SimWorld-Robotics: Synthesizing Photorealistic and Dynamic Urban Environments for Multimodal Robot Navigation and Collaboration

Yan Zhuang[1] **Jiawei Ren**[2*] **Xiaokang Ye**[2*] **Jianzhi Shen**[3] **Ruixuan Zhang**[3]
Tianai Yue[3] **Muhammad Faayez**[3] **Xuhong He**[4] **Xiyan Zhang**[3]
Ziqiao Ma[5] **Lianhui Qin**[2‡] **Zhiting Hu**[2‡] **Tianmin Shu**[3‡]
[1]University of Virginia [2]UC San Diego [3]Johns Hopkins University
[4]Carnegie Mellon University [5]University of Michigan

Abstract

Recent advances in foundation models have shown promising results in developing generalist robotics that can perform diverse tasks in open-ended scenarios given multimodal inputs. However, current work has been mainly focused on indoor, household scenarios. In this work, we present SimWorld-Robotics (SWR), a simulation platform for embodied AI in large-scale, photorealistic urban environments. Built on Unreal Engine 5, SWR procedurally generates unlimited photorealistic urban scenes populated with dynamic elements such as pedestrians and traffic systems, surpassing prior urban simulations in realism, complexity, and scalability. It also supports multi-robot control and communication. With these key features, we build two challenging robot benchmarks: (1) a multimodal instruction-following task, where a robot must follow vision-language navigation instructions to reach a destination in the presence of pedestrians and traffic; and (2) a multi-agent search task, where two robots must communicate to cooperatively locate and meet each other. Unlike existing benchmarks, these two new benchmarks comprehensively evaluate a wide range of critical robot capacities in realistic scenarios, including (1) multimodal instructions grounding, (2) 3D spatial reasoning in large environments, (3) safe, long-range navigation with people and traffic, (4) multi-robot collaboration, and (5) grounded communication. Our experimental results demonstrate that state-of-the-art models, including vision-language models (VLMs), struggle with our tasks, lacking robust perception, reasoning, and planning abilities necessary for urban environments.

Project website: SimWorld-Robotics

1 Introduction

There has been tremendous progress in engineering general-purpose robotics that can follow human instructions and perform open-ended tasks [2, 28, 15, 27, 46], thanks to the advances in robot foundation models. Training these models requires a large amount of data, much of which can be generated in high-fidelity embodied simulators, such as Habitat 3 [40], RoboTHOR [10], TDW [14], VirtualHome [38], Virtual Community [61] and BEHAVIOR [27]. They can also be systematically evaluated in diverse scenarios created in these simulators. However, current embodied simulators for robotics have been focused on tabletop [35, 58, 34, 22, 59] or household tasks [48, 27, 26, 47, 46]. In this work, we want to study how to create a realistic and scalable embodied simulator for outdoor robotics tasks.

* Equal contribution. ‡ Equal advising.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

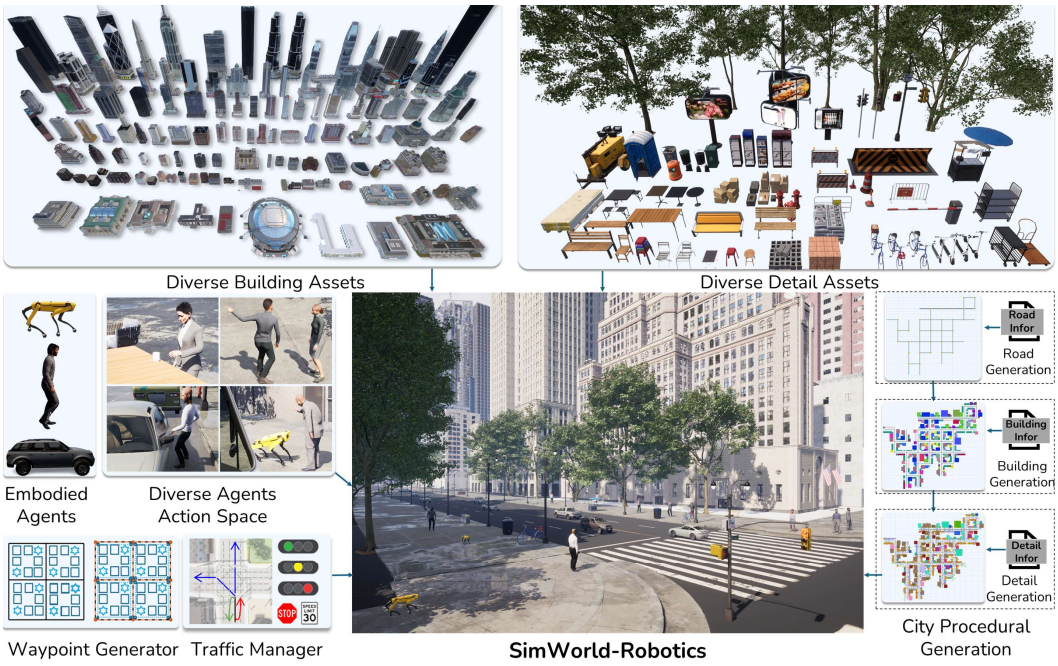

Figure 1: **Overview of SimWorld Robotics (SWR).** Built upon Unreal Engine 5, SWR is a simulation platform for large-scale, photorealistic, and dynamic urban environments. It offers diverse high-fidelity building and object assets, supports embodied agents with rich action spaces, includes a background traffic system powered by city-scale waypoint generation, and enables comprehensive city procedural generation.

Compared to indoor scenarios, robotics in outdoor environments, in particular, large urban environments, introduces additional challenges, such as (1) 3D perception, spatial reasoning and grounding in large environments; (2) safe navigation in dynamic scenes with people and traffic; (3) long-range spatial memory; and (4) multi-agent collaboration and communication in task.

There have been urban simulators developed in recent years. However, to address the critical challenges faced by real-world robotics in urban environments, they lack the necessary realism, customizability, scalability, and versatility. For instance, well-known simulators such as AirSim [45], CARLA [12] mainly focus on autonomous driving domains. While it supports some manual customization to the provided city environments, it does not support procedural city environment generation. It also does not support the flexible control of embodied agents (such as mobile robots or pedestrians) other than vehicles. More recent city simulators, such as MetaDrive [29], MetaUrban [56], significantly improve the scalability. However, the simulated environments still lack photorealism as shown in Figure 2.

Therefore, we introduce SimWorld-Robotics (SWR), a new embodied AI simulation platform for large-scale, photorealistic, and dynamic urban environments. As illustrated in Figure 1, SWR offers diverse high-fidelity building and object assets, multiple types of embodied agents with rich action spaces, a waypoint-based background traffic system, and a comprehensive procedural city generation pipeline to generate infinite cities. SWR also supports scalable data generation with fine-grained ground-truth annotations, enabling the training and evaluation of embodied agents at scale. Together, these features accelerate progress toward stronger embodied intelligence.

By leveraging SWR, we develop two novel benchmarks for robots in large, urban environments. Each benchmark evaluates crucial robot capabilities uniquely supported by the key features of SWR. As shown in 4, the first is a multimodal instruction following benchmark, SIMWORLD-MMNAV, for robot navigation, in which a robot must follow vision and language instructions to reach the target location. Unlike existing robot navigation benchmarks, we evaluate multiple robot capacities necessary for real-world urban navigation jointly, including robust 3D visual perception, grounding multimodal instructions to 3D environments, obstacle avoidance, following traffic rules, and walking around people in a socially acceptable way. The second is a multi-robot search benchmark, SIMWORLD-MRS, in which two robots must cooperate to localize and meet each

