NaviMaster: Learning a Unified Policy for GUI and Embodied Navigation Tasks

Anonymous ACL submission

Abstract

Recent advances in Graphical User Interface (GUI) and embodied navigation have driven significant progress, yet these domains have largely evolved in isolation, with disparate datasets and training paradigms. In this paper, we observe that both tasks can be formulated as Markov Decision Processes (MDP), suggesting a foundational principle for their unification. Hence, we present NaviMaster, the first unified agent capable of seamlessly integrating GUI navigation and embodied navigation within a single framework. Specifically, NaviMaster (i) proposes a visual-target trajectory collection pipeline that generates trajectories for both GUI and embodied tasks in one formulation. (ii) employs a unified reinforcement learning framework on the mix data for better generalization. (iii) designs a novel distance-aware reward to ensure efficient learning from the trajectories. Through extensive experiments on out-of-domain benchmarks, NaviMaster is shown to outperform state-of-the-art agents in GUI navigation, spatial affordance prediction, and embodied navigation. Ablation studies further confirm the efficacy of our unified training strategy, data mixing strategy, and reward design.

1 Introduction

Graphical user interface (GUI) navigation agents and embodied navigation agents aim to traverse virtual and real environments, respectively. With the rapid progress of multimodal large language models (MLLMs) (Bai et al., 2025), researchers have leveraged their strong perception and planning abilities for both kinds of agents (Wu et al., 2025; Lin et al., 2025). These agents have demonstrated considerable promise in instruction-guided multimodal navigation.

Despite the progress made by previous agents, as illustrated in Fig.1, the long-term separation between GUI and embodied navigation, and their training strategies have led to three persistent challenges. (1) They rely on two individual models for navigation, which increases training and deployment costs and prevents mutually beneficial interaction between the two tasks (Hong et al., 2025). (2) Although prior works (Rawles et al., 2023b; Ramakrishnan et al., 2021) have improved performance in respective tasks by scaling data within specific task data, they have limitations in cross-task due to the poor ability in out-of-domain (OOD) data. (3) They face a training-efficiency bottleneck: previous RFT-based models employ a sparse reward signal, rendering reinforcement learning optimization inefficient.

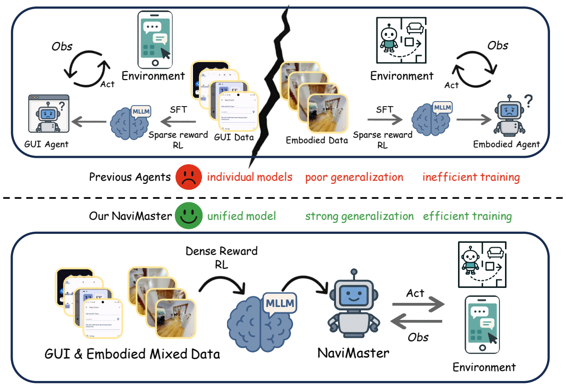


Figure 1: Comparison of NaviMaster and existing agents. Previous methods involve individual models for GUI and embodied navigation. Our NaviMaster is a unified learning framework.

To tackle these challenges, it is essential to build a unified policy that integrates GUI and embodied navigation with an efficient training strategy. Inspired by Markov Decision Processes (MDP):

$$\arg\max_{a\in\mathcal{A}} P(\mathcal{S}_{t+1} = \sigma' \mid \mathcal{S}_t = \sigma, \mathcal{A}_t = a), \quad (1)$$

where next state $\mathcal{S}_{t+1}$ is fully determined by current state–action pair (σ, a). We observe that both GUI and embodied navigation naturally conform to this MDP structure. In a unified formulation, the state $\mathcal{S}_t$ is the observation at step t, the action space $\mathcal{A}$ spans interactions with either a virtual interface or the physical environment. The action a is only operated on $\mathcal{S}_t$. This transition dynamics satisfy the Markov property: $\mathcal{S}_{t+1}$ is conditionally independent of all earlier states given $(\mathcal{S}_t, a)$. Consequently, we formalize the notion of a *Navigation Agent* that seamlessly integrates GUI and embodied navigation under one coherent framework.

Therefore, to improve generalization and data efficiency, we propose *NaviMaster*, the first unified Navigation Agent. As shown in Fig.1, NaviMaster exhibits three key breakthroughs. (1) We develop a pipeline for trajectory collection that reformulates both GUI and embodied navigation into a visual-target paradigm, which enables effective training on GUI and embodied mix data, significantly improving data diversity and generalization. (2) We build a unified reinforcement learning training framework for both GUI and embodied navigation. Specifically, we take prior reasoning thoughts and actions as historical context. Given the history and current observations as inputs, it predicts the next action. This setup unifies the I/O and the history enables more precise high-level action in long-horizon navigation. Furthermore, we extend the training strategy to estimate task-specific advantages, enabling a single policy to adapt effectively across multiple tasks. (3) We employ a distance-aware dense reward in reinforcement learning, which further improves training efficiency compared with a sparse, binary reward.

We evaluate our NaviMaster on OOD GUI and embodied navigation datasets. On test sets that are OOD relative to the training data, NaviMaster has achieved the best performance on several benchmarks compared to the current state-of-the-art baselines, demonstrating strong generalization and robustness. In summary, our key contributions are as follows:

1. We propose NaviMaster, the first unified Navigation Agent that integrates both GUI navigation and embodied navigation under a common framework.
2. We develop the visual-target trajectory collection pipeline that integrates high-quality trajectories from both GUI and embodied navigation, enriching data diversity and enhancing the model's generalization capabilities.
3. We design a distance-aware dense reward and unified reinforcement learning pipeline that improves data efficiency and further bolsters the grounding ability.

2 Related Work

2.1 Navigation Agent

GUI navigation agents aim to autonomously operate applications by perceiving UI elements and issuing precise point-level actions (Wang et al., 2025). Recent efforts tend to use the data-driven training paradigm like OS-atlas, UI-Tars (Wu et al., 2025; Qin et al., 2025). They employ large-scale datasets in a multi-stage training pipeline to further boost their UI grounding precision and planning capability. Despite this progress, most existing GUI agents rely heavily on supervised fine-tuning (SFT) with large amounts of human-annotated data, limiting their generalization capacities. To address this, models like UI-R1 and GUI-R1 (Luo et al., 2025; Lu et al., 2025) incorporate reinforcement learning (RL) inspired by DeepSeek-R1. But their scope remains restricted to GUI-only settings and lack the capacity to generalize to embodied navigation.

Embodied navigation agents control physical or simulated agents to follow language instructions in 3D spaces, demanding multimodal perception and long-horizon planning (Gao et al., 2024). Analogous to GUI navigation tasks, works on embodied navigation typically employ a multi-stage SFT strategy on large datasets to adapt open-source MLLMs for navigation tasks(e.g., RoboPoint (Yuan et al., 2024), SpaceLLaVa (Foutter et al., 2025)). Their scope remains restricted to one domain's setting, which enforces a monolithic action space, thereby diminishing the agents' capacity to generalize when the action space changes.

Very recently, Embodied Web Agent (EWA) (Hong et al., 2025) is the first work that unifies physical embodiment with live web interfaces. Although EWA unifies web and embodied tasks, it lacks an emphasis on grounding capabilities and fails to establish a comparable action space between the two navigation agent types. It also relies on zero/few-shot MLLMs without a unified navigation training paradigm, limiting its value for developing general-purpose navigation agents. In this paper, we unify the action space and realize joint training for GUI and embodied navigation in ONE agent.

2.2 Reinforcement Fine-Tuning on MLLM

Visual-RFT (Liu et al., 2025b) performs reinforcement fine-tuning on LVLMs using their own reasoning traces together with rule-based, verifiable visual rewards—e.g., IoU for detection and CLS accuracy for classification. UI-R1 (Lu et al., 2025) introduces a unified, rule-based reward that measures the click-coordinate accuracy within the ground-truth bounding box, thereby enhancing the precision of GUI action prediction. GUI-R1 (Luo et al., 2025) also adopts a similar reward design, but places greater emphasis on high-level GUI navigation capabilities. However, their reward design is strictly binary; only responses that fall within the ground truth bounding box receive a positive score. This leads to many rollouts in GRPO yielding zero reward, making the training process less effective (Zheng et al., 2025). Differently, we adopt a dense reward approach for grounding training in navigation agents. Unlike prior work that relies on binary rewards, our method assigns scores based on the proximity of the response to the ground truth, thereby improving grounding performance while promoting more efficient and stable training.

3 NaviMaster

3.1 Overview

Our proposed NaviMaster consists of three key components, including (1) the visual-target trajectory collection to reformulate the GUI and embodied navigation trajectories into a unified form with historical information, (2) the unified reinforcement learning framework to optimize the cross-scenario at the same time, and (3) the distance-aware reward to update the model parameters by additionally considering the distance between output points and target points.

