasks. We generate training data in `habitat-sim` using the HM3Dv2 (Yadav et al., 2023) train split. To evaluate generalization, we test our agent on three challenging, unseen validation splits: HM3Dv1-val (Ramakrishnan et al., 2021b), HM3Dv2-val (Yadav et al., 2023) and MP3D-val (Chang et al., 2017). These splits feature completely held-out scenes and object instances, ensuring a rigorous evaluation of the agent's ability to navigate novel environments. The primary tasks are Object-Goal (Chaplot et al., 2020) and Instance-Image-Goal Navigation (Krantz et al., 2022). Further details on dataset statistics and task definitions are provided in Appendix B.

Evaluation Metrics. We report two standard metrics: Success Rate (SR) and Success weighted by Path Length (SPL). SR measures the rate of successful episodes, while SPL weights each success by the ratio of the optimal path length to the actual path taken.

Implementation Details. CompassNav is built upon the open-source Qwen2.5-VL-7B (Bai et al., 2025) model. It is trained using our two-stage SFT-then-RFT recipe. All detailed configurations, including specific training frameworks, hyperparameters, and hardware, are deferred to Appendix E.

5.2 Main Results

To comprehensively evaluate CompassNav, we structure our comparisons in two parts. First, we benchmark against state-of-the-art modular methods, which represent the dominant approach in goal-driven navigation. Second, we compare with other end-to-end LVLMs to isolate the effectiveness of our training paradigm.

Comparison with Modular Navigation Methods. First, we benchmark our end-to-end CompassNav against state-of-the-art modular systems, with results presented in Table 1. These baselines employ

Table 1: Comparison with Modular goal-driven navigation methods. We categorize methods by whether they are End-to-End (E2E) and whether they utilize Memory (ME) (e.g., semantic maps, topological map).

Method	Venue	E2E	ME	Backbone Models		HM3D		MP3D	
				Vision	Text	SR	SPL	SR	SPL
Habitat-Web	CVPR'22	✗	✗	-	-	41.5	16.0	31.6	8.5
ESC	ICML'23	✗	✓	-	GPT-3.5	39.2	22.3	28.7	14.2
L3MVN	IROS'23	✗	✓	-	GPT-2	50.4	23.1	34.9	14.5
InstructNav	CoRL'24	✗	✓	GPT-4V	LLaMA3-70B	50.0	20.9	-	-
PSL	ECCV'24	✗	✗	CLIP	-	42.4	19.2	18.9	6.4
VoroNav	ICML'24	✗	✓	BLIP	GPT-3.5	42.0	26.0	-	-
Pixel-Nav	ICRA'24	✗	✓	LLaMA-Adapter	GPT-4	37.9	20.5	-	-
VLFM	ICRA'24	✗	✓	BLIP-2	-	52.4	**30.4**	36.4	17.5
GAMap	NeurIPS'24	✗	✓	CLIP	GPT-4	53.1	26.0	-	-
SG-Nav	NeurIPS'24	✗	✓	LLaVA-1.6-7B	GPT-4	54.0	24.9	40.2	16.0
UniGoal	CVPR'25	✗	✓	LLaVA-1.6-7B	LLaMA-2-7B	54.5	25.1	41.0	16.4
CompassNav	Ours	✓	✗	Qwen2.5-VL-7B		**56.6**	27.6	**42.0**	**17.5**

Table 2: Comparison of CompassNav with various open-source and proprietary models. Our 7B parameter model is benchmarked against models of varying scales.

Models	ObjNav		InsImageNav		AVG	
	SR	SPL	SR	SPL	SR	SPL
Open-source Models						
Qwen2-VL-7B (Wang et al., 2024)	35.9	14.8	5.30	3.60	20.6	9.20
Qwen2.5-VL-3B (Bai et al., 2025)	25.8	10.3	8.70	3.20	17.3	6.80
LLama3.2-11B (Dubey et al., 2024)	25.8	8.4	3.00	2.50	17.1	5.50
Closed-Source Models						
GPT-4o (Hurst et al., 2024)	52.4	23.5	29.8	13.2	41.1	18.4
Gemini-2.5-Flash (Comanici et al., 2025)	50.0	10.6	24.1	9.70	37.1	10.2
GPT-o4-mini (OpenAI, 2025)	59.6	26.9	33.4	13.2	46.5	20.1
Our Models						
Base Model (Bai et al., 2025)	45.3	17.5	19.8	5.20	32.6	11.4
CompassNav(SFT)	54.1	23.1	23.9	7.90	39.0	15.5
CompassNav(SFT+RFT)	**61.6**	**27.8**	**35.6**	**14.8**	**48.6**	**21.3**

complex, multi-stage pipelines, often relying on explicit memory like semantic maps (Yokoyama et al., 2024a) or history images. Instead of building an explicit world model, we focus on distilling spatial reasoning capabilities directly into the model's parameters. We show that our simpler agent achieves competitive, and often superior, performance, suggesting that a focus on the learning paradigm itself is a more direct and efficient path toward capable navigation agents.

Comparison with End-to-End LVLMs. To isolate the effectiveness of our framework from the base model's inherent capabilities, we conduct a comparison against other powerful LVLMs within the same end-to-end setting. As shown in Table 2, our CompassNav agent significantly outperforms these much larger, general-purpose models like GPT-4o (Hurst et al., 2024) and even surpasses o4-mini, a model renowned for its powerful general reasoning capabilities.

We conduct a special comparison with Nav-R1 (Liu et al., 2025a), a concurrent work that also fine-tunes an LVLM for embodied tasks. Nav-R1 aims to build a general-purpose agent and has achieved progress on multiple tasks. However, its training paradigm remains rooted in imitating a single expert path. On HM3D-OVON (Yokoyama et al., 2024b)benchmark, our method surpasses Nav-R1 on both key metrics (See Figure 5 for details). Notably, we achieve it while using one-tenth of the training data and starting from a general-purpose LVLM, rather than a 3D-specialized model. This result strongly suggests that our paradigm is a more efficient and effective path.

5.3 Ablation Studies

Efficacy of the SFT Stage.

As show in Table 3, attempting RFT directly from this base model yields only a marginal 3.7 improvement in SR due to inefficient exploration from a naive policy.Applying RFT on this SFT-initialized model further improves performance by another 12.3, confirming the synergy of our two-stage approach.And recently, some methods only teach the model to output the action space of navigation tasks in the SFT stage. In difficult goal navigation tasks, this method actually deteriorates the effect.

Table 3: Efficacy of the SFT Stage.

Config.	SR	SPL
base Model	19.8	5.20
SFT(Action only)	17.9	5.78
SFT(Full)	23.3	7.90
RFT(from Scratch)	23.5	6.95
RFT(from SFT)	**35.6**	**14.8**

Table 4: Ablation studies on the components of our framework.