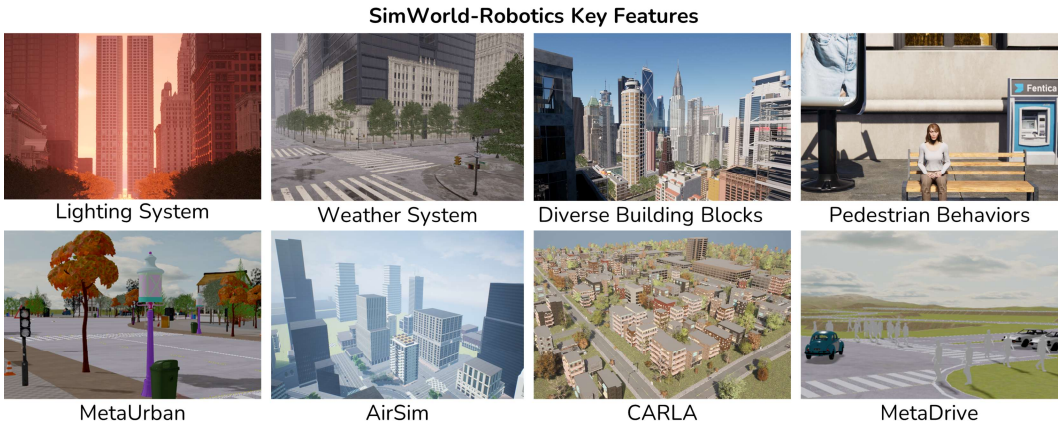

Figure 2: **Simulator Comparison** Top: Our simulator demonstrates key features including dynamic lighting (e.g., sunrise), realistic weather (e.g., rain), diverse high-fidelity buildings, and rich pedestrian behaviors. Bottom: MetaUrban and MetaDrive support large-scale maps but suffer from poor rendering quality; AirSim and CARLA offer better visuals but have limited building diversity and high repetition. All existing simulators exhibit simplistic pedestrian behaviors limited to random walking, failing to reflect real urban dynamics.

other via physical navigation and verbal communication illustrated in 5. To be successful in this benchmark, robots must be able to effectively communicate about each other's locations in a large, unfamiliar space and discuss a joint plan, combined with their 3D spatial reasoning abilities. Our experimental results demonstrate that existing models, including state-of-the-art vision-language models (VLMs), fail to achieve meaningful success on our benchmarks. This highlights the gap in current foundation models for challenging, realistic robot tasks in urban environments.

To address this gap, we introduce SimWorld-20K, a large-scale dataset for benchmarking multimodal robot navigation in photo-realistic urban environments. The dataset contains 20K training steps sampled from 200 episodes, each averaging $500\,\mathrm{m}$ in length, across 100 procedurally generated city environments with an average area of $2\,\mathrm{km}^2$. Compared to MetaUrban [56], the most recent urban simulator supporting procedural city generation, SWR offers environments that are $100\times$ larger in area and episodes that are over $1.2\times$ longer (MetaUrban: 410 meters per episode and $0.02\,\mathrm{km}^2$ on average). This enables evaluation of long-horizon, real-world-scale navigation. After fine-tuning on SimWorld-20K, QwenVL2.5-7B achieves a non-zero success rate on the test set and outperforms SOTA proprietary models across several key metrics.

In sum, our key contributions include: (1) a new embodied AI simulator, SimWorld-Robotics (SWR), that supports the creation and simulation of photorealistic and dynamic urban environments with diverse embodied agents; (2) two novel benchmarks for single robot navigation and multi-robot search tasks that evaluate key robot capacities by leverage the key, unique features of our simulator; (3) a large-scale training dataset, SimWorld-20K, that enables long-horizon multimodal robot navigation across city-scale environments; (4) a systematic evaluation of recent baseline models which identifies the significant limits of these models on the evaluated key capacities.

2 Related Work

Embodied Simulators for Urban Environments. Recent embodied simulators mostly focus on the indoor environment, mainly designed for household robots [7, 52, 38, 39, 14, 40, 23, 27]. There have been outdoor simulators [12, 45, 56, 16, 53, 13] that provide more open-ended tasks, facilitate more complicated dynamics, but face challenges in reality and multi-agent parallelism. CARLA[12] is a flexible simulator dedicated to autonomous driving. AirSim [45] focuses on simulation for drone control. EmbodiedCity [16] provides an arena for embodied AI with realistic urban settings and enriched agent control. Grutopia [53] offers a large-scale annotated scene dataset for general service robot tasks. AerialVLN [30] serves as a highly photorealistic urban landscape for UAV navigation. MetaUrban [56] features a wide range of micromobility tasks on the pedestrian, creating a highly complex urban environment. However, the dynamic scene interaction of human agents and social navigation (e.g., traffic rule following and pedestrian avoidance) has been much less explored.

Table 1: Comparison of outdoor simulation platforms across key features. The **Scenes** section includessupport for Procedural Generation (✓: supported, ×: not supported), and level of Photorealism. The **Human** section summarizes scene interaction capabilities (Scene-Interact.) and the number of supported human Actions. **Embodied Agents** indicate whether robots (Rob.), humans (Hum.), and vehicles (Veh.) are supported. **Multi-agent** denotes support for asynchronous multi-agent control. Full comparisons with other types of simulators are in Table 6 in Appendix A.2.

Simulator	Scenes		Human		Embodied Agents			Multi-agent
	Procedural Generation	Photorealistic	Scene-Interact.	Actions	Rob.	Hum.	Veh.	
CARLA [12]	×	✓	×	2	×	✓	✓	✓
AirSim [45]	×	✓	×	2	×	✓	✓	✓
MetaUrban [56]	✓	×	×	2	✓	×	×	✓
EmbodiedCity [16]	×	×	×	2	✓	×	×	×
Grutopia [53]	×	✓	×	×	✓	×	×	×
AerialVLN [30]	×	✓	×	×	✓	×	×	×
UnrealZoo [60]	×	✓	✓	10	✓	✓	✓	✓
SWR (Ours)	✓	✓	✓	26	✓	✓	✓	✓

Our simulator places the robot in a photorealistic, dynamically populated urban environment, evaluating not only multimodal instruction following but also the robots ability to adapt to real-time social cues and navigate amidst human agents in a safe, rule-aware manner.

Instruction Following Benchmarks for Robot Navigation. Following complex instructions in the real world often requires understanding both language and vision cues in tandem. As summarized in Table 8 in Appendix B.1, prior benchmarks on vision-and-language navigation have tackled instruction following, but typically in static indoor scenes or street panoramas using only textual descriptions[41]. For instance, Touchdown [8] explores city navigation via language alone in a static street view setting, and R2R [2] focuses on multi-step instructions in household tasks. These settings lack moving obstacles and additional visual guidance, making them only partial proxies for real-world urban navigation. In contrast, our multimodal robot navigation benchmark, SIMWORLD-MMNAV, challenges a robot to interpret and follow multimodal instructions (paired language instructions and visual hints) to reach the target location in a large-scale, photorealistic, dynamic city environment.

Multi-Robot Collaboration Benchmarks. There have been many recent multi-robot collaboration benchmarks as summarized in Table 9, but they do not evaluate multi-robot collaboration and communication for exploration and navigation in large urban environments. RoCo [33] studies cooperative manipulation on tabletop tasks, without the need for environmental exploration. DOROTHIE [32] introduces spoken dialogue for an ego vehicle, yet involves just one controllable agent. DriVLMe [20] simulates city-scale traffic with many vehicles, but lacks any verbal communication between robots. RobotSlang [4] evaluates robot communication but is restricted to small environments. *Where Are You?* [17] frames localization as a two-party dialogue, but the robots do not physically navigate to meet each other through a large environment.

3 SimWorld-Robotics

Built upon SimWorld [44], SimWorld-Robotics (SWR) introduces key extensions including procedural city generation, a traffic system, and support for an additional embodied agent: the quadruped robot. We begin by describing how SWR procedurally generates diverse and scalable urban environments with varying specifications. We then introduce the embodied agents supported in these environmentsvehicles, humans, and robotsand explain the logic behind asynchronously controlling multiple agents. Finally, we outline the rule-based system governing background pedestrians and traffic dynamics. Details of SWR can be found in Appendix A.