3.2 Visual-Target Trajectory Collection

As Fig.2 shows, the visual-target trajectory collection has three parts, including unified action space definition, unified trajectory initialization, and reasoning thought generation.

Unified Action Space Definition. Existing GUI and embodied trajectory data differ significantly in the action space in terms of localization action. In GUI task, the localization action is implemented by the [**CLICK** (x, y)] action with a special target (x, y) in the screenshot. In contrast, embodied navigation task implements localization using the [**MOVEFORWARD**] action without a target. The interaction paradigms for the two action spaces are different. GUI actions $\mathcal{A}_{gui}$ rely on precise target-oriented operations (e.g., clicking specific UI elements) while embodied actions $\mathcal{A}_{emb}$ emphasize egocentric motion control (e.g., navigating without explicit target selection).

The localization action difference (i.e., with or without a target) poses a problem for unifying the GUI and embodied navigation tasks. To address this, in this paper, we propose the visual-target trajectory by introducing a localization action with a target into the embodied navigation task. As shown in the left part of Fig.2, we define a specific visual target in the observation for each step in the trajectory. Hence, the localization action in the embodied navigation becomes [**MOVETO** (x, y)] with a target (x, y) in place of [**MOVEFORWARD**] action[1].

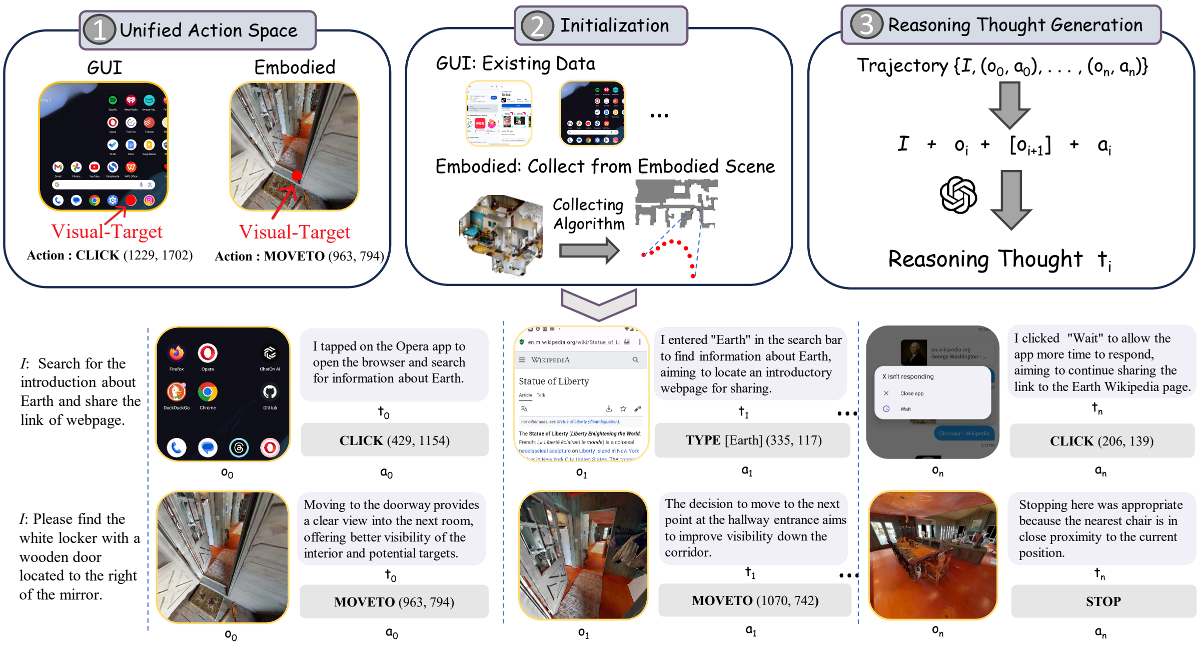
Trajectory Collection Initialization. Considering the long task with n steps, it is formulated as $\{I, (o_0, a_0), \ldots, (o_n, a_n)\}$, where I is the instruction from the user, o_i is the observation from GUI screenshot or physical environment at step i, and a_i is the action at step i ($0 \leq i \leq n$). This formulation is the same as the popular GUI trajectories. Hence, we can straightforwardly utilize the existing GUI dataset to obtain the GUI trajectories (e.g., GUI-Odyssey (Lu et al., 2024) in our experiments).

However, the existing embodied navigation dataset only provides the initial position and the destination position without the trajectories. Given an embodied navigation dataset (e.g., Matterport 3D dataset with the Habitat simulator (Yadav et al., 2023; Savva et al., 2019)), we first get the set of trajectory points on the shortest path $(\mathbf{s_0}, \mathbf{s_1}, \ldots, \mathbf{s_m})$ from the initial position to the destination position, which can be mapped by performing the A* search method (Hart et al., 1968). Each trajectory point $\mathbf{s_k}$ ($0 \leq k \leq m$) is a 3D coordinate in the global coordinate system.

Then, based on the point set, we collect observation images and generate visual-target actions for embodied navigation. We describe initialization of trajectory collection for embodied task in Algorithm 1. The first step is to align the next position to the current observation. Given the current position $\mathbf{s_k}(u_k, v_k, w_k)$ (global coordinate system), its camera rotation $\mathbf{r_k}$, and the next position $\mathbf{s_{k+1}}(u_{k+1}, v_{k+1}, w_{k+1})$ (global coordinate system), the $\mathbf{s}'_{\mathbf{k+1}}(u'_{k+1}, v'_{k+1}, w'_{k+1})$ ($\mathbf{s_k}$ coordinate system) can be obtained based on the follow-

---

[1] The full action space is in Appendix C.



I: Search for the introduction about Earth and share the link of webpage.

I tapped on the Opera app to open the browser and search for information about Earth.

CLICK (429, 1154)

en.m.wikipedia.org/wiki/Statue_of_Li

Wikipedia

Statue of Liberty

Article Talk

For other uses, see Statue of Liberty (disambiguation).

The Statue of Liberty (*Liberty Enlightening the World*; French: *La Liberté éclairant le monde*) is a colossal neoclassical sculpture on Liberty Island in New York Harbor in New York City, United States. The conne

I entered "Earth" in the search bar to find information about Earth, aiming to locate an introductory webpage for sharing.

TYPE [Earth] (335, 117)

en.wikipedia.org George Washington - ...

X isn't responding

Dinosaur - Wikipedia

I clicked "Wait" to allow the app more time to respond, aiming to continue sharing the link to the Earth Wikipedia page.

CLICK (206, 139)

I: Please find the white locker with a wooden door located to the right of the mirror.

Moving to the doorway provides a clear view into the next room, offering better visibility of the interior and potential targets.

MOVETO (963, 794)

The decision to move to the next point at the hallway entrance aims to improve visibility down the corridor.

MOVETO (1070, 742)

Stopping here was appropriate because the nearest chair is in close proximity to the current position.

STOP

Figure 2: Visual-Target Trajectory Collection contains three parts. We unify the GUI and the Embodied action space by introducing a visual target at each step. Then we initialize the trajectories from existing datasets or scenes. Last, we generate a first-person thought t_i with GPT-4o. Finally, we get our visual-target trajectories τ.

ing equation:

$$\mathbf{s}'_{\mathbf{k+1}} = \mathbf{r}_{\mathbf{k}}^{-1} \times (\mathbf{s}_{\mathbf{k+1}} - \mathbf{s}_{\mathbf{k}}) \times \mathbf{r}_{\mathbf{k}}. \tag{2}$$

After that, we project $\mathbf{s}'_{\mathbf{k+1}}$ onto the current camera observation o_i:

$$p(x_i, y_i) = \left(\frac{W}{2} + f \cdot \frac{u'_{k+1}}{w'_{k+1}}, \frac{H}{2} + f \cdot \frac{v'_{k+1}}{w'_{k+1}}\right), \tag{3}$$

where $p(x_i, y_i)$ represents the coordinates of the next position in the current observation o_i, (W, H) is the width and height of the image, and f is the camera focal length.

Due to the limitations of the camera's pitch angle and field of view, the projected coordinates may not appear within the observation. To address this, we define several custom actions to adjust the camera angle, including the left-right and up-down turning actions in embodied tasks: [**TURNLEFT**], [**TURNRIGHT**], [**TURNAROUND**], [**LOOKDOWN**] (adjusts view to face downwards) and implement them as Algorithm 1. Note that w'_{k+1} is the s'_{k+1}'s depth in the current observation, and if $w'_{k+1} < 0$, it means s'_{k+1} is behind the camera. After getting the observation o_i with a target at each position, we can represent each embodied navigation trajectory with the same action space and style of GUI navigation.