Max Bouns	SR	SPL
B = 0.5	31.3	12.2
B = 1.0	**35.6**	**14.8**
B = 1.5	27.9	11.7

(a) Efficacy of the Max Bonus.

Reward.	SR	SPL
Binary	29.5	11.1
Min-Max	29.2	12.5
SoftMax	31.3	13.1
Ours	**35.6**	**14.8**

(b) Analysis of the Reward Func.

Temperature	SR	SPL
T = 0.2	33.2	13.3
T = 0.5	**35.6**	**14.8**
T = 0.8	33.7	13.9

(c) Efficacy of the Temperature.

Analysis of the Gap-Aware hybrid Reward Function. The quantitative results in Table 4b show that our Reward significantly outperforms others that only consider absolute correctness or relative distance. A deeper analysis, shown in Figure 5, reveals the principles behind its effectiveness.Our reward function is designed to be adaptive: it should provide a strong, decisive signal in high-certainty scenarios while remaining nuanced to encourage exploration in low-certainty ones. Figure 5(left) visualizes the reward gap across different hyperparameters, our chosen parameters(T=0.5, B=1.0) achieve a strong gap in high-certainty cases while maintaining a moderate one in low-certainty cases, and as show in Table 4a and 4c, this combination also achieved the best results.

Furthermore, the training dynamics in Figure 5(right) illustrate a critical insight. While the Binary and Min-Max reward models achieve deceptively high scores during training, this merely indicates success at a simplistic proxy task: perfectly mimicking the single best action. In contrast, our Gap-Aware hybrid Reward, though yielding a lower absolute score, provides a more meaningful signal by teaching the model to evaluate all options. This fosters a more generalizable reasoning capability and demonstrates that a reward function's alignment with the true objective of robust navigation is more critical than the raw magnitude of its score.

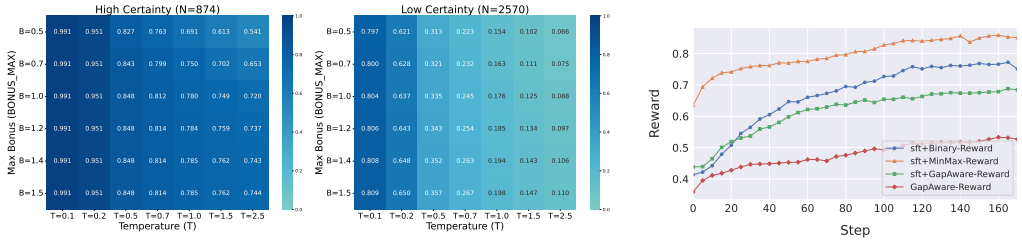


Figure 5: Left heatmaps showing the reward gap between the best and second-best actions under High and Low Certainty scenarios. Right training reward curves for different reward functions.

6 Conclusion

In this work, we present CompassNav, a framework that successfully moves beyond the prevailing paradigm of *Path Imitation* towards a more robust *Decision Understanding* approach in navigation. To enable this shift, we make two core contributions: the Compass-Data-22k dataset, which offers panoramic, per-action supervision far exceeding single-path imitation, and a novel Gap-Aware Hybrid Reward function that provides adaptive feedback tailored to decision certainty.Integrated into an SFT-then-RFT recipe, our framework transforms a 7B parameter LVLM into an expert navigator that establishes a new SOTA. It not only outperforms even larger proprietary models in simulation but also demonstrates robust performance in real-world deployment. Our findings paving the way for future research in low-cost, intelligent embodied agents.

References

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

Al-Halah, Z., Ramakrishnan, S. K., and Grauman, K. (2022). Zero experience required: Plug & play modular transfer learning for semantic visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17031–17041.

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and Van Den Hengel, A. (2018). Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683.

Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., et al. (2025). Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Cao, Y., Zhang, J., Yu, Z., Liu, S., Qin, Z., Zou, Q., Du, B., and Xu, K. (2024). Cognav: Cognitive process modeling for object goal navigation with llms. *arXiv preprint arXiv:2412.10439*.

Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., and Zhang, Y. (2017). Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*.

Chaplot, D. S., Gandhi, D., Gupta, A., and Salakhutdinov, R. (2020). Object goal navigation using goal-oriented semantic exploration. In *In Neural Information Processing Systems*.

Comanici, G., Bieber, E., Schaekermann, M., Pasupat, I., Sachdeva, N., Dhillon, I., Blistein, M., Ram, O., Zhang, D., Rosen, E., et al. (2025). Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. (2024). The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Gao, C., Jin, L., Peng, X., Zhang, J., Deng, Y., Li, A., Wang, H., and Liu, S. (2025). Octonav: Towards generalist embodied navigation. *arXiv preprint arXiv:2506.09839*.

Goetting, D., Singh, H. G., and Loquercio, A. (2024). End-to-end navigation with vision language models: Transforming spatial reasoning into question-answering. *arXiv preprint arXiv:2411.05755*.

Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al. (2024). Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Krantz, J., Lee, S., Malik, J., Batra, D., and Chaplot, D. S. (2022). Instance-specific image goal navigation: Training embodied agents to find object instances. *arXiv preprint arXiv:2211.15876*.

Krantz, J., Wijmans, E., Majumdar, A., Batra, D., and Lee, S. (2020). Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision*, pages 104–120. Springer.

Ku, A., Anderson, P., Patel, R., Ie, E., and Baldridge, J. (2020). Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. *arXiv preprint arXiv:2010.07954*.

Kwon, M., Xie, S. M., Bullard, K., and Sadigh, D. (2023). Reward design with language models.

Li, P., Wu, K., Fu, J., and Zhou, S. (2025). Regnav: Room expert guided image-goal navigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 4860–4868.

Liu, Q., Huang, T., Zhang, Z., and Tang, H. (2025a). Nav-r1: Reasoning and navigation in embodied scenes.

Liu, Z., Sun, Z., Zang, Y., Dong, X., Cao, Y., Duan, H., Lin, D., and Wang, J. (2025b). Visual-rft: Visual reinforcement fine-tuning.

Long, Y., Cai, W., Wang, H., Zhan, G., and Dong, H. (2024). Instructnav: Zero-shot system for generic instruction navigation in unexplored environment. *arXiv preprint arXiv:2406.04882*.

Mezghan, L., Sukhbaatar, S., Lavril, T., Maksymets, O., Batra, D., Bojanowski, P., and Alahari, K. (2022). Memory-augmented reinforcement learning for image-goal navigation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3316–3323. IEEE.

Nasiriany, S., Xia, F., Yu, W., Xiao, T., Liang, J., Dasgupta, I., Xie, A., Driess, D., Wahid, A., Xu, Z., et al. (2024). Pivot: Iterative visual prompting elicits actionable knowledge for vlms. *arXiv preprint arXiv:2402.07872*.

OpenAI (2025). Openai o3 and o4-mini system card. https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. (2022). Training language models to follow instructions with human feedback.

Patro, S. and Sahu, K. K. (2015). Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*.

