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

Zhihao Luo[1,2], Wentao Yan[1], Jingyu Gong[1], Min Wang[3], Zhizhong Zhang[1], Xuhong Wang[2]✉, Yuan Xie[1], Xin Tan [1,2]✉
[1]East China Normal University, [2]Shanghai AI Laboratory, [3]SenseTime Research
luozhihao@stu.ecnu.edu.cn, wangxuhong@pjlab.org.cn, xtan@cs.ecnu.edu.cn

Abstract

Recent advances in Graphical User Interface (GUI) and embodied navigation have driven 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 unifying 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 using a single formulation. (ii) employs a unified reinforcement learning framework on the mix data to improve 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 demonstrate the efficacy of our unified training strategy, data mixing strategy, and reward design. Our codes, data, and checkpoints are available at https://iron-boyy.github.io/navimaster-page/.

1 Introduction

Graphical user interface (GUI) navigation agents and embodied navigation agents are designed to traverse virtual and physical environments, respectively. Recent advances in multimodal large language models (MLLMs) (Bai et al., 2025; Jin et al., 2025) have enabled the integration of their strong perception and planning abilities for both agents (Wu et al., 2025; Lin et al., 2025). Leveraging these capabilities, such agents have shown substantial potential in instruction-guided multimodal navigation tasks.

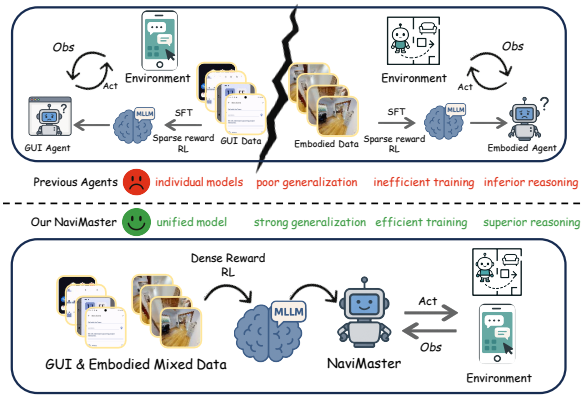


Figure 1: Previous methods involve individual models for GUI and embodied navigation. Our NaviMaster is a unified learning framework.

Despite the progress of previous agents, as illustrated in Fig. 1, the long-term separation between GUI and embodied navigation, coupled with their training strategies, has resulted in four persistent challenges. (1) These approaches employ two individual models for navigation, which increases training and deployment costs and precludes synergistic interaction between the two tasks (Hong et al., 2025). (2) Although prior works (Liu et al., 2025a; Rawles et al., 2023; Ramakrishnan et al., 2021) have improved performance in respective tasks by scaling data within specific task data, they exhibit limited cross-task performance due to poor generalization to 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. (4) Current RFT reasoning models often generate correct thoughts but wrong actions, as their "understanding" is primarily distilled from texts rather than visual observations.

To address these challenges, we propose a unified policy that integrates GUI and embodied navigation with an efficient training strategy. Inspired by VIS-Bench (Yang et al., 2025a), humans convert egocentric perceptions into an allocentric men-

*✉ Corresponding authors: wangxuhong@pjlab.org.cn, xtan@cs.ecnu.edu.cn

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 38595–38615
July 2-7, 2026

tal map to support perspective-taking and spatial reasoning. Similarly, both GUI and embodied navigation operate purely on egocentric visual observations, without direct access to a global state. Despite differences in surface-level action semantics, the two tasks are therefore isomorphic at the level of perception and decision-making: in both cases, the agent must integrate partial, first-person observations over time to implicitly construct a latent representation. We therefore characterize the unified problem as learning an egocentric-to-allocentric cognitive transformation. On the other hand, from the perspective of Markov Decision Processes (MDPs):

$$\arg\max_{a \in \mathcal{A}} P(\mathcal{S}_{t+1} \mid \mathcal{S}_t = \sigma, \mathcal{A}_t = a), \qquad (1)$$

where next state $\mathcal{S}_{t+1}$ depends solely on the current state–action pair (σ, a). Under our unified formulation, the state $\mathcal{S}_t$ corresponds to the egocentric visual observation at step t, while the action space $\mathcal{A}$ encompasses interactions in either virtual interfaces or physical environments. The transition dynamics satisfy the Markov property in both domains. Consequently, we formalize GUI and embodied navigation as a single *Navigation Agent* problem.

To this end, as shown in Fig. 1, NaviMaster incorporates three key advancements (1) We propose a trajectory collection pipeline that unifies both GUI and embodied navigation within a visual-target paradigm, enabling joint training on mix data and improving generalization. (2) We build a unified reinforcement learning training framework applicable to both navigation types. Policies optimized on a single MDP tend to overfit to task-specific correlations. By unifying GUI and embodied navigation under a distribution over MDPs, we enable the policy to learn generalizable structural representations such as visual object permanence, relative spatial reasoning, and affordance grounding. Specifically, we extend the training strategy to estimate task-specific advantages, enabling a single policy to adapt effectively across multiple tasks. Furthermore, the framework leverages prior reasoning steps and actions as historical context. Given the history and current observations as inputs, the model predicts the next action. This formulation unifies the I/O representation and enables the history to guide precise high-level actions in long-horizon navigation. (3) We employ a distance-aware dense reward in the reinforcement learning framework, which enhances training efficiency compared to a sparse binary reward.

We evaluate NaviMaster on OOD GUI and embodied navigation benchmarks. On test sets that differ significantly from the training domain, NaviMaster outperforms state-of-the-art baselines, achieving superior results across multiple datasets. These findings demonstrate strong generalization capability and robustness to distributional shifts. In summary, our key contributions are as follows:

1. We propose NaviMaster, the first unified navigation agent that jointly handles both GUI and embodied navigation within one framework.
2. We develop a visual-target trajectory collection pipeline that aggregates high-quality trajectories from both GUI and embodied navigation, thereby increasing data diversity and enhancing model generalization capability.
3. We design a distance-aware dense reward and a unified reinforcement learning pipeline, which together enhance data efficiency and further strengthen model 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 have adopted the data-driven training paradigm such as 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 enhance 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., 2025b) incorporate reinforcement learning (RL) inspired by DeepSeek-R1. However, their scope remains restricted to GUI-only settings and lack the capacity to do embodied navigation.

Embodied navigation agents control physical or simulated agents to follow language instructions in 3D spaces (Zhang et al., 2026; Ji et al., 2026; Tian et al., 2025; Ji et al., 2025), requiring multimodal perception and long-horizon planning (Gao et al., 2024; Li et al., 2026; Wang et al., 2026). Analogous to GUI navigation tasks, works on embodied navigation typically employ a multi-stage SFT strategy on large datasets to adapt open-source MLLMs