Algorithm 1 Trajectory Collection Initialization for Embodied Task

```
Input: $I$, $(\mathbf{s_0}, \mathbf{s_1}, \ldots, \mathbf{s_m})$
Output: Trajectory
Initialize Trajectory with $(I)$, $i$ with 0
for each $k \in [0, m]$ do
    if $k < n$ then
        Caculate $\mathbf{s}'_{\mathbf{k+1}}(u'_{k+1}, v'_{k+1}, w'_{k+1})$
        while $p(x_i, y_i) \notin [0, W] \times [0, H]$ do
            $o_i$ ← observation at $(s_k, r_k)$
            Update $p(x_i, y_i)$
            if $w'_{i+1} < 0$ then
                $a_j$ ← TURNAROUND
            else if $x_i < 0$ then
                $a_i$ ← TURNLEFT
            else if $x_i > W$ then
                $a_i$ ← TURNRIGHT
            else if $y_i > H$ then
                $a_i$ ← LOOKDOWN
            end if
            Append $(o_i, a_i)$ to Trajectory
            Update $r_k$ ← execute $a_i$
            $i \leftarrow i + 1$
        end while
        $a_i$ ← MOVETO $(x_i, y_i)$
        Append $(o_i, a_i)$ to Trajectory
        $i \leftarrow i + 1$
    else
        $o_i$ ← observation at $(s_k, r_k)$, $a_i$ ← STOP
        Append $(o_i, a_i)$ to Trajectory
    end if
end for
return Trajectory
```

Reasoning Thought Generation. Historical information has been proved useful for the agents (Yang et al., 2025). Most methods (Xu et al., 2025) take the implemented actions as the history. However, history with only action will confuse the system. For example, [**CLICK** (x, y)] cannot indicate the happened trajectory. By contrast, the reasoning thought "I should first open Chrome to start my search" and action "[**CLICK** (x, y)]", which clearly indicates that the agent opens the app Chrome. Some existing work (Qin et al., 2025) also proves that adding the reasoning thought to each step of the trajectory enables the model to express its decision-making process explicitly.

Therefore, to enhance reasoning ability and optimize memory usage, we generate thought for each action of the trajectory as the bottom part of Fig.2 shows. Given the initialized trajectory, we construct a data generation pipeline:

$$\langle I, o_i, a_i, [o_{i+1}] \rangle \xrightarrow{\mathcal{M}} t_i. \tag{4}$$

Specifically, we feed the task instruction I, observation o_i, and action a_i into the large language model, prompting it to generate an intention t_i from a first-person perspective that explains the reason for taking action a_i. $\mathcal{M}$ is GPT-4o (OpenAI et al., 2024) in our experiment. Note that o_{i+1} is only used for the generation in the GUI trajectory, since the GUI needs the target observation as the reference during data generation. The details generating prompts are in the Appendix A.

Therefore, our visual-target trajectory for both GUI and embodied navigation are formatted as $\tau = \{I, (o_0, t_0, a_0), \ldots, (o_n, t_n, a_n)\}$.

3.3 Unified Reinforcement Learning Framework

Since reinforcement learning performs better in generalization than supervised finetuning, we follow the R1-zero training strategy, without cold-start pretraining and directly training on our collected dataset using Group Relative Policy Optimization (GRPO). As illustrated in Fig.3, given an n-step trajectory, we take step i from the trajectory as a piece of data. It includes user instruction I, observation o_i, reasoning thoughts and actions in history $H_i = \{(t_0, a_0), (t_1, a_1), ..., (t_{i-1}, a_{i-1})\}$. Then NaviMaster learns the unified policy with GRPO. Specifically, with input queries $\{I, H_i, o_i\}$, we operate on G samples $\{\gamma_j = \pi_{\theta_{\text{old}}}(a_i | I, H_i, o_i)\}_{j=1}^{G}$ produced by the policy model π_θ. We also incorporate the depth map h_i of observation o_i as a critical prior for grounding in spatial space. We compute the advantage Adv as follows:

$$R(i, j) = R(\gamma_j, a_i, h_i), \tag{5}$$

$$Adv = \frac{R(i, j) - \text{mean}(\{R(i, j)\}_{j=1}^{G})}{\text{std}(\{R(i, j)\}_{j=1}^{G})}, \tag{6}$$

where $R(i, j)$ denotes the reward function of the response. It will be detailed in the next section.

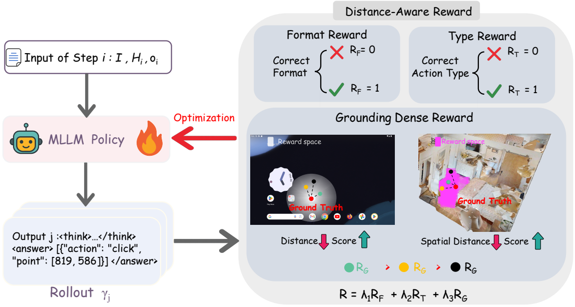


Figure 3: Overview of unified reinforcement learning framework. MLLM policy is optimized using GRPO with format, type and grounding dense reward.

3.4 Distance-Aware Reward

The criteria for successful task execution are threefold: (1) the output must be correctly parsed into an executable action, (2) the type of the executed action must match the ground truth, and (3) the action's arguments must fall within reasonable bounds. Accordingly, the reward is decomposed into three factors: format, type, and grounding. At the same time, most existing reward designs rely on binary success/failure signals. We expect the model to learn relative preferences even from failed rollouts. For instance, among unsuccessful rollouts, some may still be "better" than others. Specifically, we design a distance-aware dense reward for the grounding component based on the distance to the ground-truth point, which consists of three components, including format reward, type reward and grounding dense reward.

Format Reward. This reward $R_F(i, j)$ enforces the formatting of the output. Each response must first think and then answer, and the answer content must be valid JSON. It should follow the structure: "$\langle think \rangle \cdots \langle /think \rangle$ $\langle answer \rangle$ $json\ string$ $\langle /answer \rangle$". If a rollout satisfies this format, $R_F(i, j)$ will be set to 1. Otherwise, $R_F(i, j)$ will be set to 0.

Type Reward. This reward $R_T(i, j)$ supervises the correctness of the model's action selection. It is a binary reward that evaluates action-type correctness within a small discrete selection space. It supervises the model's ability to make high-level decisions aligned with task semantics. Let $\hat{a}_j$ be the predicted action type of the sample γ_j and a_i be the ground truth at step i. Then:

$$R_T(i, j) = \begin{cases} 1, & \text{if } \hat{a}_j = a_i, \\ 0, & \text{otherwise}. \end{cases} \tag{7}$$

Grounding Dense Reward. This reward R_G is designed to supervise the grounding ability. Specifically, this ability requires selecting the correct target within a large selection space, such as a pixel-level coordinate in an image. It evaluates the predicted location relative to the ground truth at step i. To unite and measure the grounding capability both in embodied tasks and GUI tasks, we define a distance-based dense reward instead of a sparse reward. We want that the agent can realize if it is closer to the target (UI elements in GUI and next position in embodied scene), the more reward it can get. It can avoid most of the meaningless exploration and function as a guide for the agent during training processing. The grounding reward function is designed as follows:

$$R_G(i,j) = \begin{cases} 1 - \frac{d_j}{\theta_d} & d_j < \theta_d, p_j < \theta_h, \\ 0 & otherwise, \end{cases} \tag{8}$$

where θ_d and θ_h are thresholds for distance d_j and depth disparity p_j. The d_j is the pixel-level distance between prediction point $(\hat{x}_j, \hat{y}_j)$ of γ_j and groundtruth point (x_i, y_i). In embodied environments, we consider the depth value $h_i(\hat{x}_j, \hat{y}_j)$ in addition because two close pixel points in an image can have significantly different depths in a 3D scene due to occlusion in the scene. The definitions of d_j and p_j are as follows:

$$d_j = \sqrt{(\hat{x}_j - x_i)^2 + (\hat{y}_j - y_i)^2}, \tag{9}$$

$$p_j = |h_i(\hat{x}_j, \hat{y}_j) - h_i(x_i, y_i)|, \tag{10}$$

The overall reward function is a weighted combination of the three components described above. In each step t, it can be expressed as follows:

$$R(i,j) = \lambda_1 R_F(i,j) + \lambda_2 R_T(i,j) + \lambda_3 R_G(i,j), \tag{11}$$

where hyperparameters $\lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}_+$ controlling their relative importance.