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

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. (2024). Direct preference optimization: Your language model is secretly a reward model.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. (2023). Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Ramakrishnan, S. K., Gokaslan, A., Wijmans, E., Clegg, A., Turner, J. M., Savva, M., Chang, A. X., and Batra, D. (2021a). Habitat-Matterport 3D Dataset (HM3D): 1000 large-scale 3D environments for embodied AI. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16203–16213.

Ramakrishnan, S. K., Gokaslan, A., Wijmans, E., Maksymets, O., Clegg, A., Turner, J., Undersander, E., Galuba, W., Westbury, A., Chang, A. X., et al. (2021b). Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *arXiv preprint arXiv:2109.08238*.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. (2024). Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Sun, H., Wu, J., Xia, B., Luo, Y., Zhao, Y., Qin, K., Lv, X., Zhang, T., Chang, Y., and Wang, X. (2025). Reinforcement fine-tuning powers reasoning capability of multimodal large language models. *arXiv preprint arXiv:2505.18536*.

Team, G., Georgiev, P., Lei, V. I., Burnell, R., Bai, L., Gulati, A., Tanzer, G., Vincent, D., Pan, Z., Wang, S., et al. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Team, Q. (2024). Qvq: To see the world with wisdom.

Wang, P., Bai, S., Tan, S., Wang, S., Fan, Z., Bai, J., Chen, K., Liu, X., Wang, J., Ge, W., et al. (2024). Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Yadav, K., Ramrakhya, R., Ramakrishnan, S. K., Gervet, T., Turner, J., Gokaslan, A., Maestre, N., Chang, A. X., Batra, D., Savva, M., et al. (2023). Habitat-matterport 3d semantics dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4927–4936.

Yang, W., Chen, J., Xin, X., Zhou, S., Hu, B., Feng, Y., Chen, C., and Wang, C. (2024). Psl: Rethinking and improving softmax loss from pairwise perspective for recommendation. *Advances in Neural Information Processing Systems*, 37:120974–121006.

Yin, H., Xu, X., Wu, Z., Zhou, J., and Lu, J. (2024). Sg-nav: Online 3d scene graph prompting for llm-based zero-shot object navigation.

Yin, H., Xu, X., Zhao, L., Wang, Z., Zhou, J., and Lu, J. (2025). Unigoal: Towards universal zero-shot goal-oriented navigation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19057–19066.

Yokoyama, N., Ha, S., Batra, D., Wang, J., and Bucher, B. (2024a). Vlfm: Vision-language frontier maps for zero-shot semantic navigation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 42–48. IEEE.

Yokoyama, N., Ramrakhya, R., Das, A., Batra, D., and Ha, S. (2024b). Hm3d-ovon: A dataset and benchmark for open-vocabulary object goal navigation. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5543–5550. IEEE.

Zhang, J., Wang, K., Wang, S., Li, M., Liu, H., Wei, S., Wang, Z., Zhang, Z., and Wang, H. (2024). Uni-navid: A video-based vision-language-action model for unifying embodied navigation tasks. *arXiv preprint arXiv:2412.06224*.

Zhang, L., Liu, Y., Zhang, Z., Aghaei, M., Hu, Y., Gu, H., Alomrani, M. A., Bravo, D. G. A., Karimi, R., Hamidizadeh, A., et al. (2025a). Mem2ego: Empowering vision-language models with global-to-ego memory for long-horizon embodied navigation. *arXiv preprint arXiv:2502.14254*.

Zhang, W., Bei, Y., Yang, L., Zou, H. P., Zhou, P., Liu, A., Li, Y., Chen, H., Wang, J., Wang, Y., et al. (2025b). Cold-start recommendation towards the era of large language models (llms): A comprehensive survey and roadmap. *arXiv preprint arXiv:2501.01945*.

Zhou, K., Zheng, K., Pryor, C., Shen, Y., Jin, H., Getoor, L., and Wang, X. E. (2023). Esc: Exploration with soft commonsense constraints for zero-shot object navigation. In *International Conference on Machine Learning*, pages 42829–42842. PMLR.

Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., and Farhadi, A. (2017). Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3357–3364. IEEE.

Zhu, Z., Wang, X., Li, Y., Zhang, Z., Ma, X., Chen, Y., Jia, B., Liang, W., Yu, Q., Deng, Z., Huang, S., and Li, Q. (2025). Move to understand a 3d scene: Bridging visual grounding and exploration for efficient and versatile embodied navigation.

A More related work and More Experimetns Results

A.1 Comparison of Different Embodied Navigation Paradigms

Three main paradigms exist for embodied navigation. In Vision-and-Language Navigation (VLN) (Anderson et al., 2018; Krantz et al., 2020), an agent follows dense, step-by-step natural language instructions to complete a trajectory. In contrast, Modular Goal Navigation (Yin et al., 2025; 2024; Long et al., 2024; Yang et al., 2024; Zhou et al., 2023) employs a complex system that decouples functionalities like perception, planning, and action into a multi-stage pipeline. Our End-to-End Goal Navigation (Goetting et al., 2024; Nasiriany et al., 2024) introduces a more streamlined approach, where a single, unified LVLM policy directly maps a sparse goal and visual input to an action via internal spatial reasoning. An intuitive comparison of these concepts is visualized in Figure 6.

A.2 Relationship to Deep Reinforcement Learning for Navigation

Deep Reinforcement Learning (DRL) has long been a primary approach for robot navigation. Traditional methods typically employ small-parameter, specialized models trained via online RL, where an agent directly interacts with a simulator to learn from environmental feedback (Zhu et al., 2017;

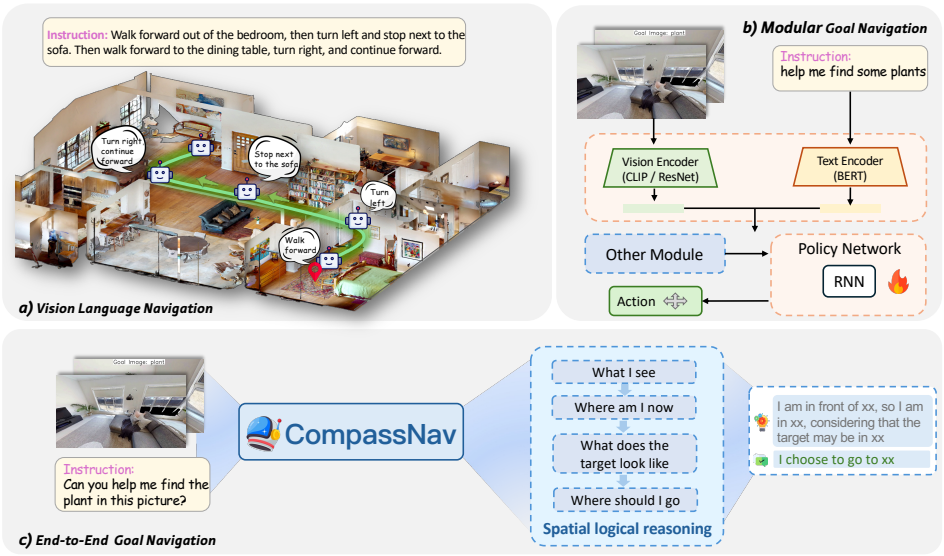


Figure 6: **A conceptual comparison of embodied navigation paradigms.**

Mezghan et al., 2022; Yang et al., 2024). While this online interaction provides access to fine-grained, per-action rewards, the resulting policies are known to suffer from poor generalization, often failing in visually distinct, unseen environments (Al-Halah et al., 2022).

