CorrectNav: Self-Correction Flywheel Empowers Vision-Language-Action Navigation Model

Zhuoyuan Yu[*12], **Yuxing Long**[*12], **Zihan Yang**[12], **Chengyan Zeng**[2], **Hongwei Fan**[12], **Jiyao Zhang**[12], **Hao Dong**[†12]

[1]CFCS, School of Computer Science, Peking University
[2]PKU-Agibot Lab

Abstract

Existing vision-and-language navigation models often deviate from the correct trajectory when executing instructions. However, these models lack effective error correction capability, hindering their recovery from errors. To address this challenge, we propose Self-correction Flywheel, a novel post-training paradigm. Instead of considering the model's error trajectories on the training set as a drawback, our paradigm emphasizes their significance as a valuable data source. We have developed a method to identify deviations in these error trajectories and devised innovative techniques to automatically generate self-correction data for perception and action. These self-correction data serve as fuel to power the model's continued training. The brilliance of our paradigm is revealed when we re-evaluate the model on the training set, uncovering new error trajectories. At this time, the self-correction flywheel begins to spin. Through multiple flywheel iterations, we progressively enhance our monocular RGB-based VLA navigation model CorrectNav. Experiments on R2R-CE and RxR-CE benchmarks show CorrectNav achieves new state-of-the-art success rates of 65.1% and 69.3%, surpassing prior best VLA navigation models by 8.2% and 16.4%. Real robot tests in various indoor and outdoor environments demonstrate CorrectNav's superior capability of error correction, dynamic obstacle avoidance, and long instruction following.

Introduction

In the Vision-and-Language Navigation (VLN) task, users control the robot to move to desired locations in unexplored environments via natural language instructions, like "Move forward and turn right into the living room to wait near the sofa." Due to its user-friendly interaction characteristic, VLN becomes a fundamental capability essential to embodied intelligence and attracts widespread research interest. During the navigation process, models inevitably predict wrong movement actions, causing the robot to deviate from the correct path. These deviations often produce misalignment between the environment and the instructions. Taking the above instruction as an example. If the robot directly turns right at the current position rather than moving forward first, it will enter the kitchen and cannot locate the sofa. At this time, the robot easily gets confused about such misalignment and fails to reach the destination.

Existing VLN models mainly focus on enhancing visual perception and multimodal reasoning capabilities by improving feature representation (An et al. 2024; Hong et al. 2023) or increasing training data (Zhang et al. 2024a, 2025). They aim to enable the model to navigate correctly as much as possible in every step. However, the reality turned out to be different from expectations. Only several imperfect stepwise predictions can accumulate significant deviation from the correct path and ultimately cause failure. The absence of self-correction ability makes previous VLN models struggle to recover from mistakes and get back on track when errors occur, which limits their overall navigation performance. This deficiency raises an important question - **Can we teach robots to self-correct errors during navigation?**

For this problem, we analyze what kinds of errors to correct and how to teach the navigation model to correct them. As a Vision-Language-Action (VLA) task, VLN requires the model to dynamically perceive the environment and follow the given instruction to navigate. Errors often come from two sources: misperception of landmarks and misunderstanding of instruction-specified actions. These errors propagate through the decision-making pipeline, adversely impacting movement prediction. Therefore, attention should be directed toward errors stemming from perception and actions. Besides, real-world applications impose time requirements on the model inference, necessitating that self-correction capabilities should be implicitly integrated into the model through training, rather than being achieved by increasing modules or the reasoning process.

Consequently, we propose Self-correction Flywheel, a novel post-training paradigm for navigation. This approach stems from our observation that well-trained navigation models still produce error trajectories when evaluated on the training set. Rather than viewing these errors as mere shortcomings, we regard them as valuable opportunities to enhance the model further. Our Self-correction Flywheel proceeds through the following four steps: **(1)** Evaluating the trained model on its training set to collect error trajectories. **(2)** Then, we design an automatic approach capable of detecting the deviations and pinpointing their exact locations in error trajectories. **(3)** After identifying deviations,

[*]These authors contributed equally.
[†]Corresponding author.
Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

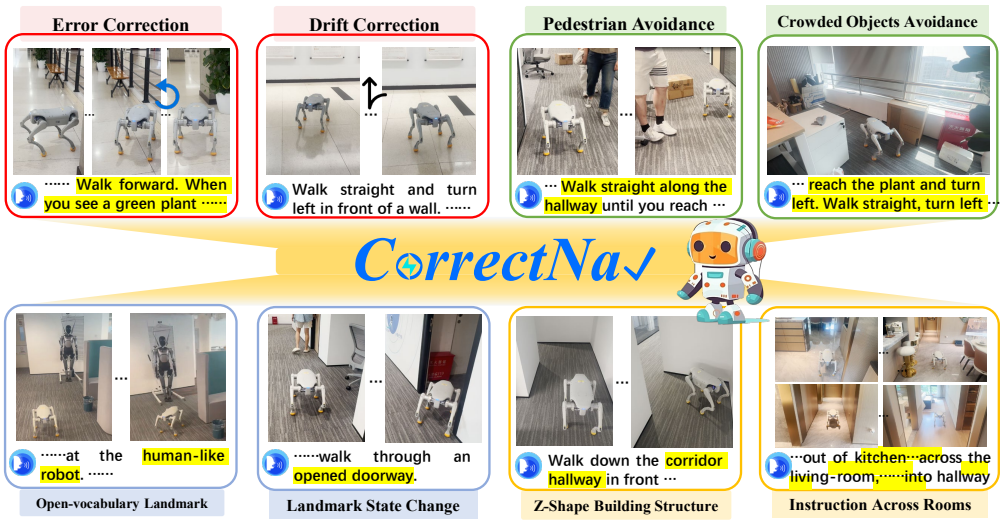

Figure 1: Diverse Capabilities of CorrectNav. The model takes only monocular RGB video and language instructions as inputs, predicting navigation actions. Empowered by the Self-correction Flywheel post-training, CorrectNav not only maintains outstanding multimodal reasoning (Blue), but also displays improved deviation correction (Red), obstacle avoidance (Green), and complex action execution (Yellow).

we create self-correction data from action and perception perspectives. For action correction, we gather trajectories that demonstrate effective recovery from deviations. For perception correction, we leverage large-scale multimodal models to analyze keyframes associated with navigation errors. **(4)** With these self-correction data, we drive the continued training of the navigation model to improve its performance. Completing the above four steps constitutes one round of the Self-correction Flywheel. When we continue to evaluate the model, which has undergone one round of self-correction training, on the training set, a remarkable thing happens. We can identify new error trajectories, thereby generating fresh self-correction data and further training the model. At this time, the Self-correction Flywheel is in motion, and the performance of the navigation model will continuously improve with multiple rounds of training iterations.