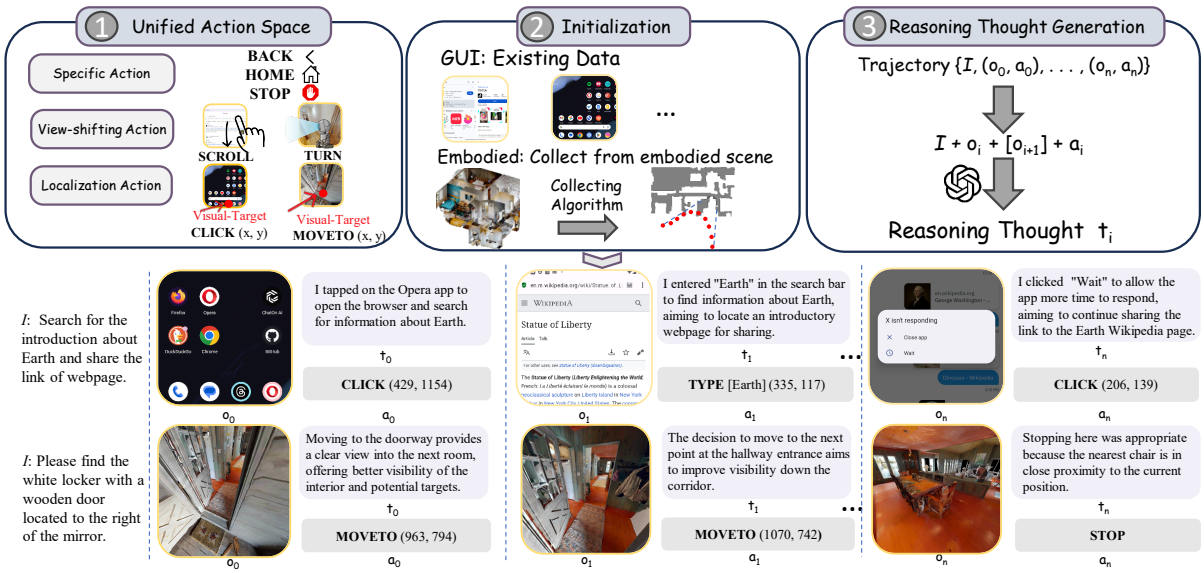


Figure 2: Visual-Target Trajectory Collection contains three parts. First, we unify the GUI and the Embodied action space by introducing a visual target at each step. Next, we initialize the trajectories using existing datasets or scenes. Last, we generate a first-person reasoning thought t_i with GPT-4o.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

for navigation tasks (e.g., RoboPoint (Yuan et al., 2025), SpaceLLaVa (Foutter et al., 2025)). Their scope remains restricted to a single domain, which enforces a monolithic action space, thereby limiting the agents' capacity to generalize when the action space changes.

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.

2.2 Reinforcement Fine-Tuning on MLLM

Visual-RFT (Liu et al., 2025c; Gao et al., 2026b) performs reinforcement fine-tuning on LVLMs using their own reasoning traces together with rule-based, verifiable visual rewards (Gan et al., 2026; Chen et al., 2025a). UI-R1 (Lu et al., 2025b) 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., 2025a). 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

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.

3.1 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 datasets exhibit substantial differences in their action spaces. We categorize actions into three types: specific action, view-shifting action, and localization action. First, specific actions have predefined, context-independent semantics (e.g., [**BACK**] in GUI, [**STOP**] in embodied

tasks) and are directly integrated into the unified action space. Second, view-shifting actions adjust the agent's viewpoint to locate targets outside the current field of view. This class encompasses actions like [**SCROLL**] in GUI and [**TURN**] in embodied environments. We standardize these transformations into four directions for each domain: [up, down, left, right] for GUI and [left, right, around, back] for embodied agents. Finally, localization actions show the greatest divergence. In GUI tasks, the localization action is performed through the [**CLICK** (x, y)] action, where (x, y) denotes a specific target position on the screenshot. In contrast, embodied navigation tasks achieve localization via the [**MOVEFORWARD**] action, which does not require an explicit target. These differences reflect distinct interaction paradigms between the two localization actions. GUI actions $\mathcal{A}_{gui}$ depend on precise, target-oriented operations (e.g., clicking specific UI elements), whereas embodied actions $\mathcal{A}_{emb}$ focus on egocentric motion control (e.g., navigating without explicit target selection).

The discrepancy in localization actions (i.e., with or without a target) creates a challenge for unifying both tasks. To address this challenge, we propose the visual-target trajectory by introducing a localization action with an explicit target into the embodied navigation task. As shown in the left part of Fig. 2, we define a visual target within the observation at each step of the trajectory. Consequently, the localization action in embodied navigation if reformulated from [**MOVEFORWARD**] to [**MOVETO** (x, y)], where (x, y) denotes target location. The complete action space is in Appendix C.

Trajectory Collection Initialization. We consider a long-horizon task consisting of n steps, represented as $\{I, (o_0, a_0), \ldots, (o_n, a_n)\}$, where I denotes the user-provided instruction, o_i is the observation from GUI screenshot or physical environment at step i, and a_i is the corresponding action at step i ($0 \leq i \leq n$). This representation is consistent with standard formulations of GUI trajectories, enabling direct reuse of existing GUI datasets. In our experiments, we leverage GUI-Odyssey (Lu et al., 2025a) to obtain trajectory data.

However, the existing embodied navigation datasets typically provide only the initial and the target positions without specifying the intermediate trajectories. For an embodied navigation dataset (e.g., Matterport 3D dataset with the Habitat simulator (Yadav et al., 2023; Savva et al., 2019)), we extract the set of trajectory points along the shortest path from the initial to the target position, denoted as $(\mathbf{s_0}, \mathbf{s_1}, \ldots, \mathbf{s_m})$, 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. The initialization procedure for trajectory collection is summarized in Algorithm 1. The first step is to align the next position with 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), $\mathbf{s'_{k+1}}(u'_{k+1}, v'_{k+1}, w'_{k+1})$ ($\mathbf{s_k}$ coordinate system) can be obtained with following equation:

$$\mathbf{s'_{k+1}} = \mathbf{r_k^{-1}} \times (\mathbf{s_{k+1}} - \mathbf{s_k}) \times \mathbf{r_k}. \quad (2)$$

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

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

where $p(x_i, y_i)$ represents the pixel 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: [**TURN** left], [**TURN** right], [**TURN** around], [**TURN** down] and implement them as Algorithm 1. Let w'_{k+1} denote the depth of s'_{k+1} in the current observation. A negative value ($w'_{k+1} < 0$) indicates that s'_{k+1} lies behind the camera. After getting the observation o_i with the target at each position, we represent each embodied navigation trajectory with the same action space and style as in GUI navigation.

Reasoning Thought Generation. Historical information has been shown to be beneficial for agent performance (Yang et al., 2025b). Most existing approaches (Xu et al., 2025) take the implemented actions as history, which may lead to ambiguity. For example, the action [**CLICK** (x, y)] does not specify the context or purpose of the interaction. By contrast, pairing the reasoning thought "I should first open Chrome to start my search" with the corresponding action "[**CLICK** (x, y)]" explicitly

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**
    Calculate $\mathbf{s'_{k+1}}(u'_{k+1}, v'_{k+1}, w'_{k+1})$
    while $p(x_i, y_i) \notin [0, W] \times [0, H]$ **do**
      $o_i \leftarrow$ observation at (s_k, r_k)
      Update $p(x_i, y_i)$
      if $w'_{i+1} < 0$ **then**
        $a_j \leftarrow$ **TURN** [around]
      else if $x_i < 0$ **then**
        $a_i \leftarrow$ **TURN** [left]
      else if $x_i > W$ **then**
        $a_i \leftarrow$ **TURN** [right]
      else if $y_i > H$ **then**
        $a_i \leftarrow$ **TURN** [down]
      end if
      Append (o_i, a_i) to Trajectory
      Update $r_k \leftarrow$ execute a_i ; $i \leftarrow i + 1$
    end while
    $a_i \leftarrow$ **MOVETO** (x_i, y_i)
    Append (o_i, a_i) to Trajectory ; $i \leftarrow i + 1$
  else
    $o_i \leftarrow$ observation at (s_k, r_k), $a_i \leftarrow$ **STOP**
    Append (o_i, a_i) to Trajectory
  end if
end for
return Trajectory

indicates that agent opens the app Chrome. Some works (Qin et al., 2025) demonstrates that augmenting each step in the trajectory with its associated reasoning allows the model to articulate its decision-making process more transparently.