4 Experiments

4.1 Implementation Details

We trained our model in the EasyR1 framework (Yaowei Zheng, 2025) and used QwenVL2.5-7B model (Bai et al., 2025) as the base model for experiments. We conduct training for ten epochs on 8 NVIDIA A800 GPUs, with a global batch size of 128 and a learning rate of 1×10^{-6}. The λ_1, λ_2, and λ_3 are experimentally set to 0.1, 1, and 1, respectively. Note that we only take 7000 pieces of data, including 3500 GUI data from GUI-Odyssey and 3500 embodied data from Matterport 3D and RoboPoint, for training, far less than existing SFT-based methods (mostly at the million level). More details are in the Appendix D.

4.2 Benchmarks and Metrics

GUI task. For the evaluation of the GUI task, we employ five distinct agent benchmarks: AC-High/Low (Li et al., 2024), AITW (Rawles et al., 2023a), GUIAct-Phone (Chen et al., 2025), LlamaTouch (Zhang et al., 2024b), and AITZ (Zhang et al., 2024a). We follow OS-Atlas (Wu et al., 2024) to take the success rate (SR) as the evaluation metric. But we argue that the metric of Type (the accuracy of the predicted action type) is not reasonable due to dataset bias. As shown in Table 1, a model predicting all actions as **[CLICK]** can still achieve a high Type prediction accuracy. Nevertheless, we report the performance of Type for comprehensive information.

We compare our model with the following methods: the proprietary GPT-4o (OpenAI et al., 2024), SFT-based models like OS-Atlas, Aguvis (Xu et al., 2025) and Qwen2.5VL-7B* fine-tuned on our trajectory data, RL-based models like GUI-R1 (Luo et al., 2025), infiGUI-R1 (Liu et al., 2025a) and UI-Shift (Gao et al., 2025).

Metric	AC-High/Low	AITW	GUIAct-Phone	LlamaTouch	AITZ
Type	59.7	57.2	58.0	64.4	55.9

Table 1: The bias of action types in testing datasets.

Embodied task. We assess our model's performance through two distinct embodied tasks. First, we conduct spatial affordance prediction to assess the model's spatial grounding ability on the metric of SR. Specifically, we employ RoboRefIT (Lu et al., 2023) to test object referring and Where2Place (Yuan et al., 2024), RoboSpatial (Song et al., 2025), RefSpatial (Zhou et al., 2025) to test free space referring. The second task is embodied navigation, which evaluates the model's practical application capabilities on the metric of SR and SPL (Success Rate Weighted by Inverse Path Length). We evaluate our approach on the unseen validation branch of ObjectNav (Batra et al., 2020). We adhere to the framework established in VLMNav (Goetting et al., 2024) and substitute the agent model in our experiments to assess

Models	AC-Low		AC-High		AITW		GuiAct-Phone		LlamaTouch		AITZ	
	Type	SR	Type	SR	Type	SR	Type	SR	Type	SR	Type	SR
GPT-4o	74.3	28.4	63.1	21.2	36.4	20.3	46.9	28.4	72.6	30.8	53.4	16.4
Qwen2.5VL-7B	83.4	62.5	68.7	47.1	56.0	38.2	55.4	38.2	72.0	51.1	56.7	28.0
Qwen2.5VL-7B*	68.0	53.9	62.5	38.8	58.1	22.4	61.9	29.6	65.1	36.9	57.4	24.8
OS-Atlas-PRO	91.6	82.0	85.7	70.4	62.5	41.4	52.2	29.4	51.8	30.1	54.0	27.6
OS-Atlas-7B	73.0	50.9	55.0	29.8	-	-	-	-	-	-	-	-
Aguvis	72.1	57.6	67.4	50.0	71.6	53.1	63.4	42.7	84.8	**60.4**	59.9	36.2
infiGUI-3B	96.0	92.1	82.7	71.1	44.6	38.5	39.0	31.0	61.0	46.5	47.2	36.8
GUI-R1-7B	85.1	66.5	71.6	48.1	66.9	50.4	55.3	42.0	82.9	58.2	66.0	43.8
UI-shift	90.9	70.3	71.7	48.9	61.0	43.1	63.0	40.9	67.5	48.4	58.2	42.0
Ours (w/o Embodied)	83.7	66.3	61.7	45.6	57.3	50.0	58.1	42.3	77.1	55.2	60.1	**48.9**
Ours (w/o GUI)	70.1	62.1	64.6	47.8	60.1	49.1	55.1	43.8	74.4	56.5	60.9	43.8
Ours	85.6	**68.9**	72.9	**54.0**	59.4	**51.2**	61.7	**46.4**	81.7	59.8	56.4	46.3

Table 2: Results on GUI tasks. The red background represents that the data source is in the training set of the corresponding model, while the green background represents that the test dataset is OOD for the model. **Bold** highlights the best results in the OOD setting, and underlined are the second-best.

the performance of our proposed approach.

We compare our model against the proprietary GPT-4o, open-source methods like Qwen2.5VL-7B (Bai et al., 2025), SpatialVLM like SpaceLLaVA (Foutter et al., 2025), the latest method RoboPoint-13B (Yuan et al., 2024).

More details of benchmarks and metrics are in Appendix E and Appendix F.

4.3 Main Results

GUI Navigation. The results are shown in the Table 2. To show the generalization capability of NaviMaster, the test data are entirely out-of-domain (OOD) to our training data, which are marked in the green background. While the rest means the test data are in-domain, marked in the red background. Compared with other state-of-the-art baselines, our NaviMaster demonstrates superior performance across the various benchmarks on SR metric, highlighting its strong generalization capability and robustness in handling OOD datasets. Additionally, compared to models trained solely on GUI data or embodied navigation data, our model trained on the mix data achieves the best performance across all test datasets. This highlights the efficiency of our collected visual-target trajectory and unified training framework, which enables high performance and strong generalization with a relatively smaller amount of data. Note that we argue Type is not a meaningful metric as discussed in Table 1, and we report it only for sufficient information.

Spatial Affordance Prediction. We utilize two types of spatial affordance predicting datasets: one is object referring, which involves identifying and localizing specific objects within a scene based on natural language descriptions; the other is free space referring, which focuses on understanding and navigating to spatial regions or locations that are not necessarily tied to specific objects but are described in natural language. Table 3 summarizes the average success rate of predicted points falling within the ground-truth mask on the four spatial affordance prediction benchmarks. Compared with baselines, NaviMaster performance best in all the spatial affordance prediction tasks. These results demonstrate that NaviMaster's fine-grained visual–spatial alignment significantly improves both object-level and free-space referring.

Models	RoboReflt	Where2Place	RoboSpatial	RefSpatial
GPT-4o	15.28	29.00	5.70	8.40
Qwen2.5VL-7B	3.46	3.00	10.31	3.55
SpaceLLaVA	21.30	11.84	2.50	4.02
RoboPoint-13B	49.82	46.77	19.70	8.40
Ours (w/o Embodied)	67.86	32.91	14.75	11.67
Ours (w/o GUI)	76.23	49.84	19.49	20.92
Ours	**77.34**	**49.90**	**20.49**	**23.32**

Table 3: Results on spatial affordance prediction.

Embodied Navigation. Since we are the first to train an agent model that has the ability to generalize performance in VLMNav, there is no prior work that trains a navigation model under the VLMNav framework for comparison. We only report our results in Table 4, which reports ObjectNav performance on the ObjectNav benchmark. NaviMaster attains the highest Success Rate (SR) of 33.10% and an SPL of 12.60%, showing a clear gain over the base model (27.23% SR / 9.68% SPL). Training exclusively on embodied data or GUI data yields slightly lower SR (31.10% and 31.00%), indicating that the mix training strategy best exploits the complementary strengths of both data.

4.4 Ablation Study

Base model. To verify that our improvements arise from the training framework rather than the intrinsic capacity of Qwen2.5VL-7B, we evaluated the

Runs	SR	SPL
Qwen2.5VL-7B	27.23	9.68
-Embodied	31.10	11.20
-GUI	31.00	10.05
-Mix	**33.20**	**12.60**

Table 4: Results on embodied navigation.

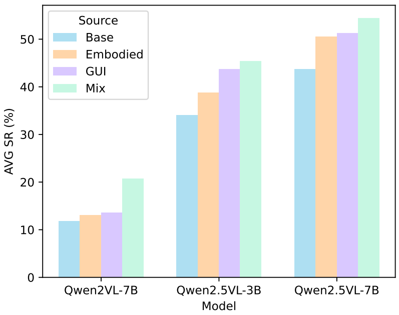


Figure 4: Performance of different base models.

framework on multiple base models that varied in parameter scale and the amount of pre-training knowledge. Specifically, we used Qwen2.5VL-3B and Qwen2VL-7B (Wang et al., 2024) for experimentation, the latter possessing comparatively limited pre-training knowledge. As shown in Fig.4, our method yields consistent performance improvements across different base models. Moreover, it is again proved that training on mix data consistently outperforms training on a single data type, regardless of the underlying model.