3.1 Procedural City Generation

The Procedural City Generation pipeline in SWR is designed to synthesize realistic, structured urban environments from minimal specifications, supporting tasks such as autonomous driving, pedestrian navigation, and multi-agent simulations. As shown in Figure 3, the pipeline follows four stages: road, building, street element, and traffic element generation. It begins by creating a road

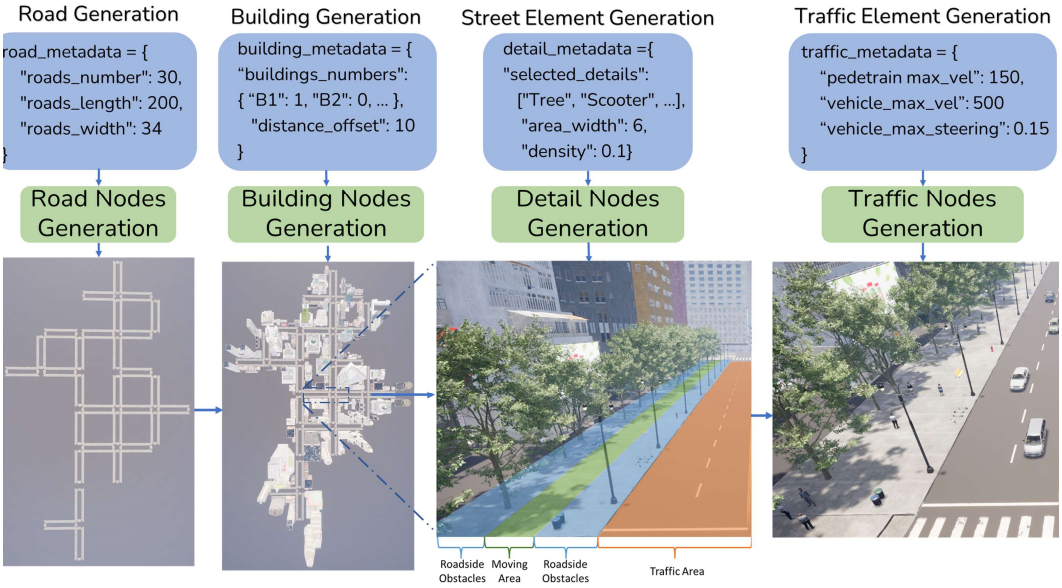

Figure 3: **Procedural City Generation.** SWR receives a user's specification and modularizes the process into road, building, details, and traffic elements generation.

network through a priority queue-based growth strategy that balances branching and depth, while ensuring plausible layouts via road-end attachment and intersection validation. Buildings are then placed along roads using collision-aware sampling and greedy gap-filling to maximize coverage and maintain uniformity. Next, contextual street elementstrees, cones, benches, and parked vehiclesare placed around buildings and sidewalks with basic accessibility constraints. Finally, dynamic traffic elements—including vehicles and pedestrians—are integrated into the environment to support research on traffic dynamics, agent-based behavior modeling, and realistic navigation scenarios. A detailed description of the city procedural generation pipeline is provided in Appendix A.6.

3.2 Embodied Agents

SWR supports three types of embodied agentshumans, vehicles, and robots. Unlike synchronous designs [38] where agents must wait for others to complete their actions, SWR allows asynchronous control, enabling each agent to act independently. To better reflect real-world conditions and support diverse tasks, SWR also provides rich action spaces for different agents and flexible observation spaces. The creation of new embodied agents in Appendix A.8.

Types of Embodied Agents. Unlike previous simulators [12, 56, 29]that focus on a specific type of embodied agent—such as autonomous vehicles, robots, or humans—and mainly support control over that particular agent type, our simulator simultaneously supports all three major categories. This unified design enables the development of broader and more diverse embodied AI tasks within a single environment. In SWR, we have included all three types of embodied agents. For the robots, we have two kinds: the scooter and the quadruped robot.

Asynchronous Multi-agent Control. To realistically model scenarios where multiple agents act independently and simultaneously, SWR uses an asynchronous multi-agent control framework. Each agent receives its observation from a centralized buffer and can submit an action only when marked as available. The buffer updates at a fixed interval (default: 0.01 seconds), checking for actions from available agents and updating their availability status. Valid actions are executed concurrently, after which all agents become unavailable until their actions complete. Once finished, agents are marked as available again and receive updated observations. The control pipeline is illustrated in Figure 7 in Appendix A.3.

Observation Space. SWR provides three primary types of visual observations: RGB images, depth images, and semantic segmentation masks, as illustrated in Figure 8 in Appendix A.4. In addition to these, SWR also offers ground-truth language descriptions and 3D bounding boxes for objects present in the environment.

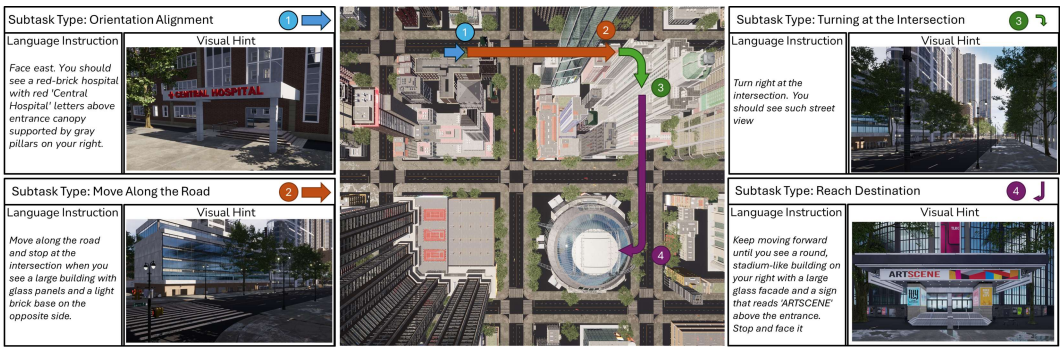

Figure 4: Illustration of a multimodal robot navigation task.

Action Space. SWR supports three types of continuous vehicle control: acceleration, braking, and steering. Each action is continuous within a set range, allowing flexible controle.g., higher acceleration leads to faster speeds, and larger steering values result in sharper turns. For robot control, actions include continuous translation (forward, backward, left, right) and free-angle rotation, enabling flexible movement and orientation, as illustrated in Figure 8 in Appendix A.5. Human agents can perform navigation actions (movement, turning) and interaction actions relevant to urban scenarios, grouped into: (1) humanobject (e.g., pick up/drop off objects, sit/stand), (2) humanvehicle (e.g., drive, enter/exit, open/close trunk), and (3) humanhuman (e.g., wave, argue, point).

3.3 Pedestrian and Traffic Simulation

The traffic simulation in SWR creates dynamic urban scenarios by orchestrating vehicle and pedestrian movement across a generated city map. It supports route assignment, intersection control, and pedestrian flow simulation, running on a fixed-time update loop for consistent real-time updates. Vehicle motion is governed by a feedback-based model using a PID controller, with empirically tuned parameters for realistic acceleration, braking, and turning dynamics [21]. Pedestrians follow a lightweight model, adjusting orientation incrementally toward their goals based on angular differences. To simulate realistic patterns, SWR uses a probabilistic routing strategy at intersections, where agents select paths based on predefined probabilities. This stochastic behavior introduces natural variability and enhances scene diversity. Details of predestrain and traffic simulation can be found in Appendix A.7.

4 Multimodal Robot Navigation Benchmark

We propose a novel multimodal robot navigation benchmark, SIMWORLD-MMNAV, where a robot must follow multimodal instructionspaired language and visual hintsto reach a target in a large-scale, photorealistic, dynamic city environment (Figure 4). This requires grounding verbal and visual references in the 3D environment based on the robots observations, while adapting to real-world complexities like traffic and pedestrians. SWR enables scalable evaluation of such tasks through (1) procedurally generated instructions with annotated, richly detailed city layouts, (2) photorealistic rendering of visual hints, and (3) simulation of pedestrians and traffic. We highlight core challenges of dynamic grounding: aligning language with visual targets, navigating around moving entities, and obeying environmental rulesall within an open-world urban setting that surpasses prior benchmarks. Details of SIMWORLD-MMNAV can be found in Appendix C.

4.1 Setup

Task Definition. Each task consists of a series of multimodal instructions $I_{1...K}$ that the robot must interpret and execute. As illustrated in Figure 4, each instruction I_k includes a natural language instruction and a visual hint illustrating what the robot can expect to see after reaching the goal described by the language instruction. These two modalities together guide the robot toward a specific location and/or orientation. Each instruction is associated with a ground-truth goal g_k. At

Table 2: Experimental results on the SIMWORLD-MMNAV benchmark (easy task set). The numbers in parentheses indicate the improvement after finetuning.