To enhance reasoning capability and optimize memory usage, we generate thought for each action in the trajectory, as illustrated in the bottom part of Fig. 2. Given an initialized trajectory, we construct the data generation pipeline as follows:

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

where the task instruction I, observation o_i, and action a_i are provided to the large language model $\mathcal{M}$, which produces an intention t_i from a first-person perspective to explain the rationale behind action a_i. In our experiments, $\mathcal{M}$ corresponds to GPT-4o (OpenAI et al., 2024). The optional observation o_{i+1} is included only in GUI trajectories, where the target observation serves as a reference for data generation. The prompts used to generate these reasoning thoughts are provided in Appendix A.

Consequently, the visual-target trajectory for both GUI and embodied navigation tasks is represented as: $\tau = \{I, (o_0, t_0, a_0), \ldots, (o_n, t_n, a_n)\}$.

3.2 Unified Reinforcement Learning Framework

Since reinforcement learning (RL) generally exhibits stronger generalization capabilities than supervised fine-tuning (SFT) , we adopt the R1-Zero training strategy, directly training on our collected dataset with Group Relative Policy Optimization (GRPO). As illustrated in Fig. 3, given an n-step trajectory, we take step i as a data sample. Each sample consists of the user instruction I, the current observation o_i, reasoning thoughts and actions in history $H_i = \{(t_0, a_0), (t_1, a_1), ..., (t_{i-1}, a_{i-1})\}$. NaviMaster then learns a unified policy with GRPO. Specifically, for the 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; for pure 2-D images h_i is set to the zero matrix. The advantage Adv is computed as follows:

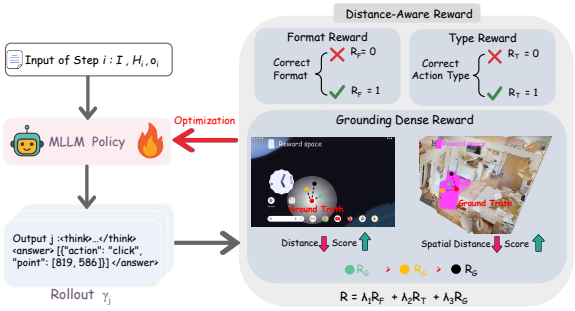


Figure 3: Overview of unified reinforcement learning framework. MLLM policy is optimized using GRPO with format, type and grounding dense reward.

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

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

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

3.3 Distance-Aware Reward

The criteria for successful task execution are threefold: (1) the model output must be correctly parsed into an executable action, (2) the type of the executed action must match the ground truth, (3) the action's arguments must fall within valid bounds. Accordingly, we decompose the reward into three components: format, type, and grounding. While most existing reward designs employ binary success/failure signals, our approach aims to capture relative preferences even among unsuccessful 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. Therefore, our reward consists 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 provide a reasoning phase, followed by a final answer, where the answer must be a valid JSON string. The required structure is: "$\langle think\rangle \cdots \langle /think\rangle \langle answer\rangle$ $json\ string$ $\langle /answer\rangle$". 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 $R_T(i,j) = [\hat{a}_j = a_i]$ evaluates the correctness of the model's action selection. $\hat{a}_j$ denotes the predicted action type for sample γ_j and a_i denotes the ground truth action type at step i. $[\cdot]$ stands for the Iverson bracket, an indicator that equals 1 when the statement inside is true and 0 otherwise. It is a binary reward that assesses whether the predicted action type matches the ground truth within a small, discrete action space. It supervises the model's ability to make high-level decisions aligned with task semantics.

Grounding Dense Reward. This reward R_G is designed to guide model's grounding ability. Specifically, this ability requires selecting the correct target within a large selection space, such as a pixel-level coordinate within an image. It evaluates the predicted location relative to the ground truth at step i. To consistently measure grounding performance across navigation tasks, we define a distance-based dense reward instead of a sparse reward. It ensures that the agent receives higher rewards when its prediction is closer to targets (UI elements or position in embodied scene). Such design can provide effective guidance during training. This reward function is as follows:

$$R_G(i,j) = (1 - \frac{d_j}{\theta_d})[d_j < \theta_d, p_j < \theta_h] \quad (7)$$

where θ_d and θ_h are thresholds for distance d_j and depth disparity p_j. The d_j denotes the pixel-level distance between predicted point $(\hat{x}_j, \hat{y}_j)$ of γ_j and the corresponding ground truth point (x_i, y_i). In embodied environments, the depth value $h_i(\hat{x}_j, \hat{y}_j)$ is also incorporated to account for potential occlusions: two pixels that are close in the 2D image may have substantially different depths in the 3D scene. The details of thresholds are in Appendix H. The definitions of d_j and p_j are as follows:

$$d_j = \sqrt{(\hat{x}_j - x_i)^2 + (\hat{y}_j - y_i)^2}, \quad (8)$$

$$p_j = |h_i(\hat{x}_j, \hat{y}_j) - h_i(x_i, y_i)|, \quad (9)$$

The overall reward function is a weighted combination of the three components described above. Here, $R(i,j) = \lambda_1 R_F(i,j) + \lambda_2 R_T(i,j) + \lambda_3 R_G(i,j)$. $\lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}_+$ controlling their relative importance of each term.

4 Experiments

4.1 Implementation Details

We trained our model using the EasyR1 framework (Zheng et al., 2025b), adopting Qwen2.5VL-7B model (Bai et al., 2025) as the base model. The training is conducted for three epochs on 8 NVIDIA A800 GPUs, with a global batch size of 128 and a learning rate of 1×10^{-6}. The hyperparameters λ_1, λ_2, and λ_3 are experimentally set to 0.1, 1, and 1, respectively. The training utilizes 20k samples, including 10k GUI samples from GUI-Odyssey and 10k embodied samples from Matterport 3D and RoboPoint. All the experiments of NaviMaster use the same amount of data for a fair comparison. More details are provided in Appendix D.

4.2 Benchmarks and Metrics

GUI task. For the evaluation of the GUI task, we employ seven distinct mobile, web, desktop and in-domain benchmarks: AC-High/Low (Li et al., 2024), AITW (Rawles et al., 2023), GUIAct-P/W (Chen et al., 2025b), Llamatouch (Zhang et al., 2024b), AITZ (Zhang et al., 2024a) OmniAct-W/D (Kapoor et al., 2024) and Odyssey (Lu et al., 2025a). We follow OS-Atlas (Wu et al., 2025) to take the success rate (SR) and grounding (GR) as the primary evaluation metrics. SR measures the per-step task success rate, while GR measures the accuracy of localizing the correct click coordinates.

We compare our model with the following methods: the proprietary GPT-4o (OpenAI et al., 2024), SFT-based models such as OS-Atlas, Aguvis (Xu et al., 2025) and Qwen2.5VL-7B* fine-tuned on our trajectory data, as well as RL-based models like GUI-R1 (Luo et al., 2025), infiGUI-R1 (Liu et al., 2025b), UI-Shift (Gao et al., 2026a) and UI-AGILE (Lian et al., 2025). We note the concurrent work of OmniActor (Yang et al., 2026), which also proposes a unified agent for both 2D and 3D tasks. However, we exclude it from comparison because it is evaluated on a subset of benchmarks and its implementation is not publicly available.