The Usage Strategies of Two Datasets. Regarding dataset utilization, there are primarily two strategies. One is to mix different types of data or tasks into a single training phase (Mix). The other is to adopt a multi-stage schedule, with each stage focusing on one specific task or subset of data (GUI-Embodied or Embodied-GUI). As shown in the left of Fig.5, the mix training strategy generally outperforms the two-stage training strategy across various benchmarks. This suggests that training with mix data in a single phase enables the model to exploit complementary information effectively, leading to superior performance and generalization.

Embodied Data Source. Our embodied data consists of two parts: one part is derived from the point-based data we construct (trajectory), and the other part is the spatial affordance prediction data from RoboPoint (affordance). We evaluate the model when trained on each source individually and on their union (affordance + trajectory). As shown in the right of Fig.5, combining both sources under an equal total data volume yields the best training performance.

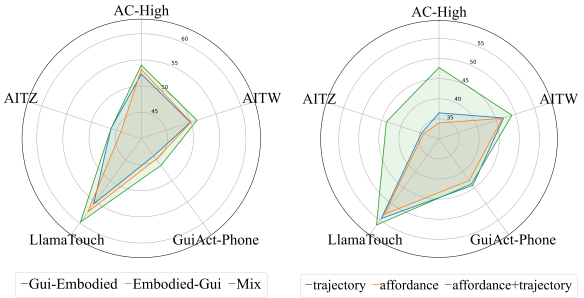


Figure 5: Left: Comparison of data usage strategies. Right: Comparison of embodied data sources.

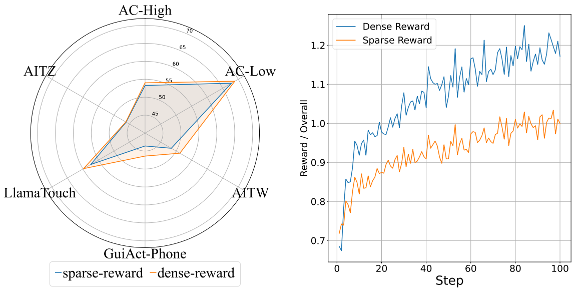


Figure 6: Left: Comparison of reward designs. Right: Reward curves.

Reward Design. We assess the effectiveness of the proposed grounding dense reward by replacing it with a sparse alternative and retraining under identical settings. Specifically, the threshold is set to $\hat{\theta}_d = 20$ in the sparse reward setting, whereas $\theta_d = 200$ is used in the dense reward setting. For the sparse reward, if the distance between the predicted point and the ground-truth point is less than $\hat{\theta}_d$, the reward is 1; otherwise, it is 0. The results in the left of Fig.6 show that the model trained with the dense reward consistently outperformed the one trained with the sparse reward. Moreover, the reward curve under dense setting in Fig.6 rises more rapidly, indicating more efficient training.

5 Conclusion

In this work, we introduce NaviMaster, the first unified navigation agent that seamlessly integrates both GUI and embodied navigation within a single reinforcement learning framework. By reformulating both navigation types into a visual-target trajectory format, we bridge the gap between two previously disparate domains, enabling joint training and cross-task generalization. Our distance-aware dense reward design further enhances the training efficiency and grounding precision, overcoming the limitations posed by traditional sparse reward signals. Extensive experimental results demonstrate the superior OOD performance of NaviMaster.

Limitations

While our NaviMaster improves the performance of both GUI and embodied navigation tasks significantly, especially in OOD scenes, it still treats GUI and embodied navigation as two different tasks. Our trajectory dataset lacks any single trajectory that interleaves GUI and embodied navigation tasks due to the collection difficulty. Future work should be constructing a navigation agent that supports interacting with the GUI and embodied environments at the same time.

Broader Impacts

The GUI or embodied data may leak personal information such as phone number or face. The data collection pipeline we proposed will not introduce any significant privacy such as personal information. It will not contain any real information as all the data source are virtual or from open-source datasets.

The navigation agent will interact with the OS system or real-world environments. This will potentially affect the functioning of the system or take risky actions to damage the environment. However, all the settings in our experiments are in virtual environments or on a monitor. We do not view this as a concern.

References

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. 2020. Objectnav revisited: On evaluation of embodied agents navigating to objects. *Preprint*, arXiv:2006.13171.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, Yuan Yao, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2025. Guicourse: From general vision language models to versatile gui agents. *Preprint*, arXiv:2406.11317.

Matthew Foutter, Daniele Gammelli, Justin Kruger, Ethan Foss, Praneet Bhoj, Tommaso Guffanti, Simone D'Amico, and Marco Pavone. 2025. Space-llava: a vision-language model adapted to extraterrestrial applications. *Preprint*, arXiv:2408.05924.

Longxi Gao, Li Zhang, and Mengwei Xu. 2025. Uishift: Enhancing vlm-based gui agents through self-supervised reinforcement learning. *Preprint*, arXiv:2505.12493.

Peng Gao, Peng Wang, Feng Gao, Fei Wang, and Ruyue Yuan. 2024. Vision-language navigation with embodied intelligence: A survey. *Preprint*, arXiv:2402.14304.

Dylan Goetting, Himanshu Gaurav Singh, and Antonio Loquercio. 2024. End-to-end navigation with vision language models: Transforming spatial reasoning into question-answering. *Preprint*, arXiv:2411.05755.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

Yining Hong, Rui Sun, Bingxuan Li, Xingcheng Yao, Maxine Wu, Alexander Chien, Da Yin, Ying Nian Wu, Zhecan James Wang, and Kai-Wei Chang. 2025. Embodied web agents: Bridging physical-digital realms for integrated agent intelligence. *Preprint*, arXiv:2506.15677.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. 2024. On the effects of data scale on ui control agents. *Preprint*, arXiv:2406.03679.

Bingqian Lin, Yunshuang Nie, Khun Loun Zai, Ziming Wei, Mingfei Han, Rongtao Xu, Minzhe Niu, Jianhua Han, Liang Lin, Cewu Lu, and Xiaodan Liang. 2025. Evolvenav: Self-improving embodied reasoning for llm-based vision-language navigation. *Preprint*, arXiv:2506.01551.

Yuhang Liu, Pengxiang Li, Congkai Xie, Xavier Hu, Xiaotian Han, Shengyu Zhang, Hongxia Yang, and Fei Wu. 2025a. Infigui-r1: Advancing multimodal gui agents from reactive actors to deliberative reasoners. *Preprint*, arXiv:2504.14239.

Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. 2025b. Visual-rft: Visual reinforcement fine-tuning. *Preprint*, arXiv:2503.01785.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. 2024. Gui odyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. *Preprint*, arXiv:2406.08451.

Yuhao Lu, Yixuan Fan, Beixing Deng, Fangfu Liu, Yali Li, and Shengjin Wang. 2023. Vl-grasp: a 6-dof interactive grasp policy for language-oriented objects in cluttered indoor scenes. *Preprint*, arXiv:2308.00640.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanjing Xiong, and Hongsheng Li. 2025. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. *Preprint*, arXiv:2503.21620.

Run Luo, Lu Wang, Wanwei He, and Xiaobo Xia. 2025. Gui-r1 : A generalist r1-style vision-language action model for gui agents. *Preprint*, arXiv:2504.10458.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, and 401 others. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, and 16 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *Preprint*, arXiv:2501.12326.

Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. 2021. Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2023a. Android in the wild: A large-scale dataset for android device control. *Preprint*, arXiv:2307.10088.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy P Lillicrap. 2023b. Androidinthewild: A large-scale dataset for android device control. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. 2019. Habitat: A platform for embodied ai research. *Preprint*, arXiv:1904.01201.

Chan Hee Song, Valts Blukis, Jonathan Tremblay, Stephen Tyree, Yu Su, and Stan Birchfield. 2025. Robospatial: Teaching spatial understanding to 2d and 3d vision-language models for robotics. *Preprint*, arXiv:2411.16537.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *Preprint*, arXiv:2409.12191.

Shuai Wang, Weiwen Liu, Jingxuan Chen, Yuqi Zhou, Weinan Gan, Xingshan Zeng, Yuhan Che, Shuai Yu, Xinlong Hao, Kun Shao, Bin Wang, Chuhan Wu, Yasheng Wang, Ruiming Tang, and Jianye Hao. 2025. Gui agents with foundation models: A comprehensive survey. *Preprint*, arXiv:2411.04890.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. 2024. Os-atlas: A foundation action model for generalist gui agents. *Preprint*, arXiv:2410.23218.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. 2025. OS-ATLAS: Foundation action model for generalist GUI agents. In *The Thirteenth International Conference on Learning Representations*.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. 2025. Aguvis: Unified pure vision agents for autonomous gui interaction. *Preprint*, arXiv:2412.04454.