Large Vision-Language Models (LVLMs) offer a promising alternative, possessing strong prior knowledge and superior generalization. However, the immense scale of these models makes the conventional online RL paradigm computationally infeasible; frameworks for efficiently updating billions of parameters through real-time interaction do not yet exist. Our work addresses this gap with a pragmatic offline paradigm. We effectively pre-collect a large-scale dataset of simulated "interaction data" (`Compass-Data-22k`), where every feasible action is annotated with an oracle reward. By training on this rich, offline data, our approach approximates the fine-grained feedback of online RL while harnessing the powerful generalization of LVLMs, thus bridging a critical gap between the two methodologies.

A.3 More experimental results

Based on current similar work, Nav-R1 was tested on other obejct navigation benchmarks HM3D-OVON (Yokoyama et al., 2024b). We also conducted tests with the same setting on the val unseen split of this benchmark. The results are shown in the table 5. Our method only used one tenth of their training data to train the original Qwen2.5-VL-7B, which still exceeded their model trained from 3D-R1. This further demonstrates the superiority of our method.

Table 5: Obejct Navigation results on HM3D-OVON val-unsenn

Method	Val-Unseen	
	SR	SPL
Uni-NaVid (Zhang et al., 2024)	39.5	19.8
MTU3D (Zhu et al., 2025)	40.8	12.1
Nav-R1 (Liu et al., 2025a)	42.2	20.1
CompassNav(ours)	**43.5**	**21.6**

B Standard Benchmarks and Tasks

Scene Datasets. Our experiments are primarily conducted on two large-scale, standard indoor navigation scene datasets: First, Habitat-Matterport 3D (HM3D) (Ramakrishnan et al., 2021a) is a large-scale dataset of 3D indoor environments, consisting of 1,000 high-resolution scans of residential and commercial spaces. Our work utilizes the version split for the Habitat Challenge, with the `train`

split comprising 800 scenes and the `val` split containing 100 scenes. We report results on two semantic annotation versions: HM3Dv1 (Habitat Challenge 2022, using HM3D-Semantics-v0.1 (Ramakrishnan et al., 2021b)) and HM3Dv2 (Habitat Challenge 2023, using HM3D-Semantics-v0.2 (Yadav et al., 2023)).Seconde, Matterport3D (MP3D) (Chang et al., 2017) is another large-scale RGB-D dataset featuring 90 building-scale scenes, composed of 10,800 panoramic views from 194,400 images.

Task Datasets. We evaluate our method on several goal-driven navigation task datasets: ObjectNav (ON) (Chaplot et al., 2020) requires agents to find an object of a given category. We evaluate on both the v1 version (2000 episodes, 20 scenes, 6 categories) and the v2 version (1000 episodes, 36 scenes, 6 categories) from the Habitat Challenge; HM3D-OVON (Yadav et al., 2023) is a large-scale open-vocabulary ObjectNav benchmark that significantly expands the semantic range, featuring over 15k instances across 379 distinct object categories; Instance Image-Goal Nav (IIN) (Krantz et al., 2022) requires agents to find a specific object instance. The goal categories, including "Chair", "Bed", "Toilet", "Couch", "Plant", and "TV", are consistent with the ObjectNav task.

Task Definition. In our work, the agent receives a posed RGB-D video stream and must execute a new action $a \in \mathcal{A}$ upon receiving a new observation. A key distinction exists between our two main tasks: for **IIN**, the goal is a specific object instance provided as a goal image; for **ON**, the goal is an object category provided as text.

Our action space $\mathcal{A}$ consists of a discrete set of high-level actions represented as arrows rendered directly onto the agent's observation. A low-level controller translates the chosen arrow into a sequence of primitive simulator actions: `move_forward` (25cm), `turn_left` (30°), and `turn_right` (30°). An episode is considered successful if the agent executes the `stop` action when it is within a predefined radius 1-meter of the goal, in less than a maximum of 500 steps.

To handle the `stop` action, we are inspired by multi-agent collaborative frameworks. Instead of cluttering the primary agent's visual input with an additional stop command overlay, we employ a dedicated Stop Agent. This agent, a non-fine-tuned Qwen2.5-VL-7B (Bai et al., 2025) model, is prompted at each step to decide whether the goal is sufficiently in view to terminate the episode. This decouples the navigation and stopping decisions, simplifying the task for the primary agent. The prompt for the Stop Agent please refer D.2

C Data Generation Pipeline

C.1 Action Proposal Module (APM)

Following VLMNav (Goetting et al., 2024), we used the Action Proposal Module (APM), a critical component that translates raw depth sensor data into a discrete set of high-level, human-interpretable candidate actions for the LVLM to reason about. The process is designed to ensure actions are safe, visually distinct, and biased towards efficient exploration.

First, at each timestep, the APM leverages the depth image to compute the traversable area within the agent's field of view. Based on the furthest navigable distance for each angle θ, an initial, dense set of candidate actions, A_{initial}, is generated. Simultaneously, we construct a 2D voxel map using accumulated depth and pose information, marking all observed areas within a 2-meter radius as explored.

To encourage systematic exploration and prevent redundant movements, we filter A_{initial} based on the exploration map. For each action, we define an exploration indicator variable e_i:

$$e_i = \begin{cases} 1 & \text{if its destination region is unexplored} \\ 0 & \text{if its destination region is explored} \end{cases} \tag{7}$$

To build the final action set, A_{final}, we prioritize actions leading to new territory while ensuring sufficient visual spacing for the VLM to discern between options. This is achieved in a two-step filtering process. First, we greedily add unexplored actions ($e_i = 1$) that maintain a minimum angular spacing of θ_δ:

$$A_{\text{final}} \leftarrow A_{\text{final}} \cup \{(\theta_i, r_i) \mid e_i = 1 \text{ and } |\theta_i - \theta_j| \geq \theta_\delta, \forall (\theta_j, r_j) \in A_{\text{final}}\} \tag{8}$$

Figure 7: Action Proposal Module(APM) module workflow diagram

To ensure the agent can still navigate within known areas and cover all directions, we then supplement A_{final} with explored actions ($e_i = 0$), subject to a larger angular spacing $\theta_\Delta > \theta_\delta$:

$$A_{\text{final}} \leftarrow A_{\text{final}} \cup \{(\theta_i, r_i) \mid e_i = 0 \text{ and } |\theta_i - \theta_j| \geq \theta_\Delta, \forall (\theta_j, r_j) \in A_{\text{final}}\} \tag{9}$$

For safety, the radius of each action is clipped via $r_i \leftarrow \min\left(\frac{2}{3} \cdot r_i, r_{\max}\right)$. If no navigable actions are found, a 180° turn action is added to handle edge cases where the agent might be stuck. The final set of actions is then rendered as arrows onto the agent's RGB observation(Figure 7).