Models	SR% ↑	Subtask SR% ↑	Distance Progress% ↑
Proprietary Models			
GPT-4o	0	33.07	15.60
Gemini 2.5 Flash	0	37.06	31.29
Reasoning Models			
GPT-o3	5.0	42.50	38.43
GPT-o3-pro	8.3	46.35	39.46
Open-sourced Models			
QwenVL 2.5 7B	0	16.86	7.82
QwenVL 2.5 72B	0	23.80	17.50
Gemma 3 27B	0	15.36	6.83
InternVL 3 78B	0	18.31	9.34
Fine-tuned Models			
QwenVL 2.5 7B$_{ft}$	4.0 (+4.0)	52.45 (+35.59)	53.63 (+45.81)
Hybrid Baselines			
HybridGPT	0	32.53	27.24
RL-based Baselines			
VLA-RL	0	28.37	22.79

any given timestep, only one instruction is presented to the robot, which must successfully complete the current instruction (i.e., reaching g_k) before receiving the next instruction I_{k+1}.

In real-world urban navigation, instructions typically fall into one of four categories: Orientation Alignment, Move Along the Road, Turn at the Intersection, and Reach Destination. As shown in Figure 4, a complete sequential instruction-following task is formed by chaining together multiple different types of instructions. Details of procedural task generation can be found in Appendix C.3.

To closely mirror real-world navigation challenges, the environments simulated incorporate both static obstacles (e.g., buildings, barriers) and dynamic agents (e.g., pedestrians, vehicles) whose trajectories are not known in advance. Task difficulty is determined by the complexity of the environment, and we define two levels: an *easy* task includes no obstacles (static objects outside of buildings, pedestrians or vehicles), and a *hard* task adds both static object obstacles and dynamic pedestrians and vehicles.

Obsevation and Action Space. Apart from the subtask description and expected visual hint, the robot has access to the egocentric RGB image, segmentation image, and depth image. Ground truth orientation is provided, assuming a built-in compass in the robot. The robot's action space includes moving in four directions, as well as turning. It can also stay still and confirm task completion. SWR also supports active perception by providing different viewpoint images.

SimWorld-20k. Based on SWR, we construct the SimWorld-20k dataset, including 100 training maps with an average area of $2\,\mathrm{km}^2$ and 200 oracle trajectories generated by A* [18] for 200 training tasks, whose average length is greater than $2.5\,\mathrm{km}$. Each trajectory contains over 100 steps, forming a training dataset of 20K steps. Details of it can be found in Appendix C.4.

Statistics. For both training and evaluation, we synthesize 100 distinct worlds. With them, we create 200 easy tasks and 200 hard tasks. Each task has 2-4 instructions. On average, each task requires traveling 500 meters over 250 steps. To ensure generalization, 33% of the buildings in the testing set are exclusive and unseen during training.

Evaluation Metrics. We evaluate navigation performance in the easy setting through three key metrics: (1) **Success Rate (SR)** [1] measures the percentage of goal arrivals, (2) **Subtask SR** [46] tracks the ratio of completed subtasks, (3) **Distance Progress** [51] evaluates instruction-following by the relative reduction in distance to the goal. The hard evaluation setting, which introduces a comprehensive traffic system and pedestrians, requires three additional safety metrics: (1) **Static Collision** counts collision between the robot and immovable objects like buildings and trees, (2)

Table 3: Experimental results on the SIMWORLD-MMNAV benchmark (hard task set).

Models	SR%↑	Stat. Coll. ↓	Dyn. Coll.↓	Red Light Viol.↓	Subtask SR%↑	Distance Progress%↑
GPT-4o	2.08	1.92	10.37	3.02	34.38	24.83
Gemini 2.5 Flash	0	3.21	4.29	7.875	32.29	29.87
QwenVL 72B	0	5.0	11.73	2.86	23.86	21.97

Dynamic Collision counts collisions with moving elements, such as pedestrians and vehicles, (3) **Traffic Light Violation** records the number of times the robot fails to adhere to traffic signal regulations.

Table 4: Most common failure modes in SIMWORLD-MMNAV.

Subtask	Failure Mode	Frequency (%)
Moving to Intersection	Misestimate the distance to the intersection	53.33
	Fail to detect the intersection	28.33
	Misidentify the reference landmark	18.33
Turning	Misinterpret the turning pattern	42.86
	Misunderstand history status summary	42.86
	Fail to detect upfront buildings	14.29
Reaching Destination	Fail to match the landmark in a different perspective	60.00
	Stop too early to face the landmark	30.00
	Fail to align the landmark	10.00

Baselines. Inspired by the recent success of large vision-language models (VLMs) on navigation tasks [54], we evaluate multiple recent VLMs as backbones using ReAct [57], including GPT-4o, GPT-o3, GPT-o3-pro [36], Gemini 2.5 Flash [49], Qwen-VL 2.5 [3], Gemma 3 [50] and InternVL [9]. Additionally, we finetune QwenVL2.5-7B on SimWorld-20k. We also test a hybrid baseline, HybridGPT, where GPT-4o is used as a high-level decision maker and A* [18] is used as a low-level motion planner. For RL baselines, we train a multimodal policy model, VLA-RL, with DeBERTa-v3 [19] as language encoding and DINOv2 [37] as visual encoding, following VLN-CE [24]. We include more baseline implementation details in Appendix C.5.

4.2 Results

Zero-shot VLMs. As shown in Table 2, among zero-shot ReAct models, Gemini 2.5 Flash achieves the highest progress score. GPT-4o exhibits a mismatch between its distance progress and subtask completion rate, primarily due to its inability to detect termination conditions, often overshooting the goal and yielding zero distance progress. QwenVL2.5-72B ranks highest among open-source models, while QwenVL2.5-7B performs comparably to significantly larger models.

Finetuned Models. After fine-tuning, QwenVL2.5-7B shows substantial improvements across all metrics and is the only model to achieve a non-zero full task success rate. However, the absolute success rate remains low, partly because the training set is grounded on oracle action traces, which limits robustness. Incorporating reinforcement learning or corrective demonstrations could further enhance performance.

Reasoning Models. All zero-shot non-reasoning models score zero in SR, highlighting a fundamental capability gap. Case studies in Appendix E.1 suggest that these models often fail in task completion due to insufficient instruction grounding or inability to handle long-horizon dependencies. However, the results for reasoning models indicate that improved reasoning abilities boost performance. In our experiment, the reasoning models show improved depth estimation and destination alignment, which further demonstrates the importance of visual and spatial reasoning in our benchmark.

Other Baselines. The hybridGPT handles local turning more stably than zero-shot GPT-4o, but lacks fine-grained control, making overall performance more sensitive to GPT-4o's first-attempt accuracy. The RL baseline, VLA-RL, fails to outperform zero-shot LLMs, indicating the difficulty of our benchmark, where sparse reward signals and visually complex spatial reasoning pose challenges for conventional vision encoders.

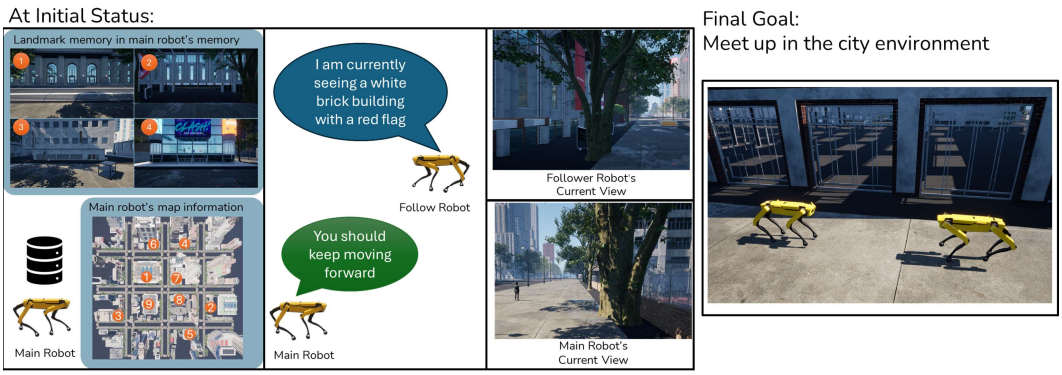

Figure 5: Illustration of a multi-robot search task.