Embodied task. We assess our model's performance through two distinct embodied tasks. First, we conduct spatial affordance prediction to

Models	Mobile												Web				Desktop		ID	
	AC-Low		AC-High		AITW		GuiAct-P		Llamatouch		AITZ		GuiAct-W		OmniAct-W		OmniAct-D		Odyssey	
	GR	SR	GR	SR	GR	SR	GR	SR	GR	SR	GR	SR	GR	SR	GR	SR	GR	SR	GR	SR
GPT-4o	38.67	28.39	30.90	21.19	35.75	26.07	26.15	26.79	46.32	38.77	27.31	20.99	45.02	41.84	42.79	34.06	63.25	50.67	14.17	5.36
Qwen2.5VL-7B	87.08	62.50	59.71	47.06	65.99	45.80	55.79	44.99	75.84	60.97	60.79	33.88	90.78	78.08	84.40	71.91	79.42	57.58	68.52	37.32
Qwen2.5VL-7B*	66.73	43.82	51.29	31.79	62.51	48.81	52.21	47.75	53.88	45.56	58.36	31.18	60.26	43.26	47.61	44.36	50.58	42.29	40.96	29.38
OS-Atlas-7B	73.37	50.94	54.90	29.83	64.89	41.38	58.52	29.43	59.25	30.11	58.30	27.58	75.61	57.02	69.35	59.15	62.87	56.73	39.74	26.96
Aguvis	76.56	57.55	72.09	49.96	69.57	53.06	63.39	42.66	75.08	60.41	62.61	36.15	37.18	27.66	54.27	48.29	26.00	23.31	-	-
inhGUI-3B	93.20	92.10	74.40	71.10	**75.58**	46.51	**66.67**	40.09	75.02	58.66	75.42	40.81	86.02	64.60	75.42	54.91	73.25	47.17	70.30	33.15
GUI-R1-7B	84.02	66.52	70.31	51.56	62.94	55.31	47.09	48.62	69.82	61.27	64.98	49.06	88.06	74.54	84.87	68.69	81.19	57.70	71.10	36.22
UI-shift	94.22	73.38	73.41	52.16	74.44	54.38	59.10	52.12	75.08	61.71	73.84	46.78	89.49	79.43	82.93	64.22	80.01	57.94	62.54	32.75
UI-AGILE	93.71	63.32	77.55	50.97	75.72	48.66	60.11	42.27	78.31	66.10	75.89	38.74	90.33	69.57	84.04	63.15	81.94	59.35	70.40	36.96
Ours (w/o Embodied)	**94.40**	67.98	77.85	53.47	71.31	58.05	60.67	52.34	78.68	63.79	77.57	52.56	89.96	83.68	85.00	72.63	79.96	61.47	70.08	46.38
Ours (w/o GUI)	81.42	64.97	29.47	30.05	74.95	49.17	76.08	47.55	81.72	64.03	76.81	37.37	91.56	77.80	66.07	51.70	72.09	51.35	64.75	33.92
Ours	93.90	**69.46**	**78.15**	**55.89**	74.92	**59.72**	64.39	**53.27**	**82.54**	**67.39**	**81.14**	**54.00**	**91.95**	**86.17**	**85.30**	**72.99**	**82.16**	**62.47**	**73.60**	**48.35**

Table 1: Results on different 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.

assess the model's spatial grounding ability on the metric of SR. Specifically, we employ RoboReflT (Lu et al., 2023) for object referring and Where2Place (Yuan et al., 2025), RoboSpatial (Song et al., 2025), RefSpatial (Zhou et al., 2025) for 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., 2025) and substitute the agent model in our experiments to assess the performance. We compare our model against the proprietary GPT-4o, open-source methods like Qwen2.5VL-7B, SpatialVLM like SpaceLLaVA (Foutter et al., 2025), the latest method RoboPoint-13B (Yuan et al., 2025). More details of benchmarks and metrics are in Appendix E and Appendix F.

4.3 Main Results

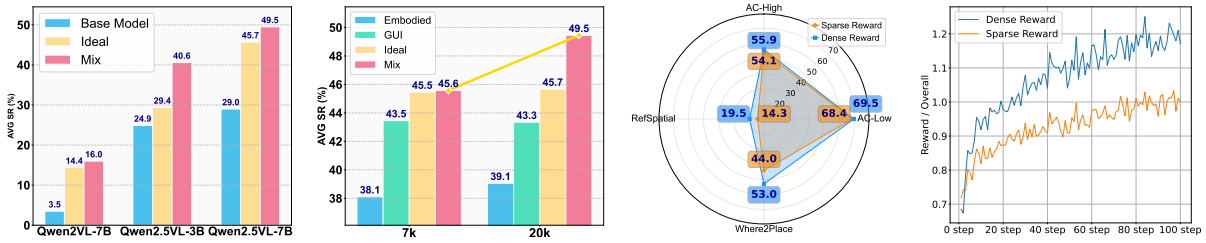
GUI Navigation. The results are shown in Table 1. To evaluate the generalization capability of NaviMaster, we employ entirely out-of-domain (OOD) test data, highlighted with a green background, which are disjoint from the training distribution. The remaining test cases, indicated with a red background, are in-domain. Compared with state-of-the-art baselines, NaviMaster demonstrates consistently superior performance across the various benchmarks, demonstrating strong generalization capability and robustness when handling OOD datasets. Additionally, compared to models trained solely on either GUI or embodied data, our model trained on the mix data achieves the highest performance across all test datasets. This demonstrates the effectiveness of our collected visual-target trajectory and unified training framework, which enable strong generalization and competitive performance with a relatively small amount of data.

Spatial Affordance Prediction. We evaluate two types of spatial affordance predicting datasets: one is object referring, which involves identifying and localizing specific objects within a scene based on language descriptions; the other is free space referring, which targets understanding and navigating to spatial regions or locations that are not necessarily tied to specific objects but are described in language. Table 2 summarizes the average success rate of predicted points falling within the ground-truth mask on the four spatial affordance prediction benchmarks. Compared to all baselines, NaviMaster performance best in all spatial affordance prediction tasks. These results demonstrate that NaviMaster's fine-grained visual–spatial alignment significantly enhances performance in both object-level and free-space referring.

Models	RoboRefIt	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	35.05	14.75	16.88
Ours (w/o GUI)	76.23	43.02	19.83	18.19
Ours	**77.34**	**52.97**	**21.65**	**19.49**

Table 2: Results on spatial affordance prediction.

Embodied Navigation. Since we are the first to train a model capable of generalizing in VLMNav, there are no prior navigation models trained under VLMNav for direct comparison. We only report our results in Table 3, which reports performance on the ObjectNav benchmark. NaviMaster achieves the highest SR of 33.10% and SPL of 12.60%, representing a substantial improvement over the base model. Training exclusively on embodied data or GUI data yields slightly lower SR, indicating that the mix training strategy effectively leverages the complementary advantages of both data sources.