To bridge the gap between the LVLM's high-level decision-making and the simulator's low-level execution, we implement a deterministic low-level controller. This controller translates the agent's chosen high-level action, parameterized as a polar coordinate tuple (r, θ), into a discrete sequence of primitive actions executable within the Habitat simulator. The primitive action space consists of `move_forward(0.25m)`, `turn_left(30°)`, and `turn_right(30°)`.

The translation process follows a standard two-phase "turn-then-move" sequence to ensure predictable navigation outcomes:

1. **Rotation Phase:** The controller first addresses the angular component, θ. The angle, given in radians, is converted to degrees. The number of required 30° turns is calculated using the ceiling function to ensure the agent turns at least the specified amount. The direction of the turn is determined by the sign of θ: a positive value corresponds to `turn_left`, and a negative value corresponds to `turn_right`.
2. **Translation Phase:** After completing the rotation, the controller handles the distance component, r. The number of 0.25m forward steps is similarly calculated using the ceiling function. The agent then executes the `move_forward` action for the calculated number of steps.

This entire sequence of primitive actions is then executed by the simulator to complete the high-level action. Algorithm 1 provides a formal description of this process.

C.2 Data Collection and Filtering

To generate our dataset, we employ a GenData Agent within the Habitat simulator. At each step, the agent receives an RGB-D observation, which is processed by the APM to generate candidate actions. For each action (r, θ), its global landing coordinates are calculated based on the agent's current pose. An oracle A* planner then computes the geodesic distance between each landing point and the goal's global coordinates.

The GenData Agent proceeds by selecting the action with the shortest distance. To collect a rich set of diverse yet effective trajectories, we implement a backtracking mechanism: if multiple actions have nearly equal A* distances, or if the decision certainty is below a threshold of 0.1, the agent records the current state (pose and candidate actions). After the current episode concludes, the GenData Agent returns to these recorded states to execute alternative choices, thus exploring and recording other viable paths to the same goal.

While this pipeline could theoretically generate a very large dataset, we prioritize data quality and training efficiency, aligning with the "small data, more epochs" characteristic of RFT. We collected an initial set of 54k trajectories. This set was then rigorously filtered. We first used CLIP to discard trajectories where the goal image was blurry or semantically unclear (an occasional artifact of the simulator). Subsequently, we applied rule-based filtering to remove episodes where the agent was

Algorithm 1 High-level to Low-level Action Translation

Input: High-level action $A_{high} = (r, \theta)$
Output: A sequence of low-level actions S_{low}

procedure TRANSLATEACTION(A_{high})
    $S_{low} \leftarrow []$ ▷ Initialize an empty sequence
    ▷ *Phase 1: Rotation*
    $total\_degrees \leftarrow \text{abs}(\theta \times 180/\pi)$
    $num\_turns \leftarrow \text{ceil}(total\_degrees/30)$
    if $\theta > 0$ **then**
        $turn\_action \leftarrow$ TURN_LEFT
    else if $\theta < 0$ **then**
        $turn\_action \leftarrow$ TURN_RIGHT
    else
        $num\_turns \leftarrow 0$
    end if
    for $i = 1$ **to** num_turns **do**
        Append $turn\_action$ to S_{low}
    end for
    ▷ *Phase 2: Translation*
    $num\_forwards \leftarrow \text{ceil}(r/0.25)$
    for $i = 1$ **to** $num\_forwards$ **do**
        Append MOVE_FORWARD to S_{low}
    end for

    return S_{low}
end procedure

stuck in repetitive loops (e.g., constant turning). The final dataset consists of approximately 10k high-quality trajectories, a scale consistent with datasets commonly used for GRPO-based tasks.

Challenges in Dense Annotation Generation. This data generation process, although conceptually simple, faces high time costs. The computational cost of performing an A* search for *every* candidate at *every* timestep across thousands of trajectories is substantial. Our pipeline implements a highly parallelized, batched query system to the A* planner, making the generation of our large-scale corpus computationally tractable.

D Dataset Structure and Prompts

D.1 DATASET STRUCTURE

Our training corpus is composed of two distinct sets with differing distributions. The Supervised Fine-Tuning (SFT) set is distributed relatively evenly across six categories: chair (N=2,884), toilet (N=2,203), bed (N=1,697), tv screen (N=1,509), sofa (N=1,500), and plant (N=1,300).

In contrast, our Reinforcement Fine-Tuning (RFT) set is intentionally curated to reflect the natural long-tail distribution of objects found in the HM3D environment. This results in a deliberately imbalanced training set: chair (N=5,697), tv screen (N=2,790), plant (N=1,946), and toilet (N=1,100). This imbalance is not an artifact of our design but a characteristic of the benchmark itself, which is evident in the official validation set for Instance ImageNav (IIN) that exhibits a similar skew: chair (510), plant (155), bed (113), tv monitor (85), sofa (80), and toilet (57).

Given this inherent long-tail nature, evaluating on a similarly skewed test set could fail to penalize models that simply overfit to the most frequent categories. Therefore, to provide a more rigorous and unbiased assessment of our model's generalization capabilities, we conduct our primary evaluation on the ObjectNav validation set, which features a significantly more balanced distribution: chair (195), sofa (187), toilet (166), bed (165), plant (152), and tv monitor (135). This allows for a stricter test of the agent's ability to succeed across both common and rare object categories.

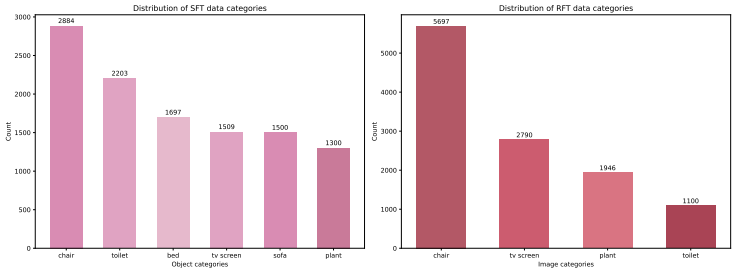


Figure 8: Category details of targets in SFT and RFT datasets

D.2 PROMPTS

The following tables show the prompt templates used for the model in various navigation tasks.

Table 6: Prompt for CompassNav in the Object Navigation Task.