Furthermore, we design a suite of navigation fine-tuning strategies, including observation randomization, instruction generation, and general multimodal data recall. Through our proposed fine-tuning and post-training strategies, we develop a new monocular RGB-based VLA navigation model CorrectNav. On VLN-CE benchmarks R2R-CE and RxR-CE, CorrectNav achieves success rates of 65.1% and 69.3%, surpassing previous state-of-the-art models by 8.2% and 16.4%. The real robot tests conducted in diverse indoor and outdoor environments demonstrate that CorrectNav possesses strong capabilities of error correction, dynamic obstacle avoidance, and long instruction following, outperforming existing navigation models.

Related Work

Vision-and-Language Navigation

Vision-and-Language Navigation (VLN) involves an embodied agent navigating to a target location following natural language instructions. Datasets like R2R (Anderson et al. 2018) and RxR (Ku et al. 2020) provide navigation instructions and trajectories in the discretized MP3D (Chang et al. 2017) environment, while VLN-CE (Krantz et al. 2020) adapts these to continuous settings. Current VLN-CE models can be categorized into two groups: topology graph-based approaches, such as BEVbert (An et al. 2023) and ETPnav (An et al. 2024), which rely on multiple sensors to predict waypoints; and models built on pretrained vision-language models (VLMs), including NaVid (Zhang et al. 2024a), Uni-NaVid (Zhang et al. 2025), and NAVILA (Cheng et al. 2024), which infer actions end-to-end according to RGB observation. Existing methods commonly employ techniques such as auxiliary tasks (Zhang et al. 2025, 2024a), instruction augmentation (Wei et al. 2025b), and dataset expansion (Wei et al. 2025a) to enhance performance, but devote less attention to error correction. For easier real robot application, we also construct our CorrectNav based on a pretrained VLM. However, we highlight the value of error correction, which helps us break through the performance bottleneck of current technologies.

Error Correction in Embodied Intelligence

Errors are usually inevitable in embodied intelligence tasks. To enhance the robustness, the ability to correct errors is essential. Error correction methods have been explored in manipulation tasks (Ha, Florence, and Song 2023; Ma et al. 2023; Duan et al. 2024; Liu, Bahety, and Song 2023). However, error correction in navigation tasks is less explored. SmartWay (Shi et al. 2025) uses closed-source large models to reflect on trajectories and decide whether to backtrack, while EnvolveNav (Lin et al. 2025) trains models to generate time-consuming chains of thought with limited improvement. These methods often require additional models or in-

ference steps, reducing efficiency and hindering deployment in the real world. In contrast, our method implicitly teaches error correction through Self-correction Flywheel training, eliminating the need for additional modules or long thinking, thereby facilitating deployment on real robots.

CorrectNav Model

Task Definition

Given a language instruction L_{nav}, the vision-and-language navigation task requires the model to predict the next navigation action $a_{t+1} \in A$ at time step t based on observation $\{O_1, O_2, \ldots, O_t\}$. Recently, to overcome the reliance on multi-sensor, researchers (Zhang et al. 2024a) have simplified observation into a sequence of monocular RGB images $\{I_1, I_2...I_t\}$ captured during navigation.

Algorithm 1: Self-correction Flywheel Post-training

Input: oracle trajectories $\{T_g^{(i)}, L_{nav}^{(i)}\}$, dataset D_{nav}, model $\mathcal{M}$, number of flywheel iteration N, distance threshold S, trajectory planner Γ
Output: Model $\mathcal{M}$
Initialize: $\mathcal{M} \leftarrow \text{Train}(D_{nav}, \mathcal{M})$
for $c = 1$ to N **do**
    $\{T_m^{(i)}\} \leftarrow \mathcal{M}(\{L_{nav}^{(i)}\})$
    Initialize $D_{new} \leftarrow \emptyset$
  for each sample i in the dataset **do**
        $K^{(i)}, T_c^{(i)} \leftarrow \text{DeviDetect}(T_g^{(i)}, T_m^{(i)}, S, \Gamma)$
        $\text{Cap}^{(i)} \leftarrow \text{MLLM\_Description}(K^{(i)})$
        $\text{Qa}^{(i)} \leftarrow \text{MLLM\_QA}(K^{(i)})$
        Add $(T_c^{(i)}, \text{Cap}^{(i)}, \text{Qa}^{(i)})$ to D_{new}
  end for
    $D_{train} \leftarrow \text{Sample}(D_{nav}) \cup \text{Sample}(D_{new})$
    $\mathcal{M} \leftarrow \text{Train}(D_{train}, \mathcal{M})$
end for

Model Structure

Our CorrectNav consists of three modules: the Vision Encoder $v(\cdot)$, the Projector $p(\cdot)$, and the Large Language Model (LLM) $f(\cdot)$. Specifically, we employ SigLIP (Zhai et al. 2023), a 2-layer MLP (Liu et al. 2024), and Qwen2 (Yang et al. 2024). Given an RGB video, the Vision Encoder extracts visual features from the sampled frames, producing $Z_v = v(\{I_1, I_2...I_t\})$. The MLP Projector maps these visual features to the semantic space of the LLM, resulting in a sequence of visual tokens $H_v = p(Z_v)$. Using the visual tokens H_v together with textual tokens X encoded from the task instruction L, the LLM $f(\cdot)$ makes predictions in an auto-regressive manner. Before navigation finetuning, CorrectNav is initialized from LLaVA-Video 7B (Zhang et al. 2024b).