Karmesh Yadav, Ram Ramrakhya, Santhosh Kumar Ramakrishnan, Theo Gervet, John Turner, Aaron Gokaslan, Noah Maestre, Angel Xuan Chang, Dhruv Batra, Manolis Savva, Alexander William Clegg, and Devendra Singh Chaplot. 2023. Habitat-matterport 3d semantics dataset. *Preprint*, arXiv:2210.05633.

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

Shenzhi Wang Zhangchi Feng Dongdong Kuang Yuwen Xiong Yaowei Zheng, Junting Lu. 2025. Easyr1: An efficient, scalable, multi-modality rl training framework. `https://github.com/hiyouga/EasyR1`.

Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. 2024. Robopoint: A vision-language model for spatial affordance prediction for robotics. *Preprint*, arXiv:2406.10721.

Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. 2024a. Android in the zoo: Chain-of-action-thought for gui agents. *Preprint*, arXiv:2403.02713.

Li Zhang, Shihe Wang, Xianqing Jia, Zhihan Zheng, Yunhe Yan, Longxi Gao, Yuanchun Li, and Mengwei Xu. 2024b. Llamatouch: A faithful and scalable testbed for mobile ui task automation. *Preprint*, arXiv:2404.16054.

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. 2025. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts. *Preprint*, arXiv:2506.02177.

Enshen Zhou, Jingkun An, Cheng Chi, Yi Han, Shanyu Rong, Chi Zhang, Pengwei Wang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, and Shanghang Zhang. 2025. Roborefer: Towards spatial referring with reasoning in vision-language models for robotics. *Preprint*, arXiv:2506.04308.

A Prompts for reasoning thought generation

Here are our prompts for generating reasoning thoughts.

Embodied Thought Generation Prompt

You are a robot in an unfamiliar environment. Now I want you to give the reason for your action.
Your action can be in the following list:

- Based on the image, predict the optimal location to move next to finish the task. Use the coordinates (x, y) (x is the pixel from left to right and y is the pixel from top to bottom) to indicate where you want to move to: `{"action_type": "move", "x": <position in horizontal (width)>, "y": <position in vertical (height)>}`.
- Turn left: `{"action_type": "turn_left"}`.
- Turn right: `{"action_type": "turn_right"}`.
- Turn around: `{"action_type": "turn_around"}`.
- Move the camera angle downward: `{"action_type": "look_down"}`.
- Based on the image, if you find the target and the target is close enough, please stop to indicate that you want to stop: `{"action_type": "stop"}`.

You will be given the view before you performed the action (which has a text label `"before"` on the bottom right), the action you chose, and the task.
This is the action you performed: `<action>`
This is the task: `<task/question>` (the picture and action is one of the steps to finish the task)
By inspecting the picture and the action performed, give a brief reason of this step. You should carefully inspect the environment and give your analysis for why to do such action rather than other actions.
If moving to a position, explain why moving to that position based on the current environment. Avoid generic reasons like “get closer to the target.”
NOTICES:

1. Coordinates are absolute coordinates (a center point defined by top-left and bottom-right coordinates).
2. If the action type is `"move"`, the point will be labeled as `"Next point"` in the before image.
3. Remember that you should give the answer from a first-person perspective and keep it around 60 words and in a single line.
4. Don’t limit yourself to begin with “I...”. try any other possible sentence structure(like the position of exchangeing description and target) if not influence the meaning."

GUI Thought Generation Prompt

You are an agent who can operate an Android phone on behalf of a user. Now I want you to give the reason for your action.
You will be given the screenshot before you performed the action (which has a text label "before" on the bottom right), the action you chose (together with the reason), and the screenshot after the action was performed (which has a text label "after" on the bottom right).
This is the action you picked: `<q_text>`
This is the task: `<task>` (The screenshots and action are one of the steps to finish the task)
This is the instruction: `<instruction>` (The instruction to solve the task)
This is the related apps: `<apps>` (Apps in the reason you output cannot go beyond the range of the app list)
By comparing the two screenshots and the action performed, give a brief reason of this step. The reason should include the detailed description for the action and the target to do so, but avoid any description related to the after screenshot.
Requirements:

- Use first-person perspective.
- Keep the response around 60 words and in a single line.
- Do not begin every sentence with "I"; feel free to vary the structure as long as the meaning remains clear.

B Prompts for training

Here is our prompts for training NaviMaster.

Spatial-referring Prompt

Your answer should be formatted as a tuple, i.e. `[x, y]`, where the tuple contains the x and y coordinates of a point satisfying the conditions above. Output the thinking process in `<think> ... </think>` tags, and the final answer in:
`<answer>["action": "moveto", "point": [x, y]]</answer>`
Note: The coordinates should be between the size of picture, indicating the absolute pixel locations of the points in the image.
Example:
`["action": "moveto", "point": [123, 300]]`

Navigation Prompt

"You are a Navigation Robot in an unfamiliar environment. In this photo <image>, the task is '{text}', with the history being '{history}'
You need to use your prior knowledge about where items are typically located within a home.
Please predict next action to find target item.
Your action can be in the following list:
Basic Action(move to a point on the ground in the picture):
- Based on the image, predict the optimal location to move next to finish the task, use the coordinates (x, y)(x is the pixel from left to right and y is the pixel from top to bottom) to indicate where you want to move to:
`[{"action": "moveto", "point": [x(position in horizontal(width)), y(position in vertical(height))]}].`
View Adjustment Actions(adjust view as current photo does not have suitable position):
- Executes a 90-degree rotation to the left from the current facing direction. Ideal for navigating around obstacles on the right, aligning with a leftward path, or adjusting the view to inspect the left side of the environment. Use this when the task requires a lateral shift to the left:
`[{"action": "turn_left"}].`
- Rotates the perspective 90 degrees to the right. This action is useful when the target object or destination is positioned on the right, or when you need to change the direction to follow a rightward route:
`[{"action": "turn_right"}].`
- Performs a 180-degree rotation, flipping the orientation to face the opposite direction. This is valuable for finding a possible way if there is no path in front of you:
`[{"action": "turn_around"}].`
- Adjusts the camera view to look downwards, without physically moving the position. This is particularly useful for examining details on the ground, such as identifying objects, reading markings, or inspecting lower-level structures:
`[{"action": "look_down"}].`
Stop Action:
- Carefully inspect the environment and judge from history, if you find your target in your view and has been close enough for about 1 meter, stop at current position:
`[{"action": "stop"}].`
Output the thinking process in <think></think> tags, and the final answer in <answer></answer> tags as follows:
<think>... </think> <answer>answer here </answer>
Note: The 'point' should contain the coordinates of the next destination. Coordinates are absolute coordinates(a center point defined by top-left and bottom-right coordinates). Ensure the predicted Example:
`{"action": "moveto", "point": [123, 300]}`

GUI Prompt

A conversation between User and Assistant. The user asks a question, and the Assistant solves it step by step. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer.
At each step, you will be given the current screenshot and the history of the conversation (include screenshot and action in each step). Based on these pieces of information and the goal, you must give the whole content of what you think and then choose to perform one of the actions in the following list (action description followed by the JSON format) by outputting the action in the correct JSON format.
Click/tap on an element on the screen. We have defined the width and height of the screenshot, use the coordinates (x, y) (x is the pixel from left to right

and y is the pixel from top to bottom) to indicate which element you want to click, both x and y are integers:
`[{"action": "click", "point": [x(position in horizontal(width)), y(position in vertical(height))]}]`
Long press on an element on the screen, similar with the click action above, use the coordinates (x, y) to indicate which element you want to long press:
`[{"action": "long_press", "point": [x(position in horizontal(width)), y(position in vertical(height))]}]`
Type text into a text field (this action contains clicking on the target field, typing in the text and pressing the enter), use the coordinates (x, y) to indicate which element you want to click, both x and y are integers:
`[{"action": "input_text", "text": <text_input>, "point": [x(position in horizontal(width)), y(position in vertical(height))]}]`
Navigate to the home screen:
`[{"action": "navigate_home"}]`
Navigate back:
`[{"action": "navigate_back"}]`
Scroll the screen or a scrollable UI element from start point to end point, use the coordinates (x, y) to indicate the two points you want to scroll:
`[{"action": "scroll", "start_point": [<start position in horizontal(width)>, <start position in vertical(height)>], "end_point": [<end position in horizontal(width)>, <end position in vertical(height)>]}]`
NOTICES: 1.Coordinates are absolute coordinates (a center point defined by top-left and bottom-right coordinates). 2.The reasoning process and answer are enclosed within `<think> ...</think>` and `<answer> ...</answer>` tags, respectively. Example:
`<think> reasoning process here </think> <answer>["action": "click", "point": [378, 871]]</answer>`