SYSTEM PROMPT:

You are an embodied robotic assistant, with an RGB image sensor. You observe the image and instructions. Given to you and output a textual response, which is converted into actions that physically move you. Within the environment. You cannot move through closed doors. "

USER PROMPT:
TASK: navigate to the nearest {goal.upper()}, and get as close to it as possible. Use your prior knowledge about where items are typically located within a home.
Current observation has $\{num_{actions} - 1\}$ potential actions shown as red arrows labeled with numbers in white circles. NOTE: choose action 0 if you want to turn around or dont see any good actions.
Critical Decision Framework:

1. Strategic decision (compose your plan internally, then select one action)
   - Where am I now(pose/heading)? What do I currently see (salient objects/structures/cues for goal.upper())
   - For each visible action, where will it likely take me next?"
   - For each visible action, estimate probability of encountering goal.upper() soon (based on priors, observed cues, spatial continuity, and last-seen evidence)
2. Action Selection
   - Output the all thinking process in <think></think> and the final answer in <answer></answer> tags.
   - Please strictly follow the format. and match potential actions to strategic decision.
3. CONSTRAINTS: Can go up and down stairs, but no closed doors.

Table 7: Prompt for Stop Agent in the Object Navigation Task.

SYSTEM PROMPT:

You are an expert visual analysis agent for a home navigation robot. Your mission is to determine if the robot has successfully reached its target object and should stop.The target object is: {goal.upper()}.

USER PROMPT:
The robot should stop ONLY if it is within 1 meter of the target object. This ensures it is close enough for any potential interaction. Additionally, it must be directly in front of the CORRECT object with a clear, unobstructed view.

CRITICAL INSTRUCTIONS

- Distinguish the target from visually similar objects. A CHAIR is not a SOFA, a SOFA is not a BED.
- Follow the reasoning steps strictly inside <think> tags.
- Provide your final command ONLY inside <answer> tags.
- Do NOT output anything outside <think> and <answer> tags.

Follow these steps in your reasoning process inside the <think> tags:

1. Identify Goal: State the target object you are looking for.
2. Scene Analysis: Briefly describe the main objects you see in the image.
3. Target Verification: Is the target object {goal.upper()} present in the image? Verify its identity against similar objects and state your confidence.
4. Proximity and Distance Assessment: If the target is present, you must evaluate if the robot is within 1 meter of it. Use visual cues to estimate this distance: Does the object appear very large and fill a significant portion of the image? Are fine details, like fabric texture or wood grain, clearly visible? Is the view completely unobstructed? The robot should only stop if it is confirmed to be within this 1-meter range.
5. Final Conclusion: Based on all the above, make a final decision on whether to stop or continue, and provide a brief justification.

Finaly, final Command (inside <answer>): Output ONLY 1 if the robot should STOP.Output ONLY 0 if the robot should CONTINUE.Do NOT output any other characters or words.

EXAMPLE: <think> 1.Identify Goal:The target is a SOFA. 2.Scene Analysis: I see a living room with a large grey sectional sofa, a coffee table, and a TV stand in the distance. 3.Target Verification: The large grey object is definitely a SOFA, not an armchair or a bed. I am highly confident. 4.Proximity and Distance Assessment: The sofa is very large in the frame,The fabric texture is clearly visible. Based on these cues, I estimate the robot is well within 1 meter of the sofa. The view is unobstructed. This is the correct stopping position. 5.Final Conclusion: The robot has successfully reached the SOFA and is positioned correctly within the 1-meter requirement. It should stop. </think> <answer>1 </answer>

Table 8: Prompt for CompassNav in the Instance Image-goal Navigation Task.

SYSTEM PROMPT:

You are an embodied robotic assistant, with an RGB image sensor. You observe the image and instructions. Given to you and output a textual response, which is converted into actions that physically move you. Within the environment. You cannot move through closed doors. "

USER PROMPT:
You can see a composite image composed of two independent images pieced together, separated by a white border:
The RIGHT image, labeled 'GOAL IMAGE', shows the TARGET OBJECT (the {goal['name']}) you need to find.

The LEFT image represents your CURRENT VIEW (what you are actually seeing now). Current observation has $\{num_{actions} - 1\}$ potential actions shown as red arrows labeled with numbers in white circles.Note: choose action 0 if you want to turn around or don't see any good actions, each arrow represents a potential action and is labeled with a number in a white circle indicating the location you would move to if you took that action. TASK:

1. Observe the right part. Identify the {goal['name']}) by focusing on intrinsic attributes like shape, color, and material, ignoring the environment surrounding the object.
2. Use your prior knowledge about where items are typically located within a home."
3. Describe what you see and any clues that could help you find the {goal['name']}).
4. Next, decide which direction you should go in.
5. You cannot go through closed doors. You cannot go up or down staircases. Ensure that the object being searched for is on the same floor.

Finally, choose the best action as your final answer.Output the thinking process in <think></think> and the final answer in <answer></answer> tags.

Table 9: Prompt for Stop Agent in the Instance Image-goal Navigation Task.

SYSTEM PROMPT:

You are an expert visual analysis agent for a home navigation robot. Your mission is to determine if your current view precisely matches the goal view, and if you should stop.

USER PROMPT:
You are given a composite image made of two parts:The LEFT image is your CURRENT EGO-VIEW.The RIGHT image is the GOAL IMAGE, showing the specific object instance you must find.You should command the robot to STOP only if the CURRENT VIEW contains the exact object from the GOAL IMAGE, and you are in an ideal position for interaction. This means you are directly in front of, and very close to, the target, with a clear, unobstructed view that matches the perspective of the GOAL IMAGE as closely as possible. CRITICAL INSTRUCTIONS

- The goal is a SPECIFIC INSTANCE, not just a category. A different chair, even of the same type, is NOT the target.
- Pay close attention to fine-grained details: color, shape, texture, and surrounding context to verify the instance match.
- Follow the reasoning steps strictly inside <think> tags.Provide your final command ONLY inside <answer> tags.Do NOT output anything outside <think> and <answer> tags.

Follow these steps in your reasoning process inside the <think> tags:

1. Analyze Goal Image (RIGHT): Briefly describe the target object instance in the GOAL IMAGE, noting its key visual features (e.g., 'a wooden armchair with blue cushions').
2. Analyze Current View (LEFT): Briefly describe the main objects you see in your CURRENT VIEW.
3. Instance Verification: Compare the objects in the CURRENT VIEW to the GOAL IMAGE. Is the specific target instance present? Justify your conclusion by comparing details like color, model, and wear/tear. State your confidence.
4. Proximity and Distance Assessment: If the target instance is present, evaluate your proximity and pose. Does the current view of the object match the scale and perspective of the goal image? Are you positioned for direct interaction?
5. Final Conclusion: Based on all the above, make a final decision on whether to stop or continue, and provide a brief justification.

Finaly, final Command (inside <answer>): Output ONLY 1 if the robot should STOP.Output ONLY 0 if the robot should CONTINUE.Do NOT output any other characters or words.