Hard Setting. We further evaluated realistic obstacle avoidance and traffic rule obedience on models that performed relatively well on the easy setting. As Table 3 shows, Gemini 2.5 Flash performs better on avoiding pedestrians and vehicles; however, its red light violation count is higher, not due to failure to stop at red lights, but because the agent often freezes after detecting a red signal, even when already within the intersection. This indicates that there is still room for improvement in pragmatic reasoning and safety alignment under real-world conditions. Reasoning models, due to their inference latency, are not suitable for this setting of real-time traffic avoidance.

Ablation Test. We construct an ablation test using GPT-4o as the backbone, and we find that the explicit ReAct framework and segmentation provide the most significant marginal improvements among all the components. Details can be found in Table 11 in Appendix C.6.

Failure Analysis. We summarize typical failure modes of VLMs as follows, with specific qualitative examples detailed in Appendix E.1. (1) **Visual Grounding.** The grounding of VLMs hinder their performance. They fail to recognize the intersection, which is vital in our setting. The relatively low perspective of the robot dog add to the difficulties. (2) **Spatial and Embodied Reasoning.** VLMs do not yet have good 3D spatial reasoning capacity. They cannot robustly estiamte how close the robot is to a certain intersection or an obstacle. (3) **Pragmatic Thinking.** To emulate real-world textual navigation, our instructions use high-level phrases like "turn right at the intersection" without detailing the specific turning and crossing actions required. Most models fail to interpret such ambiguity correctly. (4) **Memory and Planning.** VLMs sometimes fail to adapt the information in memory to the current situation. The frequency for each common failure mode is shown in Table 4.

5 Multi-Robot Search Benchmark

In many real-world urban applications like search and rescue, multiple robots must collaborate. In large, unfamiliar environments, this requires localizing one another and meeting at a convenient location. Such multi-robot search is foundational for effective collaboration, but presents unique challenges: (i) each robot has partial, egocentric perception, (ii) the environment is dynamic and safety-critical, and (iii) coordination depends on grounded natural-language dialogue. As shown in Table 9 (Appendix B.2), prior benchmarks only address parts of this problem. We introduce SIMWORLD-MRS, a benchmark to fill this gap.

5.1 Setup

Task Definition. There are two robots, a main robot and a follower robot. The main robot has explored the city, so it has the memory of a map and images of landmarks (typically over 20 landmarks) in the city. However, the follower robot is new to the city and does not have such information. Neither robot knows the other's location. Their goal is to meet each other as soon as possible by physical navigation as well as verbal communication. A task is considered successful when at least one of the robots confirms that it can see the other robot in its egocentric view.

Table 5: Experiment results on SIMWORLD-MRS benchmark.

Models	Method	CSR%↑	Task Progress%↑
Proprietary Models			
GPT-4o	Oracle Planner	65.00	76.90
Gemini 2.5 Flash	Oracle Planner	54.55	75.84
GPT-4o	RoCo	33.33	22.93
Open-sourced Models			
QwenVL 2.5 72B	RoCo	11.11	35.94

Both robots have similar observation and action spaces as in the SimWorld-MMNav benchmark. Additionally, both robots can send natural language messages to each other. Each robot can send a confirmation signal whenever it believes that it has seen the other robot.

We provide more details of the benchmark in Appendix D.

Statistics. For evaluation in the multi-robot search benchmark, we construct 100 unique urban environments, each covering an area of $2.5\,\mathrm{km}^2$. In each environment, 20 distinguishable landmarks are selecteddistributed across all the city blocksas the main robot's memory of the city. On average, the initial distance between the two robots spawning locations is $576\,\mathrm{m}$, requiring 287 steps for an oracle planner to complete the task.

Evaluation Metrics. We assess multi-agent navigation performance through two principal metrics: (1) **Collaborative Success Rate (CSR)** [31] measures the percentage of tasks completed successfully before meeting; (2) **Task Progress** averages, across robots, the fraction of their shortest-path distance that is covered by the end of an episode.

Baselines. Following the collaboration paradigm of ROCO [33], we enable robots to discuss a joint plan for concrete rendezvous behavior using a VLM. The follower robot will first describe its location using language for the main robot to localize it. Afterwards, two robots will confirm a plan, where the main robot will describe paths to a meeting location for the follower robot to follow, while the main robot will plan its own path to reach the meeting location. In SIMWORLD-MRS, we first evaluated GPT-4o and Gemini 2.5 Flash with oracal planner and then picked the best performing VLM (GPT-4o) and paired it with RoCo. We include more implementation details in the Appendix D.4.

5.2 Results

Table 5 summarizes baseline results on our multi-robot search task, with specific qualitative examples detailed in Appendix E.3. GPT-4o with the oracle planner achieves the highest CSR (52%) and task progress (68.44%) by combining precise landmark localization with optimal A* planning, showing upper-bound performance with full map access. The RoCo policy lets the follower describe its view, which the map-aware robot localizes via VLM-based retrieval. This one-shot communication enables concurrent movement and more realistic coordination. However, converting rendezvous plans into language introduces ambiguity and execution noise, often causing path deviations. Without iterative replanning, grounding or control errors can't be corrected. Consequently, GPT-4o under RoCo achieves only 33.33% CSR and 22.93% task progresssignificantly lower than the oracle baseline.

6 Conclusion

We have created a novel embodied AI simulator, SimWorld-Robotics (SWR), for synthesizing photorealistic and dynamic urban environments. It can procedurally generate infinite photorealistic urban environments. Additionally, it can populate the environments with pedestrians, vehicles, and robots. By leveraging these features, we have built two new robot benchmarks. One focuses on multimodal robot navigation (SIMWORLD-MMNAV), and the other evaluates multi-robot collaboration in searching tasks (SIMWORLD-MRS). Our experimental results reveal significant limits in strong VLM-based baselines. Our evaluation also demonstrated the value of finetuning VLMs on large-scale training sets synthesized in our simulator.

Limitations and Future Work. Our current simulator focuses only on outdoor environments. The action space of the human agents, though more diverse than prior simulators, is still limited. In the future, we intend to scale up the action space by leveraging recent human body motion generation models. We also plan to incorporate indoor scenes into SWR.

References

[1] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir R. Zamir. On evaluation of embodied navigation agents, 2018.

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683, 2018.

[3] 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, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025.

[4] Shurjo Banerjee, Jesse Thomason, and Jason J. Corso. The robotslang benchmark: Dialog-guided robot localization and navigation, 2020.

[5] Abhijat Biswas, Allan Wang, Gustavo Silvera, Aaron Steinfeld, and Henny Admoni. Socnavbench: A grounded simulation testing framework for evaluating social navigation, 2021.

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[7] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias NieSSner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments, 2017.

[8] Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments, 2020.

[9] Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, Lixin Gu, Xuehui Wang, Qingyun Li, Yimin Ren, Zixuan Chen, Jiapeng Luo, Jiahao Wang, Tan Jiang, Bo Wang, Conghui He, Botian Shi, Xingcheng Zhang, Han Lv, Yi Wang, Wenqi Shao, Pei Chu, Zhongying Tu, Tong He, Zhiyong Wu, Huipeng Deng, Jiaye Ge, Kai Chen, Kaipeng Zhang, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling, 2025.

[10] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. Robothor: An open simulation-to-real embodied ai platform, 2020.

[11] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.

[12] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[13] Yue Fan, Winson Chen, Tongzhou Jiang, Chun Zhou, Yi Zhang, and Xin Eric Wang. Aerial vision-and-dialog navigation, 2023.

[14] Chuang Gan, Jeremy Schwartz, Seth Alter, Damian Mrowca, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, Megumi Sano, Kuno Kim, Elias Wang, Michael Lingelbach, Aidan Curtis, Kevin Feigelis, Daniel M. Bear, Dan Gutfreund, David Cox, Antonio Torralba, James J. DiCarlo, Joshua B. Tenenbaum, Josh H. McDermott, and Daniel L. K. Yamins. Threedworld: A platform for interactive multi-modal physical simulation, 2021.

[15] Chuang Gan, Siyuan Zhou, Jeremy Schwartz, Seth Alter, Abhishek Bhandwaldar, Dan Gutfreund, Daniel L. K. Yamins, James J DiCarlo, Josh McDermott, Antonio Torralba, and Joshua B. Tenenbaum. The threedworld transport challenge: A visually guided task-and-motion planning benchmark for physically realistic embodied ai, 2021.