C Action Space

In our trajectory, the action space $\mathcal{A}_{gui}$ for GUI task is defined as:

$$\mathcal{A}_{gui} = \begin{cases} \textbf{CLICK}\ (x,y) \\ \textbf{SCROLL}\ (x,y)\ TO\ (x',y') \\ \textbf{LONGPRESS}\ (x,y) \\ \textbf{TYPE}\ [TEXT]\ (x,y) \\ \textbf{NAVIGATEHOME} \\ \textbf{NAVIGATEBACK} \end{cases} \tag{12}$$

The action space A_{emb} for embodied task is defined as:

$$\mathcal{A}_{emb} = \begin{cases} \textbf{MOVETO}\ (x,y) \\ \textbf{TURNLEFT} \\ \textbf{TURNRIGHT} \\ \textbf{TURNAROUND} \\ \textbf{LOOKDOWN} \\ \textbf{STOP} \end{cases} \tag{13}$$

D Training Hyperparameters

To ensure the fairness of all comparative and ablation experiments, we maintained consistent hyperparameter settings throughout the training process, as detailed in Table 5.

Table 5: Hyperparameter settings used for all reinforcement learning training.

Hyperparameter	Value
learning_rate	from 1e-6 to 0
temperature	1.0
num_generations	5
num_train_epochs	10
max_prompt_length	7000
max_response_length	1024
per_device_train_batch_size	4
gradient_accumulation_steps	16
KL coefficient	0.01
Reward coefficients λ_1, λ_2, λ_3	0.1, 1, 1

E GUI Metrics Details

In GUI task, we follow the settings in OS-Atlas, where a correct type prediction is considered accurate if the predicted action type matches the ground truth. For predictions involving grounding, an action is deemed correct if the predicted location falls within 14% of the image size relative to the ground truth.

F Embodied Benchmark and Metrics Details

For details of benchmark:

Where2Place. This benchmark contains 100 real-world images to evaluate free space referring.

RoboSpatial. There are three branch in the benchmark:“Configuration”, “Context” and “Compatibility”. We take the “Context” branch to test free space referring.

RefSpatial. We take the unseen set of RoboSpatial. This set comprises 77 samples from the Location/Placement task.

Roborefit. We take the testA set of Roborefit. For metrics, we introduce the average success rate of predicted points with in the groundtruth mask to evaluate the spatial grounding accuracy in the spatial referring task. This metric directly assesses the model's ability to accurately localize the target based on the natural language description. For the navigation task, consistent with prior works, we utilize Success Rate (SR) and Success Rate Weighted by Inverse Path Length (SPL) as our metrics. SR measures the percentage of episodes that are successfully completed. Here, we set the success threshold to 0.3, meaning that stopping within this distance from the goal will be considered a success. SPL is a measure of navigation path efficiency, which quantifies the agent's performance by considering both task success and the path efficiency relative to the optimal path.

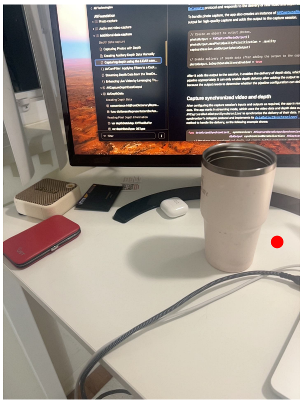
G Case Study

Fig.7 illustrates the spatial referring task, where the agent must local the coordinates in the image according to the given instruction. Our case set comprises three free space referring instances and one object referring instance.

Fig.8 illustrates the Embodied Navigation task, where the agent must follow the instruction to complete an entire navigation episode in Habitat.

Fig.9 illustrates the AC-High task, where the agent receives a high-level user instruction and, based on the history and the current smartphone screenshot, outputs the action for the current step.

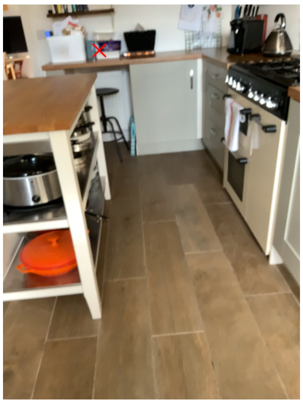
In the image, there is a cup. Pinpoint point within the vacant space situated to the right of the cup.



Thought: To locate points in the vacant space to the right of the cup, I need to determine a point that is entirely to the right of the cup and within the designated area mentioned. The cup and the wires in front of it create a clear boundary for the vacant space of interest. Visually, the area to the right of the cup forms a rectangular space, and I can identify points within this space.

Action: **MOVETO** (1322, 1141)

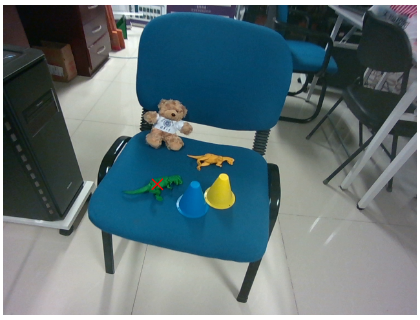
Please point out the object between the white box and the farthest black pot.



Thought: The question asks for the object between the white box and the farthest black pot. The white box is located on the left side, and the farthest black pot appears to be on the right side, further back in the image. The object between these two is the blurry object in the middle, which appears to be a kitchen appliance or container

Action: **MOVETO** (123, 58)

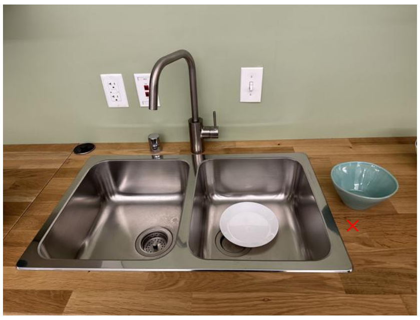
Grab the green dinosaur model



Thought: The green dinosaur model is located on the left side of the chair, below the teddy bear and to the left of the two plastic cones.

Action: MOVETO (237, 276)

Locate one spot within the vacant area that is in front of the teal bowl.

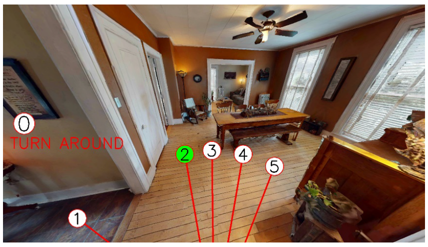


Thought: The teal bowl is located towards the right side of the image. There is a vacant area in front of the teal bowl, which is the space between the bowl and the edge of the countertop. I need to identify one point within this vacant area.

Action: MOVETO (539, 340)

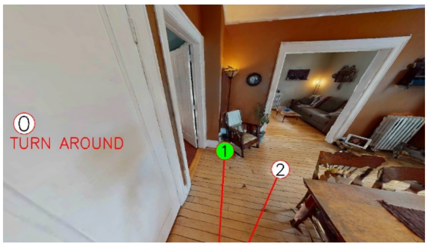
Figure 7: Case studies in RefSpatial, RoboSpatial, RoboReflt and Where2Place.

Navigate to the nearest sofa



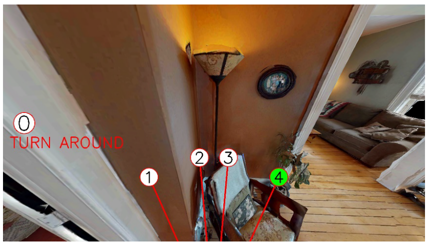
Thought: From the image, I can see a wooden floor, a dining area with a table and chairs, a wall with a window and blinds, a door, and a room in the background. The dining area is likely near the central area of the house, and sofas are typically found in living rooms, which can be inferred from the hallway leading to the dining area. Since I see the dining area and a hallway leading further, I would guess that the living room is likely the next room, which may contain the sofa. The sofas are usually found in living rooms, not dining rooms or entrances. To get to the nearest sofa, I should proceed down the hallway, as the living room area is presumably the next room. I should avoid taking the actions that lead through closed doors or into areas that would take me away from the living room. The path to the living room is indicated by the hallway directly ahead, and I should move towards that direction. The labeled action that aligns with moving further down the hallway goes to position 2.