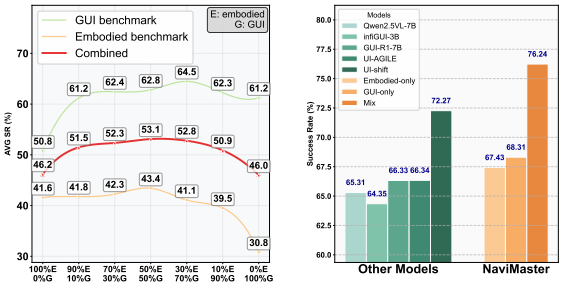


(a) (b) (c) (d)

Figure 4: Ablation Studies. (a) Performance of different base models. (b) Performance of different data scales. (c) Performance of dense and sparse reward. (d) Reward curves.

Runs	SR	SPL
Qwen2.5VL-7B	27.23	9.68
RoboPoint	13.01	2.80
SpaceLLaVA	14.98	2.62
Ours (w/o GUI)	31.10	11.20
Ours (w/o Embodied)	31.00	10.05
Ours	**33.20**	**12.60**

Table 3: Results on embodied navigation.



(a) (b)

Figure 5: (a) Performance of different ratios of training data. (b) Performance on our visual-relation benchmark.

4.4 Discussions and Analysis

For all analysis experiments, we task AC-High/Low as GUI benchmarks and Where2Place, RefSpatial as Embodied benchmarks. More details and other ablation studies can be found in Appendix G.

Impact of data ratio. We investigate the impact of varying data mixing ratios on performance across the two tasks. As illustrated in Fig. 5a, we evaluate four distinct ratios (0:10, 1:9, 3:7 and 5:5) and report the average SR on both tasks. The results show that overall performance peaks at 5:5 ratio, demonstrating that our mixed training approach enhances cross-domain generalization. Notably, even with imbalanced mixtures, models trained on mix data outperform those trained on a single dataset.

Advantages of mixing. As stated in the introduction, to evaluate the visual reasoning enabled by our unified-training strategy, we manually constructed a visual-relation benchmark consisting of 100 instances from the AndroidControl dataset. Each instance contains a landmark UI element and a final target defined by its spatial relationship to that landmark. Importantly, the instructions require the model to infer the target location from a relational description (e.g., "Click on the 2nd button above the PUSH-UPS") rather than directly naming the target. This design compels the model to reason over the image instead of merely learning a straightforward language-to-pixel mapping. As shown in Fig. 5b, the model trained with mix data achieves substantially higher accuracy than models trained on a single data type and also outperforms the existing RFT model with reasoning capabilities.

Base model. To verify that our improvements arise from unified training rather than a strong base model, we evaluated the framework on Qwen2.5VL-3B (smaller parameter scale) and Qwen2VL-7B (Wang et al., 2024) (less pre-training knowledge). Fig. 4a presents the average success rate for each mode. Here, 'Ideal' denotes the average success rates of two specialist models, each trained and evaluated solely within its respective domain (GUI or embodied). Training on mix data consistently outperforms training on single data, regardless of the underlying model.

Data scales. We investigate the impact of data scale on our mixture strategy's performance relative to single-task training. We evaluate two scales, 7k and 20k samples, while keeping the same training setting for all models. As shown in Fig. 4b, the mix-data approach consistently outperforms single-task training, demonstrating its benefits in both large-data regimes and relatively few samples.

Reward Design. We assess the effectiveness of the proposed grounding dense reward by replacing it with a sparse alternative. 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 predicted point's distance to 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. 4c 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. 4d rises

more rapidly, indicating more efficient training.

5 Conclusion

We introduce NaviMaster, the first agent unifying GUI and embodied navigation in a single RL framework by reformulating both tasks into a visual-target trajectory format. This unification enables joint training and cross-task generalization. We propose a distance-aware dense reward to improve learning efficiency and spatial grounding capability. Experiments demonstrate NaviMaster achieves superior OOD generalization over prior methods.

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.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (62302167, U23A20343, W2521174, and 62476090); in part by the Science and Technology Commission of Shanghai (25511103300, 25511104302, 23ZR1420400); in part by Young Elite Scientists Sponsorship Program by CAST (YESS20240780).

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.

Cong Chen, Kaixiang Ji, Hao Zhong, Muzhi Zhu, Anzhou Li, Guo Gan, Ziyuan Huang, Cheng Zou, Jiajia Liu, Jingdong Chen, Hao Chen, and Chunhua Shen. 2025a. Gui-shepherd: Reliable process reward and verification for long-sequence gui tasks. *Preprint*, arXiv:2509.23738.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, and 1 others. 2025b. Guicourse: From general vision language model to versatile gui agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21936–21959.

Matthew Foutter, Daniele Gammelli, Justin Kruger, Ethan Foss, Praneet Bhoj, Tommaso Guffanti, Simone D'Amico, and Marco Pavone. 2025. Spacellava: A vision-language model adapted to extraterrestrial applications. In *2025 IEEE Aerospace Conference*, pages 1–23. IEEE.

Guo Gan, Yuxuan Ding, Cong Chen, Yuwei Ren, Yin Huang, and Hong Zhou. 2026. Android coach: Improve online agentic training efficiency with single state multiple actions. *Preprint*, arXiv:2604.07277.

Longxi Gao, Li Zhang, Pengzhi Gao, Wei Liu, Jian Luan, and Mengwei Xu. 2026a. GUI-shift: Enhancing VLM-based GUI agents through self-supervised reinforcement learning. In *The Fourteenth International Conference on Learning Representations*.

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

Zhenkun Gao, Xuhong Wang, Xin Tan, and Yuan Xie. 2026b. Tpru: Advancing temporal and procedural understanding in large multimodal models. In *The Fourteenth International Conference on Learning Representations*.

Dylan Goetting, Himanshu Gaurav Singh, and Antonio Loquercio. 2025. End-to-end navigation with vision-language models: Transforming spatial reasoning into question-answering. In *International Conference on Neuro-symbolic Systems*, pages 22–35. PMLR.

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 Wang, and Kai-Wei Chang. 2025. Embodied web agents: Bridging physical-digital realms for integrated agent intelligence. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Yuzhou Ji, Qijian Tian, He Zhu, Xiaoqi Jiang, Guangzhi Cao, Lizhuang Ma, Yuan Xie, and Xin Tan. 2026. S2d: Sparse to dense lifting for 3d reconstruction with minimal inputs.

Yuzhou Ji, He Zhu, Junshu Tang, Wuyi Liu, Zhizhong Zhang, Xin Tan, and Yuan Xie. 2025. Fastlgs: Speeding up language embedded gaussians with feature grid mapping. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pages 3922–3930.

Yizhang Jin, Jian Li, Tianjun Gu, Yexin Liu, Bo Zhao, Jinxiang Lai, Zhenye Gan, Yabiao Wang, Chengjie Wang, Xin Tan, and Lizhuang Ma. 2025. Efficient multimodal large language models: a survey. *Visual Intelligence*, 3(1).

Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem AlShikh, and Ruslan Salakhutdinov. 2024. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. In *European Conference on Computer Vision*, pages 161–178. Springer.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

LinFeng Li, Jian Zhao, Yuan Xie, Xin Tan, and Xuelong Li. 2026. Compassnav: Steering from path imitation to decision understanding in navigation. In *The Fourteenth International Conference on Learning Representations*.

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. *Advances in Neural Information Processing Systems*, 37:92130–92154.