[16] Chen Gao, Baining Zhao, Weichen Zhang, Jinzhu Mao, Jun Zhang, Zhiheng Zheng, Fanhang Man, Jianjie Fang, Zile Zhou, Jinqiang Cui, Xinlei Chen, and Yong Li. Embodiedcity: A benchmark platform for embodied agent in real-world city environment, 2024.

[17] Meera Hahn, Jacob Krantz, Dhruv Batra, Devi Parikh, James M. Rehg, Stefan Lee, and Peter Anderson. Where are you? localization from embodied dialog, 2021.

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

[19] Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing, 2023.

[20] Yidong Huang, Jacob Sansom, Ziqiao Ma, Felix Gervits, and Joyce Chai. Drivlme: Enhancing llm-based autonomous driving agents with embodied and social experiences, 2024.

[21] Harshit Jain and Priyal Babel. A comprehensive survey of pid and pure pursuit control algorithms for autonomous vehicle navigation, 2024.

[22] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts, 2023.

[23] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Aniruddha Kembhavi, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai, 2022.

[24] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments, 2020.

[25] Linh Kästner, Volodymyir Shcherbyna, Huajian Zeng, Tuan Anh Le, Maximilian Ho-Kyoung Schreff, Halid Osmaev, Nam Truong Tran, Diego Diaz, Jan Golebiowski, Harold Soh, and Jens Lambrecht. Arena 3.0: Advancing social navigation in collaborative and highly dynamic environments, 2024.

[26] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, C. Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks, 2021.

[27] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, Mona Anvari, Minjune Hwang, Manasi Sharma, Arman Aydin, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Silvio Savarese, Hyowon Gweon, Karen Liu, Jiajun Wu, and Li Fei-Fei. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 80–93. PMLR, 14–18 Dec 2023.

[28] Heng Li, Minghan Li, Zhi-Qi Cheng, Yifei Dong, Yuxuan Zhou, Jun-Yan He, Qi Dai, Teruko Mitamura, and Alexander G. Hauptmann. Human-aware vision-and-language navigation: Bridging simulation to reality with dynamic human interactions. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 119411–119442. Curran Associates, Inc., 2024.

[29] Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning, 2022.

[30] Shubo Liu, Hongsheng Zhang, Yuankai Qi, Peng Wang, Yaning Zhang, and Qi Wu. Aerialvln: Vision-and-language navigation for uavs, 2023.

[31] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *NeurIPS*, 2017.

[32] Ziqiao Ma, Ben VanDerPloeg, Cristian-Paul Bara, Huang Yidong, Eui-In Kim, Felix Gervits, Matthew Marge, and Joyce Chai. Dorothie: Spoken dialogue for handling unexpected situations in interactive autonomous driving agents, 2022.

[33] Zhao Mandi, Shreeya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large language models, 2023.

[34] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks, 2022.

[35] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations, 2021.

[36] OpenAI. Gpt-4o system card, 2024.

[37] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024.

[38] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs, 2018.

[39] Xavier Puig, Tianmin Shu, Shuang Li, Zilin Wang, Yuan-Hong Liao, Joshua B. Tenenbaum, Sanja Fidler, and Antonio Torralba. Watch-and-help: A challenge for social perception and human-ai collaboration, 2021.

[40] Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander William Clegg, Michal Hlavac, So Yeon Min, Vladimír Vondru, Theophile Gervet, Vincent-Pierre Berges, John M. Turner, Oleksandr Maksymets, Zsolt Kira, Mrinal Kalakrishnan, Jitendra Malik, Devendra Singh Chaplot, Unnat Jain, Dhruv Batra, Akshara Rai, and Roozbeh Mottaghi. Habitat 3.0: A co-habitat for humans, avatars and robots, 2023.

[41] Claudia Pérez-D'Arpino, Can Liu, Patrick Goebel, Roberto Martín-Martín, and Silvio Savarese. Robot navigation in constrained pedestrian environments using reinforcement learning, 2020.

[42] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. Reverie: Remote embodied visual referring expression in real indoor environments, 2020.

[43] Weichao Qiu, Fangwei Zhong, Yi Zhang, Siyuan Qiao, Zihao Xiao, Tae Soo Kim, and Yizhou Wang. Unrealcv: Virtual worlds for computer vision. In *Proceedings of the 25th ACM International Conference on Multimedia*, MM '17, page 12211224, New York, NY, USA, 2017. Association for Computing Machinery.

[44] Jiawei Ren, Yan Zhuang, Xiaokang Ye, Lingjun Mao, Xuhong He, Jianzhi Shen, Mrinaal Dogra, Yiming Liang, Ruixuan Zhang, Tianai Yue, Yiqing Yang, Eric Liu, Ryan Wu, Kevin Benavente, Rajiv Mandya Nagaraju, Muhammad Faayez, Xiyan Zhang, Dhruv Vivek Sharma, Xianrui Zhong, Ziqiao Ma, Tianmin Shu, Zhiting Hu, and Lianhui Qin. Simworld: An open-ended realistic simulator for autonomous agents in physical and social worlds, 2025.

[45] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles, 2017.

[46] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[47] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, C. Karen Liu, Silvio Savarese, Hyowon Gweon, Jiajun Wu, and Li Fei-Fei. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments, 2021.

[48] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat, 2022.

[49] Gemini Team. Gemini: A family of highly capable multimodal models, 2025.

[50] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Pluciska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil,

Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025.

[51] Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-dialog navigation, 2019.

[52] Nathan Tsoi, Alec Xiang, Peter Yu, Samuel S. Sohn, Greg Schwartz, Subashri Ramesh, Mohamed Hussein, Anjali W. Gupta, Mubbasir Kapadia, and Marynel Vázquez. Sean 2.0: Formalizing and generating social situations for robot navigation. *IEEE Robotics and Automation Letters*, pages 1–8, 2022.

[53] Hanqing Wang, Jiahe Chen, Wensi Huang, Qingwei Ben, Tai Wang, Boyu Mi, Tao Huang, Siheng Zhao, Yilun Chen, Sizhe Yang, Peizhou Cao, Wenye Yu, Zichao Ye, Jialun Li, Junfeng Long, Zirui Wang, Huiling Wang, Ying Zhao, Zhongying Tu, Yu Qiao, Dahua Lin, and Jiangmiao Pang. Grutopia: Dream general robots in a city at scale, 2024.

[54] 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. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution, 2024.

[55] Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

[56] Wayne Wu, Honglin He, Jack He, Yiran Wang, Chenda Duan, Zhizheng Liu, Quanyi Li, and Bolei Zhou. Metaurban: An embodied ai simulation platform for urban micromobility, 2024.

[57] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

[58] Yifan Yin, Zhengtao Han, Shivam Aarya, Jianxin Wang, Shuhang Xu, Jiawei Peng, Angtian Wang, Alan Yuille, and Tianmin Shu. Partinstruct: Part-level instruction following for fine-grained robot manipulation, 2025.

[59] Shengqiang Zhang, Philipp Wicke, Lütfi Kerem Şenel, Luis Figueredo, Abdeldjallil Naceri, Sami Haddadin, Barbara Plank, and Hinrich Schütze. Lohoravens: A long-horizon language-conditioned benchmark for robotic tabletop manipulation, 2023.

[60] Fangwei Zhong, Kui Wu, Churan Wang, Hao Chen, Hai Ci, Zhoujun Li, and Yizhou Wang. Unrealzoo: Enriching photo-realistic virtual worlds for embodied ai, 2025.

[61] Qinhong Zhou, Hongxin Zhang, Xiangye Lin, Zheyuan Zhang, Yutian Chen, Wenjun Liu, Zunzhe Zhang, Sunli Chen, Lixing Fang, Qiushi Lyu, Xinyu Sun, Jincheng Yang, Zeyuan Wang, Bao Chi Dang, Zhehuan Chen, Daksha Ladia, Jiageng Liu, and Chuang Gan. Virtual community: An open world for humans, robots, and society, 2025.

A SimWorld-Robotics Details

A.1 Assets

Building and Detail Assets We utilize a wide range of high-fidelity buildings and street-level details, all sourced from Unreal Engines official marketplace, Fab.com. All assets are used in compliance with their respective licenses and terms of use. The ground-truth attribution for building assets are made into word cloud, shown in Figure 6.