Navigation Fine-tuning

Navigation Action Prediction We collect oracle navigation trajectories from VLN-CE R2R and RxR train splits in MP3D indoor scenes. Each oracle trajectory contains one navigation instruction and step-wise RGB observations and navigation actions $\tau = (L_{nav}, \{(I_t, a_t)\}_{t=1}^{T})$. To enhance visual diversity, we implement a set of domain randomization strategies. These strategies encompass randomizing camera height, adjusting the field of view, varying observation resolution, and altering illumination conditions, as shown in Figure 2. With these strategies, we collected more than 2.1 million step-wise navigation action prediction data D_{nav}, including 527K samples from R2R and 1.58 million samples from RxR. In this task, we take navigation instruction L_{nav} and step-wise RGB observations $\{I_1, I_2...I_t\}$ as CorrectNav's input and require the model to predict an action trunk $\{a_{t+1}, a_{t+2}...a_{t+m}\}$ with m steps.

Trajectory-based Instruction Generation In this task, we collect complete oracle navigation trajectories from VLN-CE R2R and RxR datasets. Among these trajectories, 10K are from R2R and 20K are from RxR. CorrectNav needs to generate language-format navigation instructions based on the monocular RGB observation history. During the training, we input the RGB observations of the whole oracle trajectory $\{I_1, I_2...I_T\}$, and take the corresponding instruction L_{nav} as the target.

General Multimodal Data Recall The format of our downstream navigation task differs significantly from general multimodal training tasks. Only training on the navigation tasks results in general multimodal capability forgetting during the training. To address this, we include a subset of video data from the LLaVA-Video 178K dataset (Zhang et al. 2024b). We focus on Activitynet-QA (Yu et al. 2019) and NextQA (Xiao et al. 2021), which emphasize temporal and spatial scene understanding, aligning with our goals. Therefore, we randomly sample 240K training instances from ActivityQA and NextQA to maintain the model's general multimodal abilities.

Self-correction Flywheel Post-training

To teach the navigation model how to recover from deviations, we propose a new post-training paradigm, Self-correction Flywheel. One iteration of training includes model evaluation, deviation detection, self-correction data creation, and continued training. These four steps can form a closed loop at both ends to create a self-correcting flywheel. Through multiple training iterations, self-correction capabilities can be specifically improved. The overview is introduced in Algorithm 1. Each step will be detailed below.

Step 1 - Model Evaluation on Train Split The training splits of R2R-CE and RxR-CE provide a large number of instructions and oracle trajectory pairs. In the dataset, each oracle trajectory is defined by a sequence of ordered reference points, denoted as $T_g = (G_1, \ldots, G_n)$. During the navigation fine-tuning, we have already used this data to provide step-by-step supervision signals for CorrectNav training. Although the model has been trained on these data, we found that it still makes errors when evaluated on the training set. We realize that this is an excellent source for collecting correction data. The training dataset not only con-

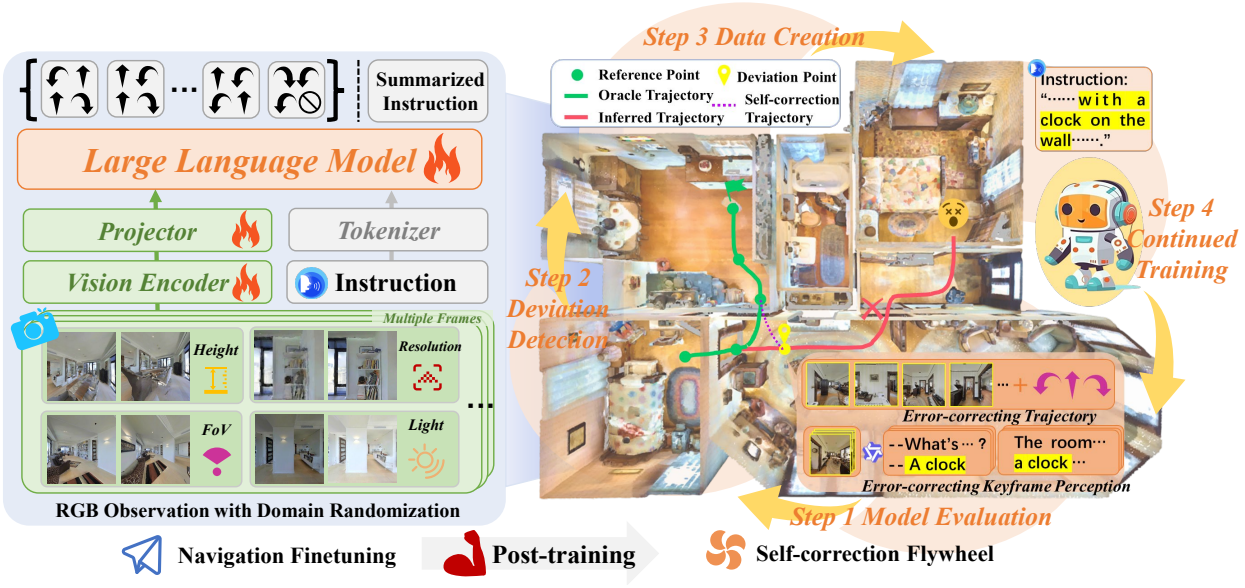

Figure 2: The overview of CorrectNav training. CorrectNav is first finetuned on the navigation tasks (Left), including action prediction and instruction generation. To enhance vision diversity, we implement a suite of domain randomization strategies. Subsequently, CorrectNav is post-trained with our proposed Self-correction Flywheel paradigm (Right). This paradigm operates in a continuous loop of *model evaluation*, *deviation detection*, *data creation*, and *continued training*. Specifically, the data creation part can automatically collect error-correcting trajectory and keyframe perception data. Through multiple training iterations, CorrectNav can learn how to recover from deviations.

tains abundant data but also includes ground truth reference points. Therefore, we collect error trajectories produced during model evaluation on the training set. These trajectories can be denoted by $T_m = (M_1, \ldots, M_m)$, where M_i represents the position of the robot at the i timestep.