Shuquan Lian, Yuhang Wu, Jia Ma, Yifan Ding, Zihan Song, Bingqi Chen, Xiawu Zheng, and Hui Li. 2025. Ui-agile: Advancing gui agents with effective reinforcement learning and precise inference-time grounding. *Preprint*, arXiv:2507.22025.

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.

Bangwei Liu, Yicheng Bao, Shaohui Lin, Xuhong Wang, Xin Tan, Yingchun Wang, Yuan Xie, and Chaochao Lu. 2025a. Idmr: Towards instance-driven precise visual correspondence in multimodal retrieval. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6320–6329.

Yuhang Liu, Pengxiang Li, Congkai Xie, Xavier Hu, Xiaotian Han, Shengyu Zhang, Hongxia Yang, and Fei Wu. 2025b. 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. 2025c. Visual-rft: Visual reinforcement fine-tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2034–2044.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. 2025a. Guiodyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22404–22414.

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. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 976–983. IEEE.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanjing Xiong, and Hongsheng Li. 2025b. 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, and 1 others. 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 (Round 2)*.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy P Lillicrap. 2023. 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, and 1 others. 2019. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347.

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. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15768–15780.

Qijian Tian, Xin Tan, Yuan Xie, and Lizhuang Ma. 2025. Drivingforward: Feed-forward 3d gaussian splatting for driving scene reconstruction from flexible surround-view input. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 7374–7382.

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.

Sen Wang, Bangwei Liu, Zhenkun Gao, Lizhuang Ma, Xuhong Wang, Yuan Xie, and Xin Tan. 2026. Explore with long-term memory: A benchmark and multimodal llm-based reinforcement learning framework for embodied exploration.

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. 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. In *International Conference on Machine Learning*, pages 69772–69805. PMLR.

Karmesh Yadav, Ram Ramrakhya, Santhosh Kumar Ramakrishnan, Theo Gervet, John Turner, Aaron Gokaslan, Noah Maestre, Angel Xuan Chang, Dhruv Batra, Manolis Savva, and 1 others. 2023. Habitat-matterport 3d semantics dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4927–4936.

Jihan Yang, Shusheng Yang, Anjali W Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. 2025a. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10632–10643.

Longrong Yang, Zhixiong Zeng, Yufeng Zhong, Huang Jing, Liming Zheng, Lei Chen, Haibo Qiu, Zequn Qin, Lin Ma, and Xi Li. 2026. Omniactor: A generalist GUI and embodied agent for 2d&3d worlds. In *The Fourteenth International Conference on Learning Representations*.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, and 1 others. 2025b. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. In *International Conference on Machine Learning*, pages 70576–70631. PMLR.

Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. 2025. Robopoint: A vision-language model for spatial affordance prediction in robotics. In *Conference on Robot Learning*, pages 4005–4020. PMLR.

Jiwen Zhang, Jihao Wu, Teng Yihua, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. 2024a. Android in the zoo: Chain-of-action-thought for gui agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12016–12031.

Lechao Zhang, Haoran Xu, Jingyu Gong, Xuhong Wang, Yuan Xie, and Xin Tan. 2026. World2minecraft: Occupancy-driven simulated scenes construction. In *The Fourteenth International Conference on Learning Representations*.

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. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–13.

Haizhong Zheng, Yang Zhou, Brian R Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. 2025a. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

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

Enshen Zhou, Jingkun An, Cheng Chi, Yi Han, Shanyu Rong, Chi Zhang, Pengwei Wang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, and 1 others. 2025. Roborefer: Towards spatial referring with reasoning in vision-language models for robotics. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

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:

$$\mathcal{A}_{gui} = \begin{cases} \textbf{CLICK}\ (x, y) \\ \textbf{SCROLL}\ [up, down, right, left] \\ \textbf{LONGPRESS}\ (x, y) \\ \textbf{TYPE}\ [TEXT]\ (x, y) \\ \textbf{HOME} \\ \textbf{BACK} \end{cases} \tag{10}$$

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

$$\mathcal{A}_{emb} = \begin{cases} \textbf{MOVETO}\ (x, y) \\ \textbf{TURN}[left, right, around, down] \\ \textbf{STOP} \end{cases} \tag{11}$$

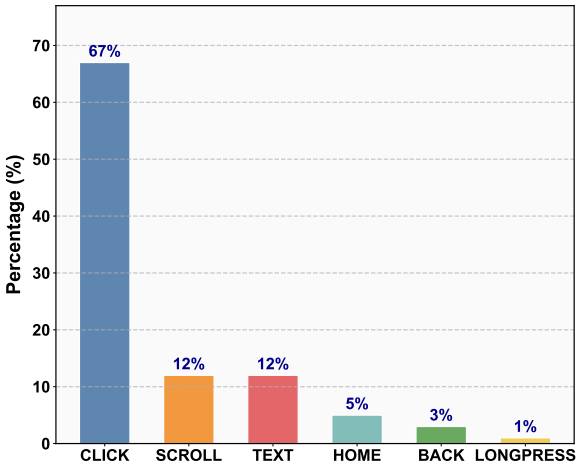


Figure 6: Data distribution of GUI-Odyssey

D Training Settings

D.1 Training Hyperparameter

To ensure the fairness of all comparative and ablation experiments, we maintained consistent hyperparameter settings throughout the training process, as detailed in Table 4.

Hyperparameter	Value
learning_rate	from 1e-6 to 0
temperature	1.0
num_generations	5
num_train_epochs	3
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

Table 4: Hyperparameter settings used for all reinforcement learning training.

In our implementation, whenever the concatenated trajectory exceeds 7000 tokens, we iteratively remove the minimum number of oldest steps until the prompt length drops just below the 7000 limit.

D.2 Sampling strategy

Selection bias could significantly affect the results. To avoid this, our sampling strictly follows the original distribution of GUI-Odyssey. We first computed the action-type distribution in the full 173k dataset. As shown in Fig. 6, when sampling the subset, we preserved this distribution, ensuring that the sampled data statistically matches the source dataset. Therefore, no additional bias was introduced during the sampling process.

E GUI Benchmark and Metrics Details

All the GUI benchmark are the test set of open-source dataset. The in-domain testing is from our GUI data source. We sample 4800 cases to test our in-domain performance.

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.

Here, we argue that the metric of Type (the accuracy of the predicted action type) is unreliable due to dataset bias. As shown in Table 5, a naive model that predicts all actions as [**CLICK**] can still achieve a high Type prediction accuracy. Despite this limitation, we report Type results for completeness in Table 6.

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

Table 5: The bias of action types in testing datasets.

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 Detailed Analysis

G.1 Statistical Stability and Error Statement

Our results are based on multiple runs and are empirically stable. All reported results are averaged over multiple trials, and to reduce variance during inference, we set the vLLM (Kwon et al., 2023) temperature to 0. Moreover, the success criteria for our tasks are range-based rather than relying on a single exact output. Therefore, even if model predictions vary slightly due to stochasticity, the evaluation outcome is generally unaffected.

G.2 Detailed statistics

The detailed results of base model ablation, data scales ablation, data scale ablation and reward design ablation is shown in the Table.7.

G.3 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 Table.7, Navimaster-robopoint the embodied data source only from affordance data. Navimaster-navi represents model embodied data are only from trajectory. Results show that combining both sources under an equal total data volume yields the best training performance.