Action: **MOVETO ②**



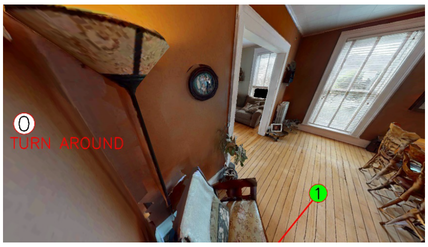
Thought: In your observation, you can see a hallway with a couch and other pieces of furniture in the room ahead. The red arrow labeled 1 points to a chair, and the red arrow labeled 2 points to a general area near what appears to be the edge of the room, possibly a wall or corner. Since the task is to navigate to the nearest sofa, I should head towards the area labeled 1, as it is closer to the room with the sofa. The action that achieves this goal best is to move forward in the direction of the labeled 1.

Action: **MOVETO ①**



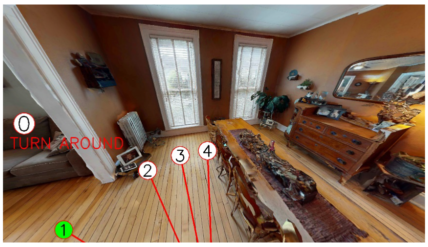
Thought: In the image, there is a living room area visible with a sofa on the right side. The sofa appears to be made of fabric and is partially covered with a blanket or throw. The red arrows and numbers in the image are pointing to different areas, but the sofa is clearly visible and within proximity of the highlighted area. Based on the visual information, the sofa is to the right of the robot's current position. The red arrow labeled "4" points closest to the sofa, indicating the direction to go for the best approach. Therefore, the robot should navigate to the right to get closer to the sofa.

Action: **MOVETO ④**



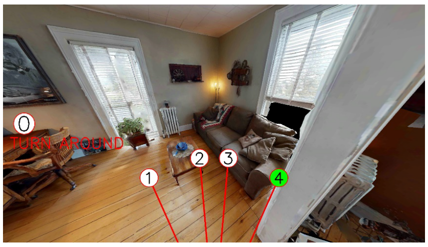
Thought: Based on the image, I see an indoor scene with a sitting area in the background. There are several pieces of furniture visible, but the couch is not immediately in view. The red arrow labeled '1' points towards the living area where the couch is likely to be located. The other text instructs me to turn around, which I will do to face the living area where the couch is presumably located.To achieve the task of navigating to the nearest sofa, the best action would be to move in the direction pointed towards by the red arrow labeled '1'. This corresponds to moving forward into the living area.

Action: **MOVETO ①**



Thought: In the observation, the room contains a sofa on the left side near the entrance, a table with framed pictures, and a dining table on the right. The sofa is the closest visible piece of furniture to the entrance. To navigate to the nearest sofa and get as close to it as possible, I should move towards the left side of the image.The action labeled as 1 is the one that corresponds to the sofa's location. Therefore, taking action 1 will lead me towards the sofa.

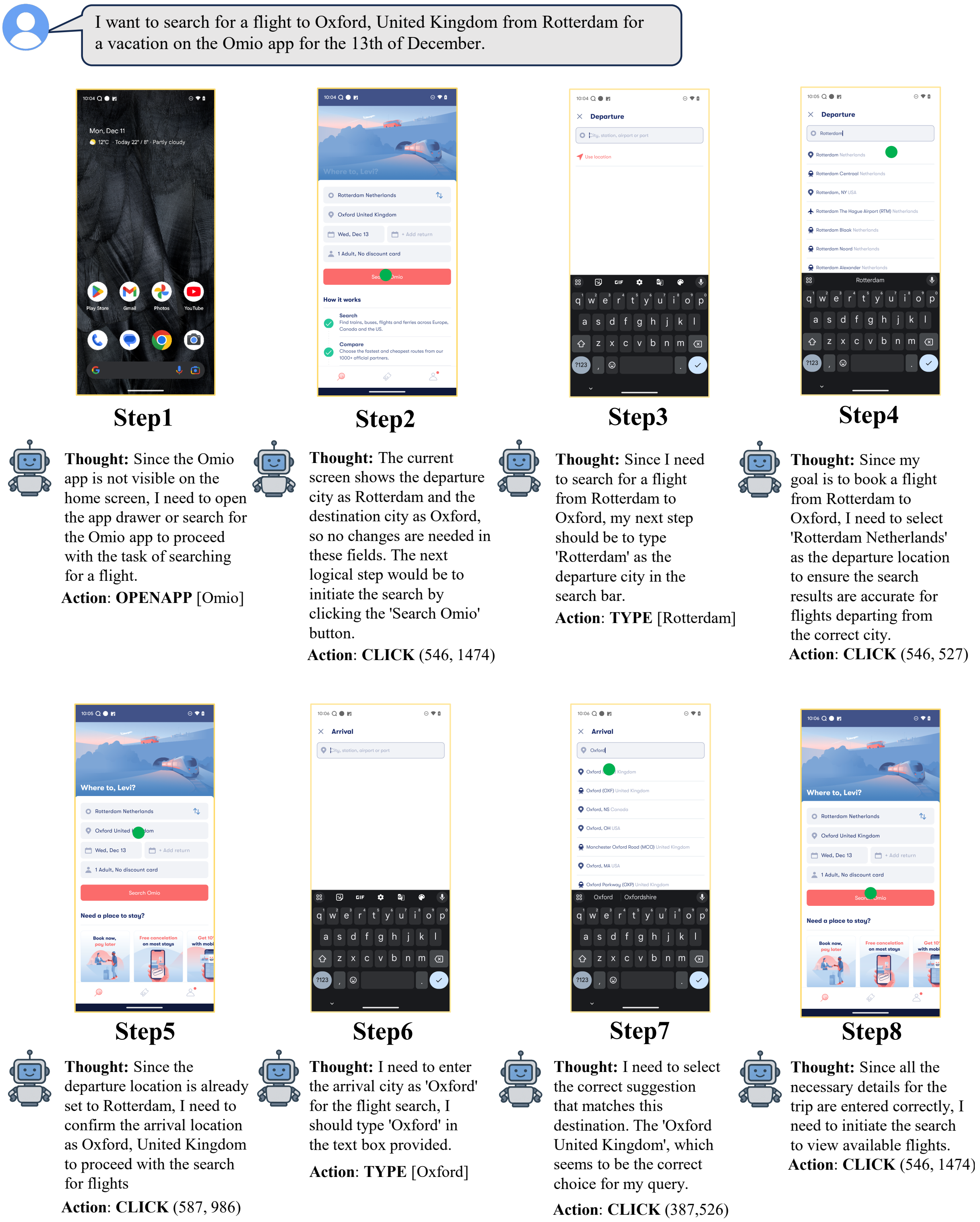
Action: **MOVETO ①**



Thought: In the image, I see a living room with a section of a sofa on the right. The sofa is partially visible, and it's covered with a blanket and some pillows. The room has a window with blinds, a small table in the center, and some wall decor. The area is spacious and well-lit with natural light. The action arrows indicate possible movement paths, but since we are looking for the nearest sofa, let's focus on area 3 and 4.Action 4 seems to be leading towards the sofa, making it the best choice for approaching the nearest sofa. Therefore, my response is: {'action': 4}.This action will take me towards the nearest sofa.

Action: **MOVETO ④**

Figure 8: Case study in ObjectNav.



I want to search for a flight to Oxford, United Kingdom from Rotterdam for a vacation on the Omio app for the 13th of December.

Step1

Thought: Since the Omio app is not visible on the home screen, I need to open the app drawer or search for the Omio app to proceed with the task of searching for a flight.

Action: **OPENAPP** [Omio]

Step2

Thought: The current screen shows the departure city as Rotterdam and the destination city as Oxford, so no changes are needed in these fields. The next logical step would be to initiate the search by clicking the 'Search Omio' button.

Action: **CLICK** (546, 1474)

Step3

Thought: Since I need to search for a flight from Rotterdam to Oxford, my next step should be to type 'Rotterdam' as the departure city in the search bar.

Action: **TYPE** [Rotterdam]

Step4

Thought: Since my goal is to book a flight from Rotterdam to Oxford, I need to select 'Rotterdam Netherlands' as the departure location to ensure the search results are accurate for flights departing from the correct city.

Action: **CLICK** (546, 527)

Step5

Thought: Since the departure location is already set to Rotterdam, I need to confirm the arrival location as Oxford, United Kingdom to proceed with the search for flights

Action: **CLICK** (587, 986)

Step6

Thought: I need to enter the arrival city as 'Oxford' for the flight search, I should type 'Oxford' in the text box provided.

Action: **TYPE** [Oxford]

Step7

Thought: I need to select the correct suggestion that matches this destination. The 'Oxford United Kingdom', which seems to be the correct choice for my query.

Action: **CLICK** (387,526)

Step8

Thought: Since all the necessary details for the trip are entered correctly, I need to initiate the search to view available flights.

Action: **CLICK** (546, 1474)

Figure 9: Case study in AC-High.