Step 2 - Trajectory Deviation Detection Since the collected error trajectories lack annotations indicating where deviations occur, we develop a method to detect such deviations. The key principle is to assess deviations by measuring the distance between the error trajectories and the oracle trajectories. To compute the distance from a robot position M_i to the oracle trajectory T_g, we begin by uniformly interpolating between reference points, which forms an evenly spaced sequence T_g'. For each robot position $M_i \in T_m$, we define the distance from M_i to the T_g as

$$h_i = \min_{x \in T_g'} \|M_i - x\|_2$$

We further define the orthogonal foot of M_i on T_g as

$$P_i = \arg\min_{P \in T_g'} \|M_i - P\|_2.$$

Let S be a predefined threshold. If there exists a timestep t such that

$$h_t > S \quad \text{and} \quad h_i \leq S, \quad \forall i < t, i \in N^*$$

Then we claim that the model begins to deviate from the oracle trajectory at M_t. The observations near timestep t can be marked as keyframes for error correction.

Step 3 - Self-correction Data creation By analyzing deviations in error trajectories, we identify that navigation errors primarily originate from perception and action. Accordingly, we propose self-correction tasks and data creation methods addressing these two aspects.

Error-correcting Trajectory To teach the model how to recover from deviations, we collect error-correcting trajectories based on the detected deviations. Given an oracle trajectory T_g and model trajectory T_m with deviations, we already detect the deviation point M_t and the corresponding orthogonal foot P_t in Step 2. If P_t lies on the segment $\overline{G_k G_{k+1}}$ ($G_k, G_{k+1} \in T_g$), we can know the model has correctly passed through G_k and all previous reference points but deviates slightly while moving towards G_{k+1}. We then utilize a trajectory planner Γ to generate a new trajectory

$$T_e = (M_t, G_{k+1}, \ldots, G_n)$$

This trajectory begins at M_t, passes through the subsequent reference points, and concludes at the destination G_n. Thus, we get an error-correcting trajectory, which can serve as training data for action correction. The training is similar to navigation action prediction. To ensure the model focuses on learning correction behavior, action learning is only performed on the error-correcting trajectory T_e while the trajectory before M_t solely provides observation history.

Keyframe Perception To truly equip CorrectNav with error-correcting ability, we should not only teach it what to act but also why. Errors in the process of vision-language navigation often stem from multimodal perception errors made by the navigation model near the deviation position M_t. To enhance the multimodal perception capabilities of

	Observation				R2R-CE Val-Unseen				RxR-CE Val-Unseen			
	S.RGB	Pano.	Depth	Odo.	NE ↓	OS ↑	SR ↑	SPL ↑	NE ↓	SR ↑	SPL ↑	nDTW ↑
HPN+DN* (Krantz et al. 2021)		✓	✓	✓	6.31	40.0	36.0	34.0	-	-	-	-
CMA* (Hong et al. 2022)		✓	✓	✓	6.20	52.0	41.0	36.0	8.76	26.5	22.1	47.0
VLN⟳BERT* (Hong et al. 2022)		✓	✓	✓	5.74	53.0	44.0	39.0	8.98	27.0	22.6	46.7
Sim2Sim* (Krantz and Lee 2022)		✓	✓	✓	6.07	52.0	43.0	36.0	-	-	-	-
GridMM* (Wang et al. 2023c)		✓	✓	✓	5.11	61.0	49.0	41.0	-	-	-	-
Ego^2-Map* (Hong et al. 2023)		✓	✓	✓	5.54	56.0	47.0	41.0	-	-	-	-
DreamWalker* (Wang et al. 2023a)		✓	✓	✓	5.53	59.0	49.0	44.0	-	-	-	-
Reborn* (An et al. 2022)		✓	✓	✓	5.40	57.0	50.0	46.0	5.98	48.6	42.0	63.3
ETPNav* (An et al. 2024)		✓	✓	✓	4.71	65.0	57.0	49.0	5.64	54.7	44.8	61.9
HNR* (Wang et al. 2024)		✓	✓	✓	4.42	67.0	61.0	51.0	5.50	56.3	46.7	63.5
BEVBert* (An et al. 2023)		✓	✓	✓	4.57	67.0	59.0	50.0	-	-	-	-
HAMT+ScaleVLN* (Wang et al. 2023b)		✓	✓	✓	4.80	-	55.0	51.0	-	-	-	-
AG-CMTP (Chen et al. 2021)		✓	✓	✓	7.90	39.0	23.0	19.0	-	-	-	-
R2R-CMTP (Chen et al. 2021)		✓	✓	✓	7.90	38.0	26.0	22.0	-	-	-	-
InstructNav (Long et al. 2024)		✓	✓	✓	6.89	-	31.0	24.0	-	-	-	-
LAW (Raychaudhuri et al. 2021)	✓		✓	✓	6.83	44.0	35.0	31.0	10.90	8.0	8.0	38.0
CM2 (Georgakis et al. 2022)	✓		✓	✓	7.02	41.0	34.0	27.0	-	-	-	-
WS-MGMap (Chen et al. 2022)	✓		✓	✓	6.28	47.0	38.0	34.0	-	-	-	-
AO-Planner (Chen et al. 2024)		✓	✓		5.55	59.0	47.0	33.0	7.06	43.3	30.5	50.1
Seq2Seq (Krantz et al. 2020)	✓		✓		7.77	37.0	25.0	22.0	12.10	13.9	11.9	30.8
CMA (Krantz et al. 2020)	✓		✓		7.37	40.0	32.0	30.0	-	-	-	-
RGB-Seq2Seq (Krantz et al. 2020)	✓				10.10	8.0	0.0	0.0	-	-	-	-
RGB-CMA (Krantz et al. 2020)	✓				9.55	10.0	5.0	4.0	-	-	-	-
NaVid (Zhang et al. 2024a)	✓				5.47	49.0	37.0	35.0	-	-	-	-
Uni-NaVid (Zhang et al. 2025)	✓				5.58	53.5	47.0	42.7	6.24	48.7	40.9	-
NaVILA (Cheng et al. 2024)	✓				5.22	62.5	54.0	49.0	6.77	49.3	44.0	58.8
StreamVLN (Wei et al. 2025a)	✓				4.98	64.2	56.9	51.9	6.22	52.9	46.0	61.9
CorrectNav (Ours)	✓				**4.24**	**67.5**	**65.1**	**62.3**	**4.09**	**69.3**	**63.3**	**75.2**