Figure 6: **Building Attribution in SWR** Each building asset in SWR is paired with a ground-truth description initially generated by language model and later verified and corrected by human in our team.

Embodied Agent Assets Quadruped robot models are sourced from CGTrader and integrated into SWR in compliance with the platform's licensing terms. Vehicles are adapted from the official Unreal Engine package `CitySampleVehicle`, with customized blueprint logic to support the specific requirements of our simulation. Human agents are created using Unreal Engines MetaHuman framework, providing diverse, realistic character models.

A.2 Full Comparison of SWR with Prior Simulators

Table 6: Comparison of simulation platforms across key features. The **Scenes** section includes environment Type (U: Urban, I: Indoor), support for Procedural Generation (✓: supported, ×: not supported), and level of Photorealism. The **Human** section summarizes scene interaction capabilities (Scene-Interact.) and the number of supported human Actions. **Embodied Agents** indicate whether robots (Rob.), humans (Hum.), and vehicles (Veh.) are supported. **AMC** denotes support for asynchronous multi-agent control.

Simulator	Scenes			Human		Embodied Agents			AMC
	Type	Procedural Generation	Photorealistic	Scene-Interact.	Actions	Rob.	Hum.	Veh.	
Matterport3d [7]	I	×	✓	×	×	×	×	×	×
Sean2.0 [52]	I	×	×	×	3	✓	×	×	×
Arena3.0 [25]	I	✓	×	×	8	✓	×	×	×
AI2THOR [23]	I	✓	×	×	×	✓	×	×	✓
TDW [14]	I	✓	✓	×	50	✓	×	×	×
SocNavBench [5]	I	×	×	×	2	✓	×	×	×
Habitat 3.0 [40]	I	✓	×	✓	4	✓	✓	×	✓
VirtualHome 2.0 [39]	I	✓	×	✓	25	×	✓	×	×
BEHAVIOR [27]	I	✓	✓	×	×	✓	×	×	×
CARLA [12]	U	×	✓	×	2	×	✓	✓	✓
AirSim [45]	U	×	✓	×	2	×	✓	✓	✓
MetaUrban [56]	U	✓	×	×	2	✓	×	×	✓
EmbodiedCity [16]	U	×	×	×	2	✓	×	×	×
Grutopia [53]	U	×	✓	×	×	✓	×	×	×
AerialVLN [30]	U	×	✓	×	×	✓	×	×	×
SWR (Ours)	U	✓	✓	✓	26	✓	✓	✓	✓

A.3 Asynchronous Multi-agent Control

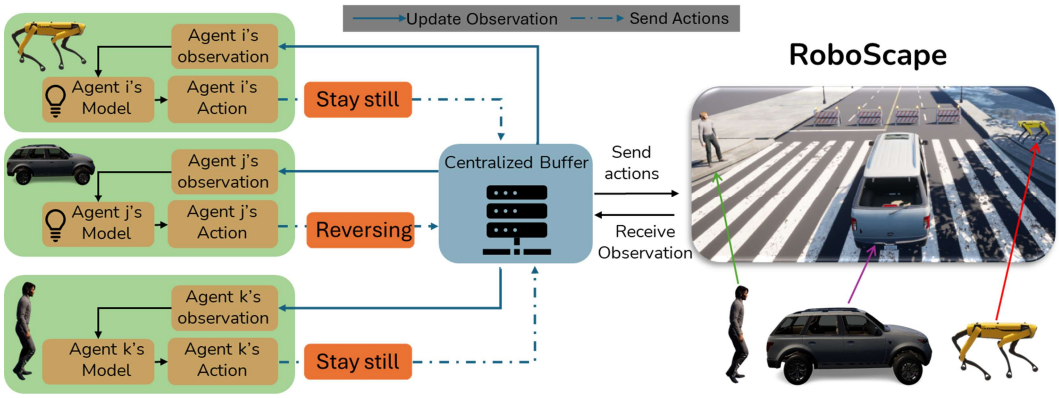

Figure 7: **Asynchronous multi-agent control.** To ensure each agent independently selects and executes actions based on its local observation and policy model, a centralized buffer serves as the interface between agents and the environment, receiving actions from all agents and updating their observations based on the resulting environmental changes.

The control pipeline is illustrated in Figure 7, where each agentregardless of embodiment (robot, vehicle, or pedestrian)independently perceives its local observation and selects an action using its policy model. These actions are asynchronously sent to a centralized buffer, which mediates the interaction between agents and the environment by updating each agents observation based on the environments response to all executed actions.

A.4 Observation Space

As illustrated in Figure 8a, the robot receives multimodal observations at each step, including RGB images, semantic segmentation maps, and depth maps, enabling a rich understanding of its surrounding environment.

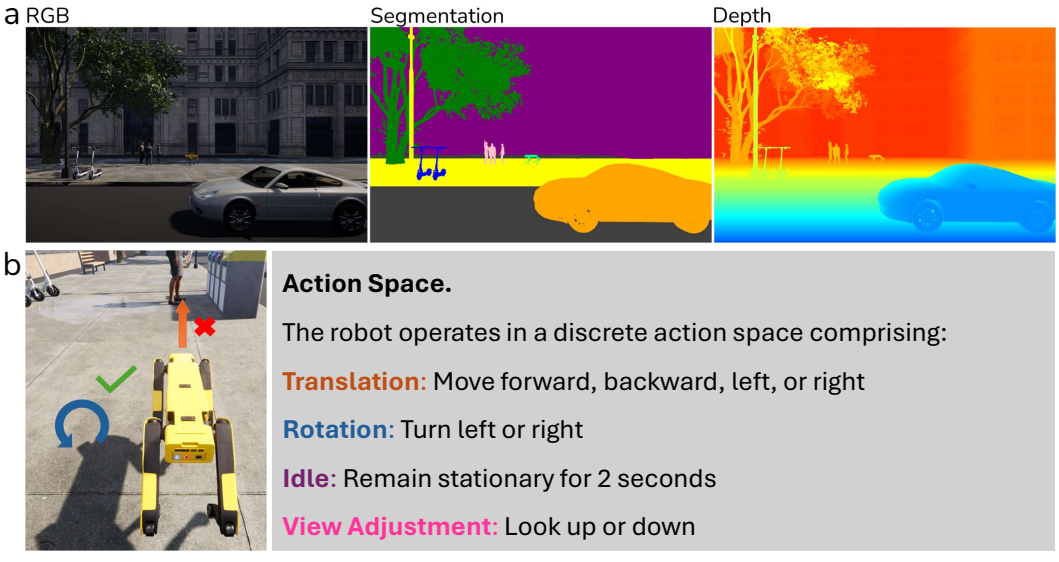

Figure 8: **Multimodal observation and discrete action space**.(a) The robot perceives the environment via RGB, segmentation, and depth modalities. (b) Its action space includes translation, rotation, idling, and view adjustment.

A.5 Robot Action Space

The robot executes actions within a discrete control space composed of directional movements, rotations, idle behavior, and vertical view adjustments, as illustrated in Figure 8b.

A.6 Procedural City Generation

The Procedural City Generation pipeline is a central component of the SWR simulator, designed to generate realistic and structured urban environments from minimal input specifications. This pipeline offers a highly modular and extensible architecture that supports the creation of diverse cityscapes, enabling a wide range of embodied AI tasks such as autonomous driving, pedestrian navigation, and multi-agent simulations.

As illustrated in Figure 3, the city generation process is organized into four sequential stages: road generation, building generation, street element generation, and traffic element generation. Each stage progressively adds layers of realism and complexity to the simulated environment.

Road Generation: The process begins with the creation of a road network, which serves as the backbone of the city. Roads are generated through an initiation phase and a tree-like growth process that balances depth and branching using a priority queue. Mechanisms such as road-end attachment and intersection checking ensure a coherent and plausible layout.

Building Generation: Once roads are established, buildings are procedurally placed along road segments. For each side of the road, candidate positions are sampled while checking for space availability and avoiding collisions. A greedy strategy is used to fill remaining gaps near road ends, maximizing spatial utilization and maintaining visual uniformity.