EXAMPLE: <think> 1.Analyze Goal Image (RIGHT): The goal image shows a specific wooden dining chair with a distinctive high back and a circular, light-colored seat cushion. 2.Analyze Current View (LEFT): I see a dining area in my current view. There are several chairs around a table. One of them is a high-backed wooden chair. 3.Instance Verification: Comparing the chair in the current view to the goal image, the wood grain, the high-back design, and the specific circular cushion match perfectly. It is the correct instance. I am highly confident. 4.Proximity and Pose Assessment: The chair in the current view is very close, taking up a large portion of the frame, similar to the goal image. I am facing it directly. The pose is ideal for stopping. 5.Final Conclusion: I have successfully reached the correct chair instance and am positioned correctly. I should command the robot to stop. </think> <answer>1 </answer>

E Model and Training Details

E.1 Training Configurations

Supervised Fine-Tuning (SFT) Details We conduct the Supervised Fine-Tuning (SFT) stage using the LLaMA-Factory framework. The training process was performed on 8 A800 GPUs and took approximately 12 hours to complete. The key configurations and hyperparameters are summarized below.We initialize our model from the `Qwen2.5-VL-7B-Instruct` checkpoint. The task is set up as a standard supervised fine-tuning problem, with a maximum sequence length (`cutoff_len`) of 4096 tokens.

Fine-tuning Strategy. We employ Parameter-Efficient Fine-Tuning (PEFT) using the LoRA method to reduce computational overhead. The LoRA modules are applied to all projection layers (`q_proj, v_proj, k_proj, o_proj`) as well as the feed-forward layers (`gate_proj, up_proj, down_proj`). Key LoRA hyperparameters include a rank (`lora_rank`) of 64, an alpha (`lora_alpha`) of 128, and a dropout rate of 0.05.

Training and Optimization. The model is trained for 3 epochs with a global batch size of 128 (8 GPUs $\times$ 8 per_device_batch_size $\times$ 2 gradient_accumulation_steps). We use the AdamW optimizer with a cosine learning rate scheduler, starting with a learning rate of 2×10^{-4} and a weight decay of 0.01. The warmup ratio is set to 0.03 of the total training steps. For efficiency, we utilize DeepSpeed ZeRO Stage 2, BF16 mixed-precision training, and gradient checkpointing.

Reinforcement Fine-Tuning (RFT) Details We conduct the Reinforcement Fine-Tuning (RFT) stage using EasyR1, a simplified version of the Verl framework. The training was performed on 8 NVIDIA H200 GPUs, with each model being trained for approximately 170 steps. Due to the computational demands of the rollout process, each training step took about 20 minutes.The actor model was initialized from the LoRA weights obtained during the SFT stage. We use the **GRPO** (`adv_estimator: grpo`) algorithm for policy optimization. A KL penalty (`kl_coef`) of 1×10^{-2} is applied between the actor and the frozen reference policy to regularize the policy updates and prevent divergence.

Optimization. We use the AdamW optimizer with a learning rate (`lr`) of 1×10^{-6} and a weight decay of 1×10^{-2}. The global batch size for policy updates was set to 128, with a per-device micro-batch size of 4.

Rollout and Sampling. During the experience generation (rollout) phase, we sample $N = 5$ distinct responses for each prompt from the training set, which is similar to the number of high-level actions generated by APM. The sampling process uses a high temperature of 1.0 and a top-p of 0.99 to encourage exploration. The rollout batch size is set to 512. For evaluation, the sampling is more deterministic, with $N = 1$ and a temperature of 0.5. Our custom Gap-Aware hybrid reward function, implemented as a Python script, is used to score the generated responses.

E.2 Justification for a Memory-Agnostic Training Approach

We acknowledge that sophisticated memory is crucial for embodied navigation. The central debate, however, is not *whether* to use memory, but *how* it should be integrated with large-scale LVLMs. We identify two distinct integration philosophies:

First is the training-integrated memory approach, where historical context is directly "baked into" the model's input and weights during training. The second is the externally-interfaced memory approach, where complex memory systems (e.g., a SLAM-generated topological map) (Yin et al., 2025; Cao et al., 2024) operate as external modules that the agent can query at inference time. We argue that for powerful, pre-trained LVLMs, the latter is a more promising and architecturally sound direction. Complex memory systems like SLAM excel at providing structured, metric/topological knowledge, but there is currently no effective methodology to distill such a complex system into the weights of an LVLM via end-to-end training. They are best utilized as supplementary, external tools.

Based on this perspective, we focused our empirical validation on the first philosophy: training-integrated memory. We implemented and tested the two most popular paradigms currently used in the E2E navigation literature for this purpose: summarizing history as text (a-la OctoNav (Gao et al., 2025)) and concatenating historical images (a-la VLN-R1 (Qi et al., 2025)). As shown in Table 10, our experiments consistently found that both of these mainstream methods for integrating memory into the training process lead to a degradation in performance.

Table 10: Ablation studies on the impact of including historical information in the input prompt. using both the Qwen2.5-VL-7B and o4-mini models for the Object-Goal Navigation task on HM3Dv2

Model	Method	SR	SPL
Qwen2.5-VL-7B	No history	45.3	17.5
Qwen2.5-VL-7B	With history - Text	45.0↓0.3	13.5↓4.0
Qwen2.5-VL-7B	With history - Image	34.7↓10.6	10.9↓6.6

Model	Method	SR	SPL
o4-mini	No history	59.6	26.9
o4-mini	With history - Text	54.1↓5.5	26.3↓0.6

We attribute this failure to fundamental flaws in how these methods handle historical context. For text summaries, the process of transcribing visual history into language is inherently lossy. It forces rich, high-dimensional visual information through the narrow bottleneck of text, inevitably simplifying or omitting crucial details. Furthermore, any errors or biases from the VLM performing the transcription are then propagated forward, potentially providing the agent with a distorted or misleading memory. Conversely, for concatenated images, the challenge becomes one of temporal confusion. We hypothesize the model struggles to consistently distinguish the current, actionable observation from the passive historical frames, sometimes misinterpreting a past view as its present reality. This spatiotemporal misalignment can lead to critical errors in self-localization and immediate planning.

Therefore, our decision to adopt a memory-agnostic training approach is a deliberate and principled one. It is not an omission, but a rejection of the current, ineffective paradigms for training-integrated memory. By decoupling the core policy from a fixed memory format, we ensure that CompassNav is trained as a pure and robust decision-making core. This modular design makes our framework flexible and future-proof, allowing it to serve as a powerful foundation that can readily integrate with more advanced, externally-interfaced memory solutions as this important research direction evolves.

F Real-World Deployment Considerations

To validate the practical applicability and sim-to-real transfer capabilities of our CompassNav agent, we successfully deployed it on a physical robotic platform. This section details the hardware, system architecture, and qualitative results from our real-world experiments.