Table 1: Comparison with state-of-the-art methods on the Val-Unseen split of R2R-CE (Anderson et al. 2018) and RxR-CE (Ku et al. 2020). * indicates methods using the waypoint predictor from Hong et al. (2022). CorrectNav outperforms all methods that do not rely on simulator pre-trained waypoint predictors, even when those methods leverage additional inputs such as depth, panoramic views, and odometry.

CorrectNav during correction training, we select the observation frame at M_t, as well as the frames before and after M_t, as correction keyframes $\{K_1, K_2, K_3\}$. We then leverage a multimodal LLM Qwen-VL-Plus to create vision analysis data based on these key correction frames as shown in the right part of Figure 2. The first type of vision analysis is describing potential navigation landmarks, such as furniture, decorations, or architectural structures, that appear in the given frames.

$$C_i = \text{MLLM}(K_i,\, L_{cap})$$

The second type of vision analysis is generating visual question answering pairs about the frames. These questions concentrate on important visual elements in navigation, including object relative position, object color, and the current robot's orientation.

$$\{(Q_j, A_j)\}_{j=1}^{x} = \text{MLLM}(K_i,\, L_{qa})$$

During the training, we input the observation video $\{I_1, I_2 ... K_i\}$ and use the caption C as the target, train CorrectNav to comprehend the current observation; With the same video, and for any (Q_i, A_i), we instruct CorrectNav to answer Q_i based on current observation (K_i), activating CorrectNav to comprehend error correction behavior.

Step 4 - Model Continued Training With collected self-correction data, we continue the training of CorrectNav. To enhance efficiency, we randomly sample half of the error-correcting trajectories along with their corresponding keyframe perception data for training. Additionally, we incorporate oracle trajectories from the original training data to maintain training stability. The number of these oracle trajectories is set to be half the number of the sampled error-correcting trajectories. By leveraging these automatically generated data, we can further train CorrectNav to enhance its self-correction capabilities. At this time, we have completed one round of Self-correction Flywheel training.

Multi-Round Self-Correction Iteration When we test the self-corrected CorrectNav on the training set again, new error trajectories emerge. These new errors allow us to generate fresh correction task data for the continued training of CorrectNav. This starts the Self-correction Flywheel, enabling multiple rounds of self-correction training iterations.

Implementation Details

CorrectNav is trained on a server with 8 NVIDIA A100 GPUs. The navigation finetuning requires 80 hours while one iteration of Self-correction Flywheel consumes 20 hours. At inference time, CorrectNav takes 16 sampled RGB frames as input and predicts an action chunk with 4 effective actions. More details are included in the appendix.

Experiments

We conduct experiments to answer the questions: (1) How does CorrectNav compare to state-of-the-art models on VLN-CE benchmarks? (2) What improvements does CorrectNav achieve through Self-correction Flywheel iterations? (3) What is the individual impact of each self-correction training technique on CorrectNav 's performance? (4) How effective is CorrectNav in real-world scenarios?

Simulation Experiments

Environment and Metrics We evaluate our VLA on the VLN-CE benchmarks, which provide continuous environments for executing navigational actions in reconstructed

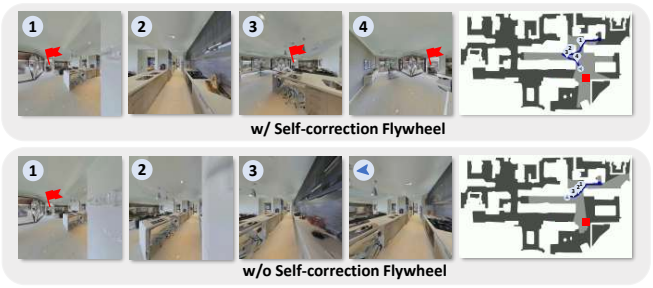

Figure 3: Case study about CorrectNav with and without Self-correction Flywheel post-training. Left Top: CorrectNav mistakenly enters the wrong path, loses the target, and then promptly turns back to return to the correct path. Right Top: CorrectNav first enters the front door, and after realizing there is no target (steps), it leaves and directly enters the correct side door. Vanilla CorrectNav fails in both cases.

photorealistic indoor scenes. We focus on the Val-Unseen split in both R2R (Room-to-Room) and RxR (Room-across-Room) datasets with VLN-CE, as these are the two most recognized benchmarks in VLN. Following the setting of VLN-CE (Krantz et al. 2020) benchmark, we take Habitat 3.0 (Puig et al. 2023) as the simulator to conduct the evaluation. Besides, we employ the following widely used evaluation metrics: Navigation Error (NE), Oracle Success Rate (OS), Success Rate (SR), Success-weighted Path Length (SPL), and normalized dynamic time wrapping (nDTW). Navigation Error, representing the average distance in meters between the agent's final location and the target; Success Rate, indicating the proportion of paths with NE less than 3 meters; Oracle Success Rate, representing the SR given oracle stop policy. nDTW (Ilharco et al. 2019) involves time warping to measure the distance between the model trajectory and ground truth.

Comparison with other VLN-CE Models We compare our VLA with existing VLN-CE models on R2R-CE and RxR-CE benchmarks. These baselines include waypoint predictor-based models and navigation large models. From Table 1, although our CorrectNav only takes monocular RGB observation as input, it outperforms all existing models on R2R-CE and RxR-CE benchmarks. Compared to the state-of-the-art navigation large model StreamVLN, CorrectNav achieves an improvement of 8.2% and 16.4% in success rates on R2R-CE and RxR-CE, respectively. Additionally, CorrectNav outperforms the top waypoint predictor-based models, surpassing HNR by 4.1% on R2R-CE and by 13.0% on RxR-CE.