G.4 Data Usage

Regarding dataset utilization, there are primarily two strategies. One is to mix different types of data 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). Details were shown in the Table.7, Navimaster-1gui2embodied means training on GUI data at stage 1 and then on embodied data for stage 2. Navimaster-1embodied2gui represents model trained on embodied data first and then on GUI data. The average results shown in Fig. 7, the mix training strategy generally outperforms the two-stage training strategy across various benchmarks. This suggests that training with mix

	AC-Low	AC-High	AITW	GuiAct-P	Llamatouch	AITZ	GuiAct-W	OmniAct-W	OmniAct-D	Odyssey
Ours (w/o Embodied)	81.16	69.50	71.47	71.10	88.21	64.25	91.21	95.25	95.79	78.54
Ours (w/o GUI)	80.47	71.45	53.15	51.44	90.15	45.29	83.62	93.82	90.15	71.30
Ours	81.08	73.88	67.47	68.71	86.38	61.79	92.13	94.06	95.62	77.51

Table 6: TYPE results on different GUI tasks.

Models	AC-High	AC-Low	Where2Place	RefSpatial
Qwen2.5VL-3B	64.57/34.43	85.58/58.78	2.96	3.36
Navimaster-3B	66.02/45.26	79.85/60.41	14.83	15.59
Navimaster-3B(w/o GUI)	45.89/28.75	75.21/56.25	7.00	7.79
Navimaster-3B(w/o Embodied)	45.89/28.75	75.21/56.25	7.00	7.79
Qwen2VL-7B	19.11/2.30	17.98/7.24	3.00	1.30
Navimaster-qwen2vl	24.18/14.43	31.54/18.81	22.95	7.79
Navimaster-qwen2vl(w/o GUI)	19.37/13.04	26.46/15.71	15.98	9.09
Navimaster-qwen2vl(w/o Embodied)	23.53/13.87	30.65/18.74	15.01	1.30
Navimaster-7k	76.90/52.66	93.85/70.41	41.07	18.19
Navimaster-7k(w/o GUI)	26.86/29.06	77.78/63.37	44.94	15.08
Navimaster-7k(w/o Embodied)	74.83/52.34	94.13/69.48	37.73	14.28
Navimaster-1gui2embodied	74.87/54.83	94.19/70.09	47.77	18.82
Navimaster-1embodied2gui	74.55/54.73	93.64/71.04	47.97	19.21
Navimaster-hardreward	76.19/54.07	92.56/68.39	44.01	14.28
Navimaster-robopoint	73.32/52.34	93.94/68.94	52.04	19.67
Navimaster-navi	73.64/52.44	93.90/69.37	47.99	16.49
Navimaster-1_1_1	75.03/53.20	94.20/69.31	49.96	20.14
Navimaster-0.1_1_2	77.33/54.98	93.70/69.19	47.06	22.14
Navimaster-0.1_2_1	76.04/54.10	94.06/70.72	46.06	23.34

Table 7: Detailed Statistics

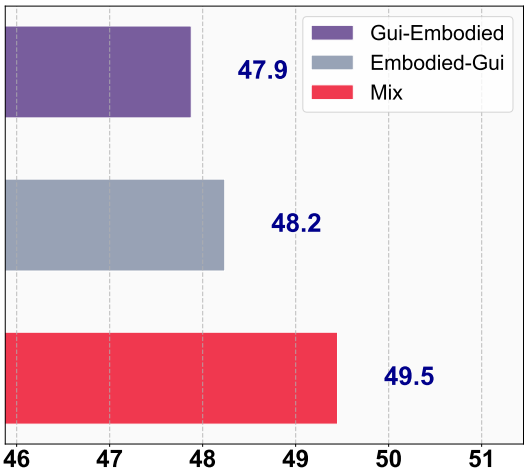


Figure 7: Performance of different data usage strategies.

data in a single phase enables the model to exploit complementary information effectively.

G.5 Hyperparameters

The ablation on hyperparameters focuses on the three reward weights, λ_1, λ_2, and λ_3. We conducted three additional experiments with configurations different from the main setup: NaviMaster_0.1_1_2 ($\lambda_1 = 0.1$, $\lambda_2 = 1$, $\lambda_3 = 2$), NaviMaster_0.1_2_1 ($\lambda_1 = 0.1$, $\lambda_2 = 2$, $\lambda_3 = 1$), and NaviMaster_1_1_1 ($\lambda_1 = 1$, $\lambda_2 = 1$, $\lambda_3 = 1$). As shown in Table 7, the configuration used in our main experiments achieves the best overall performance among these variants.

G.6 Reward Components

We attribute the performance improvements to gains in both navigation accuracy and instruction adherence. To decouple these factors, we provide the reward components ablation study in Table 8. It shows that removing the format or type reward leads to a drop (1–2 points) across all navigation benchmarks.

Reward	AC-High	AC-Low	Where2Place	RefSpatial
NaviMaster	55.89	69.46	52.97	19.49
w/o format reward	54.02	68.95	52.02	20.77
w/o type reward	55.66	70.60	48.99	18.18

Table 8: Ablation of Reward Components

H Discussion of Threshold

H.1 Standardized High Resolution

We did not use low-resolution images (e.g., 224×224 or 480×640) for embodied observations. Instead, we maintained a high-resolution standard: the minimum GUI resolution is 1280×720, and all embodied observations are uniformly resized to 1980×1080. This decision was deliberate, as prior works have shown that using high-resolution images during training and inference significantly enhances the model's crucial grounding capability.

H.2 Rationale for Absolute Threshold

We chose an absolute threshold over a relative one to prevent severe reward hacking. If a relative threshold were used, on larger-resolution images, the acceptable error range would scale proportionally, making the success condition much easier. This would lead to many localization-inaccurate rollouts being incorrectly deemed "successful", which would seriously mislead the model's training process.The absolute threshold of 200 pixels avoids this issue, ensuring that only highly accurate samples receive a positive reward, thereby preventing incorrect samples from skewing the training, even if it makes successful rollouts marginally more challenging on high-resolution data.

H.3 Alignment with Established Metrics

Furthermore, the $\theta_d = 200$ pixels threshold is not arbitrary; it is consistent with widely accepted evaluation metrics in the GUI community. Success in GUI tasks is commonly judged by whether the predicted action is within a certain area (e.g., $\mathbf{14}\%$) of

the ground truth. Given our minimum resolution of 1280×720, a 200-pixel radius circle aligns almost perfectly with this standard:

$$\pi \times (200^2)/(1280 \times 720) \approx 14\%$$

H.4 Training with relative threshold

We are prepared to conduct additional experiments using a relative threshold for reward definition. Specifically, we would normalize the model prediction and ground truth to a $0 - 1000$ range and set the reward threshold to $\mathbf{140}$ for training. The Table 9 shows the advantage of an absolute reward threshold.

threshold	AC-High	AC-Low	Where2Place	RefSpatial
absolute	55.89	69.46	52.97	19.49
relative	53.61	67.56	51.99	18.48

Table 9: Comparison of relative and absolute threshold

H.5 Ablation of threshold θ_d

We provide experiments in Table 10, which evaluates different distance thresholds to further validate the robustness of our formulation.