Hardware and System Architecture Our experiments were conducted on the ROSMASTER X3(Figure 9), a versatile mobile robot developed by Yahboom. It is equipped with Mecanum wheels that allow for omnidirectional movement, providing high agility in constrained indoor environments. For perception, the robot utilizes an Orbbec Astra Pro depth camera to capture RGB-D images, which serve as the primary visual input for our model. The robot operates on the ROS 2 (Robot Operating System) framework, facilitating robust communication and control.

Figure 9: ROSMASTER X3

Our 7B model is computationally intensive, so our real-world deployment used a three-part system. This setup consisted of the ROSMASTER X3 robot, a local computer that acted as a relay, and a remote server with an NVIDIA H200 GPU to run the CompassNav model. All three devices communicated over the same Local Area Network (LAN). The process worked in a continuous loop: the robot sent its current camera view to the local computer, which combined the image with the task instruction (e.g., "find the trash can") and forwarded the pair to the server. On the server, our model processed the input and decided on a action command. This command was sent back to the local computer, which then translated it into low-level motor instructions for the robot to execute. This real-time setup allowed our model to act as the robot's brain and guide its navigation.

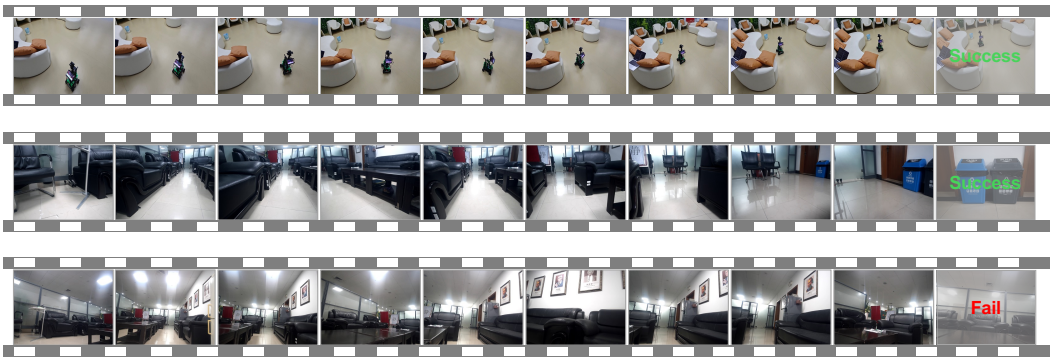


Figure 10: **Qualitative results from real-world deployment.** The first two rows show a successful navigation episode controlled by our CompassNav agent, presented from third-person and first-person perspectives, respectively. The agent successfully navigates a complex office environment to reach the "trash can". The third row shows a comparative trial using the closed-source model GPT-4o in a zero-shot setting.Despite its strong general capabilities, GPT-4o fails the task by colliding with an obstacle midway through the episode.

Qualitative Sim-to-Real Results We conducted several ObjectNav tasks in real-world indoor office environments. Figure 10 presents a compelling comparison of navigation episodes for the task "find the trash can".

As illustrated, our CompassNav agent, demonstrates robust navigation capabilities. The first two rows show that it can effectively process real-world visual information, make sensible decisions to maneuver around furniture, and successfully reach its goal. In contrast, the third row highlights the limitations of a general-purpose model GPT-4o (Hurst et al., 2024) for this embodied task, it struggled with spatial reasoning and obstacle avoidance, leading to task failure. These results strongly suggest that our proposed training framework not only excels in simulation but also produces a robust policy that successfully transfers to the complexities and visual nuances of the real world, validating the effectiveness of our approach.More detailed deployment videos can be viewed in our supporting materials.

G Qualitative Analysis

Thought: The current view shows a living room area with no chairs in sight. I need to find the chair, which is visible in the goal image. The goal image shows a dining table with four chairs around it (brown wooden chairs). The chairs in the living room are either couches or an ottoman—not the target. The chairs are likely off to the right side of the living room near the table in the dining area. I should first move around the couches (left/dark blue couch, ottoman) around the living space and look for an exit or hallway to the right that leads to the dining room. The only listed action that will allow me to re-orient and navigate to that area is action 0 (turn around). Once I turn around, I can locate a path to the right of the living area where the dining room and chairs are likely located.

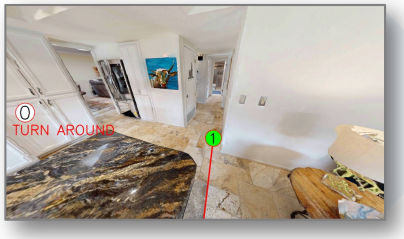
Action: 0

Thought: The goal is to find the chair. In the current view, I see a glass sliding door to the outside, a closed white door slightly to the left, and a blue upholstered armchair just to the right of my viewpoint. I also see a bench in front of me and the interior kitchen beyond. The closed white door would not be eligible. There are five forward arrows: arrows 1 and 2 lead out through the glass door (no chairs outside), arrow 3 goes to the closed door, arrow 5 take me to a hallway, and arrow 4 goes toward the center of the kitchen floor. In order to leave the living room and find the dining table, I choose action 5 to move toward and possibly closer to the chair.

Action: 5

Thought: I am in the kitchen facing an island. On my right (arrow 5) I can move into the open dining area where I see dining chairs arranged around a table. The goal is to find a chair. The chairs I can already see in the right image look like dining chairs and the left arrow 4 leads down a hallway or another room where I have not yet observed any chairs. Therefore the best way to find the chairs visible in the right image is to move into the dining area via arrow 5.

Action: 5



Thought: My current view appears to be in a kitchen, looking towards a central hallway. The goal is to find a dining chair, which is typically located in a dining room or an open-plan dining area. The available actions are "0: TURN AROUND" and "1: MOVE FORWARD". Turning around (Action 0) would likely orient me back into the kitchen, which is a less probable location for a formal dining set. Action 1, moving forward, leads down a main corridor which likely connects to the primary living and dining spaces of the house. To find a new area like a dining room, proceeding along the main thoroughfare is the most logical exploratory step. Therefore, moving forward (Action 1) is the best immediate action to advance towards the goal.

Action: 1

Thought: - I am in a narrow hallway with no chairs in sight. There is two possible forward actions (1 and 2): 1 continues down the hallway toward a larger room, 2 leads into a small doorway (likely a bathroom).The goal image shows a dining table with chairs. These chairs appear wooden, with cushions in dark and lighter shades. Bathrooms typically do not contain the same type of chair. If I turn around, I return to what I likely came from, but chairs could be behind me too, and it's an extra step. Action 1 continues down the hallway toward a possible dining room. Action 2 leads into a small bathroom – unlikely to have the chairs I see in the right image. Thus, going forward via action 1 seems most promising to enter the room with visible chairs.

Action: 1

Figure 11: A qualitative, step-by-step visualization of our CompassNav agent. Task: find the chair in given image.