The Effect of Self-correction Training Technologies To study the contribution of different self-correction training technologies to the improvement of model performance, we conducted ablation studies by removing each technology individually during the first iteration of the Self-correction Flywheel. As shown in Table 2, eliminating any of these technologies results in reduced performance of CorrectNav on both the R2R-CE and RxR-CE Val-Unseen splits. Notably, removing the Navigation Trajectory Correction strategy causes the most substantial performance drop. These ablation results confirm that each self-correction training technology we proposed contributes positively to enhancing the model's overall effectiveness.

	R2R-CE Val-Unseen			RxR-CE Val-Unseen		
	NE ↓	SR ↑	SPL ↑	NE ↓	SR ↑	SPL ↑
CorrectNav	**4.50**	**63.0**	**59.0**	**4.40**	**63.1**	**57.0**
w/o Navigation Trajectory Correction	4.92	59.2	57.2	4.55	60.7	55.1
w/o Error-correcting Keyframe Perception	4.70	60.1	56.5	4.47	61.0	56.3
w/o Data Sampling Strategy	4.71	60.0	57.5	4.47	62.2	56.2

Table 2: Ablation study of self-correction flywheel post-training tasks on the Val-Unseen splits of R2R-CE (Anderson et al. 2018) and RxR-CE (Ku et al. 2020). The experiments are conducted based on the 1_{st} self-correction flywheel post-training.

The Power of Self-correction Flywheel Iteration To investigate the power of the Self-correction Flywheel on improving the navigation performance, we evaluate CorrectNav 's success rate and navigation errors on the R2R-CE and RxR-CE Val-Unseen benchmarks after each Self-correction Flywheel training iteration. The experimental results are plotted as the line graphs shown in Figure 5. From the figure, we can observe that as the Self-correction Flywheel training iterations progress, CorrectNav demonstrates a continuous performance improvement on both benchmarks in the first three iterations. This quantitatively demonstrates that multiple iterations of the Self-correction Flywheel can effectively enhance the capabilities of the navigation VLA model. When CorrectNav's performance drops in the fourth iteration, we stop the training. Additionally, we conduct a qualitative analysis of the Self-correction Flywheel's effects, as illustrated in Figure 3. From the figure, CorrectNav post-trained with Self-correction Flywheel achieves error correction capability compared with the vanilla CorrectNav.

Real Robot Experiments

For real-world experiments, we use the AgiBot Lingxi D1 quadruped robot as our platform. Each Lingxi D1 robot is

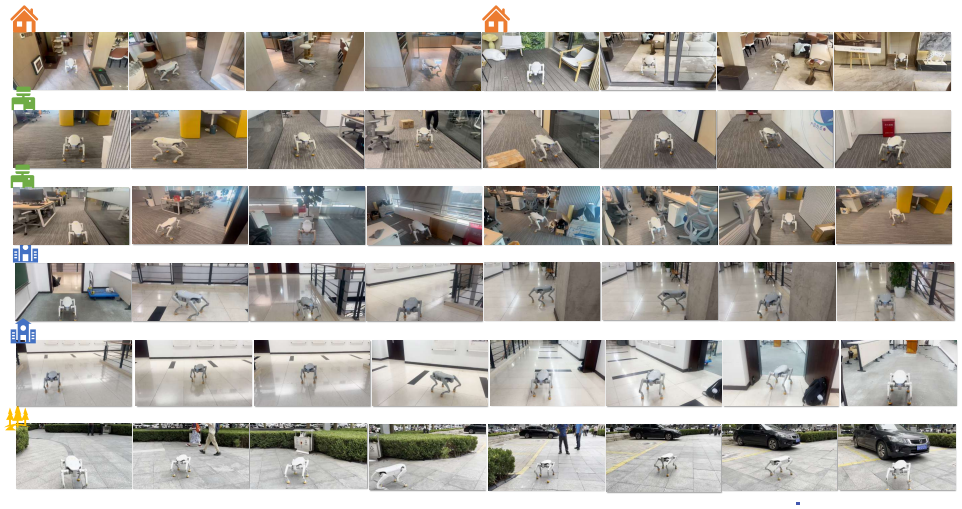

Figure 4: Qualitative results from the real-world deployment of CorrectNav. (c)(d) The robot dynamically avoids pedestrians and obstacles, correctly passing through cluttered environments to reach the destination. (e)(f) The robot successfully recovers from a navigation error to complete a long-horizon instruction. (g) The robot completes outdoor long-distance navigation. Videos are included in the appendix.

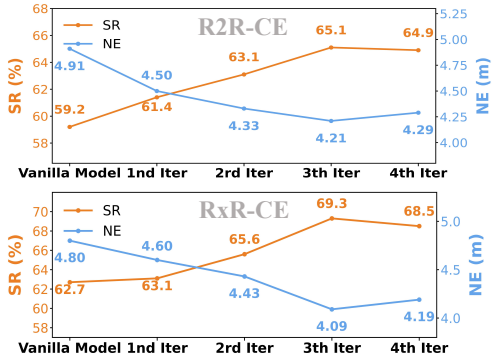

Figure 5: CorrectNav's performance on R2R-CE and RxR-CE Val-Unseen splits over Self-correction Flywheel iterations.

	Office				Home				Campus			
	Simple		Complex		Simple		Complex		Simple		Complex	
	NE↓	SR↑	NE↓	SR↑	NE↓	SR↑	NE↓	SR↑	NE↓	SR↑	NE↓	SR↑
NaVid (Zhang et al. 2024a)	1.88	0.55	4.89	0.30	2.22	0.50	5.27	0.15	1.94	0.55	5.02	0.25
NaVILA (Cheng et al. 2024)	2.06	0.45	5.25	0.20	2.21	0.45	5.49	0.10	1.97	0.50	5.18	0.20
CorrectNav (Ours)	**1.52**	**0.80**	**1.81**	**0.75**	**1.33**	**0.95**	**1.54**	**0.80**	**1.47**	**0.85**	**1.86**	**0.75**

Table 3: Real-world experiments in different environments. Simple and Complex refer to simple and complex instruction-following tasks, respectively.

equipped with a monocular RGB camera and robust motion APIs. After the robot receives a navigation instruction, it will upload the RGB observation image to the CorrectNav deployed on the remote server with an NVIDIA A100 GPU. CorrectNav will predict the navigation action chunk with four actions and call the D1 motion API to execute.