θ_d	AC-High	AC-Low	Where2Place	RefSpatial
100 pixel	55.06	69.85	48.88	19.48
200 pixel	55.89	69.46	52.97	19.49
300 pixel	55.77	69.60	48.14	16.88
400 pixel	53.61	68.58	51.06	16.68

Table 10: Ablation of threshold θ_d

H.6 Ablation of threshold θ_h

We clarify that the introduction of depth into the reward signal is primarily intended for robustness, not for a direct enhancement of policy performance, thereby reducing the reliance on sensor priors as the main source of improvement. If the reward were based purely on 2D distance from the image projection, it would fail in corner cases—for example, when an object far away and an object nearby appear spatially aligned and close in the 2D image. Such misleading 2D calculations necessitate the introduction of depth to compute the true 3D distance, ensuring a robust and accurate proximity reward signal. We have conducted an ablation study on the effect of the depth signal on the overall performance in Table 11. These results confirm that the presence of the depth signal does not lead to a significant performance boost (it is not a primary source of gain). However, setting an overly strict depth threshold (θ_h) severely limits the reward availability and degrades performance, which supports our claim that depth's role is to ensure robustness by preventing misleading 2D projection errors, not to artificially inflate the success rate.

θ_h	AC-High	AC-Low	Where2Place	RefSpatial
w/o θ_h	55.09	69.46	52.16	19.49
0.1 m	51.56	67.98	36.84	16.88
1 m	55.89	69.46	52.97	19.49
10 m	55.89	69.46	52.97	19.49

Table 11: Ablation of threshold θ_h

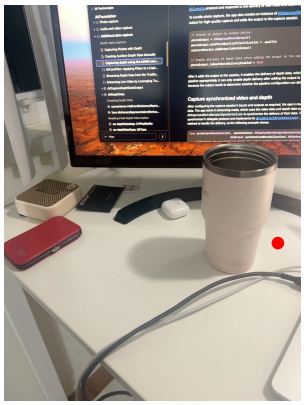
I Case Study

Fig. 8 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. 9 illustrates the Embodied Navigation task, where the agent must follow the instruction to complete an entire navigation episode in Habitat.

Fig. 10 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. Fig. 11 illustrates the general task, where the agent receives an user instruction and face both GUI and embodied environments. It outputs the thoughts and 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 8: 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.

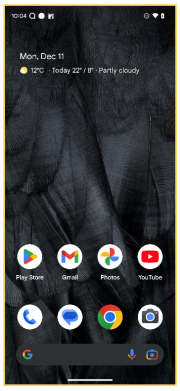
Action: **MOVETO ①**

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 9: Case study in ObjectNav.

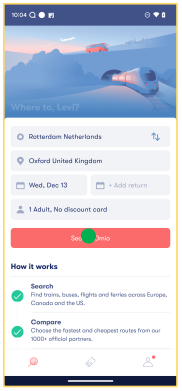
I want to search for a flight to Oxford, United Kingdom from Rotterdam for a vacation on the Omio app for the 13th of December.



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.

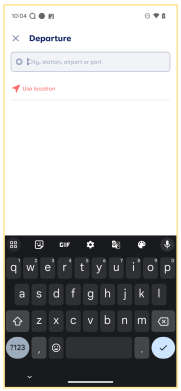
Action: **OPENAPP** [Omio]



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.

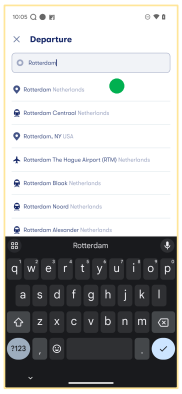
Action: **CLICK** (546, 1474)



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.

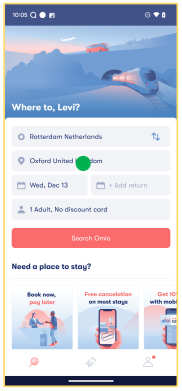
Action: **TYPE** [Rotterdam]



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.

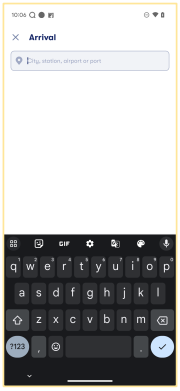
Action: **CLICK** (546, 527)



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

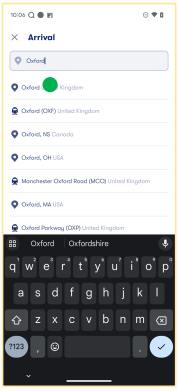
Action: **CLICK** (587, 986)



Step6

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

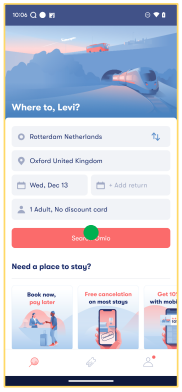
Action: **TYPE** [Oxford]



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)



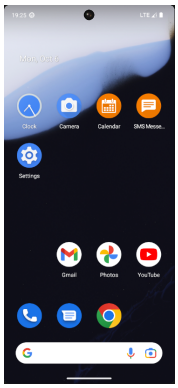
Step8

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 10: Case study in AC-High.

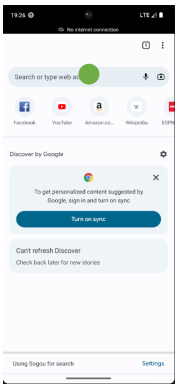
Please help me search on the browser for how to prune bonsai and navigate to the location of the plant.



Thought: I need to open the app drawer or search for the Chrome app to proceed with the task of searching for pruning bonsai.

Action: **OPENAPP** [Chrome]

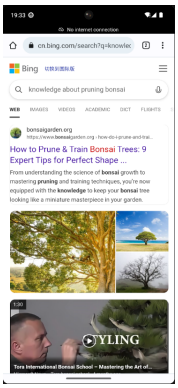
Step1



Thought: Now that I have entered the browser, I need to type "knowledge about pruning bonsai" in the search bar in the middle.

Action: **TYPE** [knowledge about pruning bonsai] (182, 153)

Step2



Thought: Since the image shows the search results for pruning bonsai, I have obtained the search results, and I should end the current task.

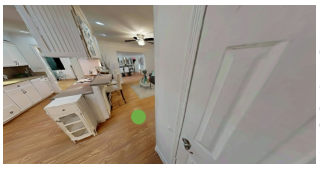
Action: **STOP**

Step3

Step4

Thought: To find the nearest plant, I should look for a location in the house where plants are typically kept. As the door appears to be closed, it won't allow for movement outside. I need to navigate back inside to search for a plant.

Action: **TURN** [left]



Step5

Thought: In the image, I see a kitchen area with white cabinets and a wooden floor. The plant is not visible in the immediate area, the best course of action is to move towards the corridor

Action: **MOVETO** (562, 473)

Step6

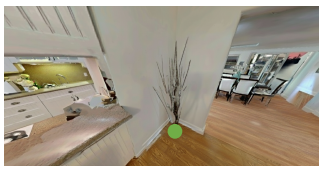
Thought: In the image, I observe a kitchen area and a living room. There are no plants in the immediate kitchen area. I need to move to living room, where a plant is more likely to be located.

Action: **MOVETO** (792, 460)

Step7

Thought: In the image, I see a living room with a sofa, a coffee table, and a painting of a horse on the wall. There are no visible plants in the image. To find the plant, I should move in the general direction of the dining area.

Action: **MOVETO** (240, 638)



Step8

Thought: There's a plant with tall dried stalks standing near the corner of the dining area, closer to the wall.To get as close as possible to the plant, I should move towards the dining area as it is the closest direction with a clear path to the plant.

Action: **MOVETO** (673, 541)

Step9

Thought: The image shows a close-up of a plant with long, thin leaves. The plant appears to be very close to the camera. I have finished the task.

Action: **STOP**

Figure 11: Case study in general task.