To comprehensively evaluate the effectiveness of our approach, we conduct comparisons with two state-of-the-art navigation large models, NaVID and NaVILA in the office, home, and campus. In each scenario, we test every model on 20 simple instructions and 20 complex instructions, respectively. Complex instructions involve long trajectories, complex architectural structures, crowded obstacles, and dynamic scene changes. The real-world quantitative performances in terms of Success Rate (SR) and Navigation Error (NE) metrics are reported in Table 3. Figure 4 demonstrates the real-world qualitative performances of CorrectNav in indoor and outdoor environments. More real-world experiment results are shown in the appendix.

From Table 3, we can observe that compared to existing navigation large models, CorrectNav demonstrates a stronger ability to execute navigation instructions in the real world. As shown in Figure 4, such improvement mainly stems from the deviation correction capabilities that CorrectNav acquires through Self-correction Flywheel post-training. These capabilities enhance the robustness of CorrectNav on complex instructions, enabling it to quickly correct its own errors or adapt to changes in the environment.

Limitation and Future Work

Although monocular RGB observation-based VLA navigation models, including our CorrectNav, save on the cost of additional sensors, they face a shared limitation: inadequate precision in perceiving the relative positional relationship between the robot's body and its surroundings. The potential risk arising from this deficiency is that when the robot, such as a quadrupedal robot, passes close to obstacles, its hind legs may scrape against them. A promising direction for future research is to explore how to incorporate the robot's body dimensions and state information as prior knowledge for navigation model inference, thereby further improving monocular RGB-based VLA navigation models.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62136001) and National Youth Talent Sup- port Program (8200800081)

References

An, D.; Qi, Y.; Li, Y.; Huang, Y.; Wang, L.; Tan, T.; and Shao, J. 2023. Bevbert: Multimodal map pre-training for language-guided navigation.

An, D.; Wang, H.; Wang, W.; Wang, Z.; Huang, Y.; He, K.; and Wang, L. 2024. Etpnav: Evolving topological planning for vision-language navigation in continuous environments.

An, D.; Wang, Z.; Li, Y.; Wang, Y.; Hong, Y.; Huang, Y.; Wang, L.; and Shao, J. 2022. 1st place solutions for rxr-habitat vision-and-language navigation competition.

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 (CVPR)*.

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. *International Conference on 3D Vision (3DV)*.

Chen, J.; Lin, B.; Liu, X.; Liang, X.; and Wong, K.-Y. K. 2024. Affordances-Oriented Planning using Foundation Models for Continuous Vision-Language Navigation. *arXiv preprint*.

Chen, K.; Chen, J. K.; Chuang, J.; Vázquez, M.; and Savarese, S. 2021. Topological planning with transformers for vision-and-language navigation.

Chen, P.; Ji, D.; Lin, K.; Zeng, R.; Li, T.; Tan, M.; and Gan, C. 2022. Weakly-supervised multi-granularity map learning for vision-and-language navigation.

Cheng, A.-C.; Ji, Y.; Yang, Z.; Gongye, Z.; Zou, X.; Kautz, J.; Bıyık, E.; Yin, H.; Liu, S.; and Wang, X. 2024. Navila: Legged robot vision-language-action model for navigation. *arXiv preprint arXiv:2412.04453*.

Duan, J.; Pumacay, W.; Kumar, N.; Wang, Y. R.; Tian, S.; Yuan, W.; Krishna, R.; Fox, D.; Mandlekar, A.; and Guo, Y. 2024. AHA: A vision-language-model for detecting and reasoning over failures in robotic manipulation. *arXiv preprint arXiv:2410.00371*.

Georgakis, G.; Schmeckpeper, K.; Wanchoo, K.; Dan, S.; Miltsakaki, E.; Roth, D.; and Daniilidis, K. 2022. Cross-modal map learning for vision and language navigation.

Ha, H.; Florence, P.; and Song, S. 2023. Scaling Up and Distilling Down: Language-Guided Robot Skill Acquisition. arXiv:2307.14535.

Hong, Y.; Wang, Z.; Wu, Q.; and Gould, S. 2022. Bridging the gap between learning in discrete and continuous environments for vision-and-language navigation. In *CVPR*.

Hong, Y.; Zhou, Y.; Zhang, R.; Dernoncourt, F.; Bui, T.; Gould, S.; and Tan, H. 2023. Learning navigational visual representations with semantic map supervision.

Ilharco, G.; Jain, V.; Ku, A.; Ie, E.; and Baldridge, J. 2019. General Evaluation for Instruction Conditioned Navigation using Dynamic Time Warping. arXiv:1907.05446.

Krantz, J.; Gokaslan, A.; Batra, D.; Lee, S.; and Maksymets, O. 2021. Waypoint models for instruction-guided navigation in continuous environments.

Krantz, J.; and Lee, S. 2022. Sim-2-sim transfer for vision-and-language navigation in continuous environments.

Krantz, J.; Wijmans, E.; Majundar, A.; Batra, D.; and Lee, S. 2020. Beyond the Nav-Graph: Vision and Language Navigation in Continuous Environments.

Ku, A.; Anderson, P.; Patel, R.; Ie, E.; and Baldridge, J. 2020. Room-Across-Room: Multilingual Vision-and-Language Navigation with Dense Spatiotemporal Grounding.

Lin, B.; Nie, Y.; Zai, K. L.; Wei, Z.; Han, M.; Xu, R.; Niu, M.; Han, J.; Lin, L.; Lu, C.; and Liang, X. 2025. EvolveNav: Self-Improving Embodied Reasoning for LLM-Based Vision-Language Navigation. arXiv:2506.01551.

Liu, H.; Li, C.; Li, Y.; and Lee, Y. J. 2024. Improved baselines with visual instruction tuning.

Liu, Z.; Bahety, A.; and Song, S. 2023. REFLECT: Summarizing Robot Experiences for Failure Explanation and Correction. *arXiv preprint arXiv:2306.15724*.

Long, Y.; Cai, W.; Wang, H.; Zhan, G.; and Dong, H. 2024. InstructNav: Zero-shot System for Generic Instruction Navigation in Unexplored Environment. arXiv:2406.04882.

Ma, Y. J.; Sodhani, S.; Jayaraman, D.; Bastani, O.; Kumar, V.; and Zhang, A. 2023. VIP: Towards Universal Visual Reward and Representation via Value-Implicit Pre-Training. arXiv:2210.00030.

Puig, X.; Undersander, E.; Szot, A.; Cote, M. D.; Partsey, R.; Yang, J.; Desai, R.; Clegg, A. W.; Hlavac, M.; Min, T.; Gervet, T.; Vondrus, V.; Berges, V.-P.; Turner, J.; Maksymets, O.; Kira, Z.; Kalakrishnan, M.; Malik, J.; Chaplot, D. S.; Jain, U.; Batra, D.; Rai, A.; and Mottaghi, R. 2023. Habitat 3.0: A Co-Habitat for Humans, Avatars and Robots.

Raychaudhuri, S.; Wani, S.; Patel, S.; Jain, U.; and Chang, A. 2021. Language-Aligned Waypoint (LAW) Supervision for Vision-and-Language Navigation in Continuous Environments.

Shi, X.; Li, Z.; Lyu, W.; Xia, J.; Dayoub, F.; Qiao, Y.; and Wu, Q. 2025. SmartWay: Enhanced Waypoint Prediction and Backtracking for Zero-Shot Vision-and-Language Navigation. arXiv:2503.10069.

Wang, H.; Liang, W.; Van Gool, L.; and Wang, W. 2023a. Dreamwalker: Mental planning for continuous vision-language navigation.

Wang, Z.; Li, J.; Hong, Y.; Wang, Y.; Wu, Q.; Bansal, M.; Gould, S.; Tan, H.; and Qiao, Y. 2023b. Scaling data generation in vision-and-language navigation.

Wang, Z.; Li, X.; Yang, J.; Liu, Y.; Hu, J.; Jiang, M.; and Jiang, S. 2024. Lookahead Exploration with Neural Radiance Representation for Continuous Vision-Language Navigation.

Wang, Z.; Li, X.; Yang, J.; Liu, Y.; and Jiang, S. 2023c. Gridmm: Grid memory map for vision-and-language navigation.

Wei, M.; Wan, C.; Yu, X.; Wang, T.; Yang, Y.; Mao, X.; Zhu, C.; Cai, W.; Wang, H.; Chen, Y.; Liu, X.; and Pang, J. 2025a. StreamVLN: Streaming Vision-and-Language Navigation via SlowFast Context Modeling. arXiv:2507.05240.

Wei, Z.; Lin, B.; Nie, Y.; Chen, J.; Ma, S.; Xu, H.; and Liang, X. 2025b. Unseen from Seen: Rewriting Observation-Instruction Using Foundation Models for Augmenting Vision-Language Navigation. arXiv:2503.18065.

Xiao, J.; Shang, X.; Yao, A.; and Chua, T.-S. 2021. NExT-QA: Next Phase of Question-Answering to Explaining Temporal Actions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9777–9786.

Yang, A.; Yang, B.; Hui, B.; Zheng, B.; Yu, B.; Zhou, C.; Li, C.; Li, C.; Liu, D.; Huang, F.; Dong, G.; Wei, H.; Lin, H.; Tang, J.; Wang, J.; Yang, J.; Tu, J.; Zhang, J.; Ma, J.; Yang, J.; Xu, J.; Zhou, J.; Bai, J.; He, J.; Lin, J.; Dang, K.; Lu, K.; Chen, K.; Yang, K.; Li, M.; Xue, M.; Ni, N.; Zhang, P.; Wang, P.; Peng, R.; Men, R.; Gao, R.; Lin, R.; Wang, S.; Bai, S.; Tan, S.; Zhu, T.; Li, T.; Liu, T.; Ge, W.; Deng, X.; Zhou, X.; Ren, X.; Zhang, X.; Wei, X.; Ren, X.; Liu, X.; Fan, Y.; Yao, Y.; Zhang, Y.; Wan, Y.; Chu, Y.; Liu, Y.; Cui, Z.; Zhang, Z.; Guo, Z.; and Fan, Z. 2024. Qwen2 Technical Report. arXiv:2407.10671.

Yu, Z.; Xu, D.; Yu, J.; Yu, T.; Zhao, Z.; Zhuang, Y.; and Tao, D. 2019. ActivityNet-QA: A Dataset for Understanding Complex Web Videos via Question Answering. In *AAAI*, 9127–9134.

Zhai, X.; Mustafa, B.; Kolesnikov, A.; and Beyer, L. 2023. Sigmoid loss for language image pre-training.

Zhang, J.; Wang, K.; Wang, S.; Li, M.; Liu, H.; Wei, S.; Wang, Z.; Zhang, Z.; and Wang, H. 2025. Uni-NaVid: A Video-based Vision-Language-Action Model for Unifying Embodied Navigation Tasks. arXiv:2412.06224.

Zhang, J.; Wang, K.; Xu, R.; Zhou, G.; Hong, Y.; Fang, X.; Wu, Q.; Zhang, Z.; and He, W. 2024a. NaVid: Video-based VLM Plans the Next Step for Vision-and-Language Navigation.

Zhang, Y.; Wu, J.; Li, W.; Li, B.; Ma, Z.; Liu, Z.; and Li, C. 2024b. Video Instruction Tuning With Synthetic Data. arXiv:2410.02713.