Street Element Generation: Smaller environmental elements such as trees, road cones, benches, and parked vehicles are generated around buildings and alongside roads. These details are categorized and placed based on contextual zoneseither surrounding buildings or within designated sidewalk areas. While collisions with other objects are not strictly enforced for performance reasons, the placement respects basic accessibility constraints.

Traffic Element Generation: The final stage involves populating the city with dynamic actors such as cars, pedestrians, and agents. These elements bring life to the simulation and interact with the static environment, enabling research in traffic flow, behavioral modeling, and agent-based navigation.

Internally, the pipeline utilizes dedicated managers for roads, buildings, and elements. Each manager maintains spatial data using both lists and quadtree structuresenabling efficient queries, spatial indexing, and collision checks. Procedural rules and constraints guide item generation to ensure the city is functionally consistent and visually appealing.

By separating generation into clearly defined stages and maintaining a rule-driven architecture, this pipeline provides a robust foundation for scalable and customizable city simulation within SWR.

A.7 Background Traffic System

In SWR, both pedestrians and vehicles are controlled using a rule-based traffic system, as illustrated in Figure 9. A waypoint system is constructed over the entire procedurally generated city, consisting of two types of waypoints: road waypoints and intersection waypoints. At each intersection, four intersection waypoints are sampled, one at each corner. For every road segment connecting two intersections, road waypoints are sampled at 17-meter intervals, linking the intersection waypoints at both ends. Prior to traffic simulation, we sample a sequence of connected waypoints for each pedestrian and vehicle. During simulation, each agent follows its assigned path. When an agent reaches an intersection waypoint, it checks the traffic light status: if the light is green and the remaining duration exceeds 15 seconds, the agent proceeds; otherwise, it waits until the signal turns green, ensuring safe and realistic traffic behavior.

Figure 9: **Traffic system in SWR.** Agents follow sampled waypoints and obey traffic lights at intersections, proceeding only if it's green signal.

A.8 Customization in SWR

The two benchmark tasks included in the paper are intended as case studies to demonstrate the functionalities and utilities of SWR. SWR was explicitly designed to support user-friendly customization of embodied agents, environments and tasks.

A.8.1 New Embodied Agents

To add a new robot type in SWR, one can leverage the existing Python API of SWR to conveniently customize action spaceseither continuous or discreteas well as observation spaces such as RGB, depth, or semantic segmentation images. The required additional work (1) obtaining a new robot asset, typically from the Unreal Engine Marketplace; (2) defining the robot's actions using Unreal's Blueprint system; (3) integrating these actions with our Python API to enable high-level control; and (4) attaching our camera components to support the desired observation space.

A.8.2 New Environments

Users can generate diverse and realistic urban layouts through our Python API by providing simple metadata inputs (e.g., number of streets, street length, object categories, and their spatial distributions, such as 10% trees, 5% tables and chairs). This allows researchers to create arbitrarily large and varied city environments.

A.8.3 New Tasks

Defining tasks in SWR is similar to standard Gym environments [6], as the Python API of SWR follows the same format for agent control (including pedestrians and vehicles). Users can spawn different types of agents, customize observation spaces (e.g., RGB, depth, or semantic segmentation) and action spaces (continuous or discrete actions), and program new goal definitions (e.g., language instructions, target images, or spatial objectives). Additionally, while the current pedestrians follow rule-based logic, users can override behaviors to simulate complex or rare cases. For instance, one can simulate a jaywalking pedestrian by scripting a few agents to cross during a red light, while placing a robot at the intersection to evaluate its reaction.

A.9 SWR Technical and Operational Details

Underlying simulation engine. Our simulation is built on UE5(Unreal Engine 5), leveraging its native Chaos physics pipeline: each object is assigned an appropriate collision mesh, and at each fixed simulation tick performs discretetime integration of Newtons equationsresolving forces, collisions, and joint constraints via its iterative solver.

Python API and runtime environment. On top of UE5, weve implemented a dedicated Python layer that communicates with the engine through an updated UnrealCVbased TCP server [43], where highlevel commands issued in Python are forwarded over TCP straight into the UE5 runtime. On top of this API, we provide a standard Gym interface so that researchers can plug in and benchmark any baseline with minimal effort. We will distribute a Windows executable and a Singularity container

for Ubuntu and macOS so that users can run SWR out of the box without local compilation; all binaries, container images, and the Gym wrapper will be opensourced upon publication.

Human-in-the-loop interface. We support a human-in-the-loop interface through which a human operator uses a mouse and keyboard to control the robot; all trajectories are recorded automatically as expert demonstrations for downstream training.

A.10 SWR Computational Requirements

To ensure wide accessibility, SWR supports adjustable rendering resolutions, allowing deployment across both high-end servers and modest laptops.

Recommended Setup

- **CPU:** Intel Core i7-12700H or AMD Ryzen 9 5900HS
- **GPU:** NVIDIA RTX 3070 or GPU with more than 6 GB
- **RAM:** 32 GB

Minimum Setup (60 FPS for SWR)

- **CPU:** Intel Core i7-11300H or AMD Ryzen 9 4800H
- **GPU:** NVIDIA RTX 2060 (notebook)
- **RAM:** 16 GB

A.11 Running Efficiency of SWR

We evaluated all baselines on a headless machine with an AMD EPYC 9534 CPU, L40S GPU and 64 GB RAM. We can run 2 instances in parallel with a fixed 60 fps. Here's the table when we stress-test the runtime performance with different settings.

Table 7: Runtime Performance Stress Test

Resolution	Rendering Quality	GPU Utils (%)	CPU Utils (%)	RAM (MB)
640×360	low	30.04	16.41	561.56
720×600	low	29.72	17.73	596.4
1280×720	low	26.5	18.56	734.81
640×360	high	30.03	16.59	564.04
720×600	high	29.61	17.47	589.3
1280×720	high	27.44	19.08	738.6

B Benchmark Comparison

B.1 Comparing SimWorld-MMNav with Prior Vision-Language Navigation Benchmarks

Table 8: Comparison of instruction following benchmarks for navigation.

Benchmark	Env		Agent		Route		
	Type	Num	Type	Act Space	Gen	Acts	Num
CVDN [51]	Indoor, Static	90	Camera	Graph-based	Manual	7	7,415
REVERIE[42]	Indoor, Static	90	Camera	Graph-based	Manual	5	7,000
TouchDown [8]	Outdoor, Static	1	Camera	Graph-based	Procedural	35	9,326
ANDH [13]	Outdoor, Static	1	Drone	6	Manual	7	6,269
AerialVLN [30]	Outdoor, Static	25	Drone	8	Manual	204	8,446
SimWorld-MMNav (Ours)	Outdoor, Dynamic	400	Robot	7	Procedural	100	400

Table 9: Comparison of multi-robot collaboration and navigation benchmarks. Dynamic indicates whether pedestrians, vehicles, or other moving obstacles are present.

Benchmark	Env		Agent		Comm.		Map	Objective
	Type	Scale	Num	Type	Modality	Max Len	Access	
RoCo [33]	Indoor, Static	Room	2	Mobile arm	None	–	None	Collab. pick&place
DOROTHIE [32]	Outdoor, Dynamic	Road	1 (ego car)	Vehicle	Spoken dlg.	$\sim$20 tok.	Full	Dialogue nav.
MetaUrban [56]	Outdoor, Dynamic	City	1+	E-scooter/robot	None	–	Full	Micromobility nav.
DriVLMe [20]	Outdoor, Dynamic	City	10+	Vehicle	None	–	Full	Coop. driving
RobotSlang [4]	Indoor, Static	Lab	23	Mobile base	Natural-lang.	80 char	Partial	Joint object search
Where Are You? [17]	Outdoor, Static	City	2 (tourist+guide)	Pedestrian	Natural-lang.	40 tok.	Split	Localization via dialog
SimWorld-MRS (Ours)	Outdoor, Dynamic	City-scale	2	Robot	Natural-language	128 char	Split	Rendezvous / meet-up

B.2 Comparing SimWorld-MRS with Prior Multi-Robot Collaboration and Navigation Benchmarks

C SIMWORLD-MMNAV

In this section, we present **SimWorld-MMNav**, a single-robot benchmark designed to evaluate multimodal navigation in large-scale urban environments. We begin by describing our procedural task generation pipeline, which enables the creation of diverse and realistic navigation tasks. We then introduce the **SWR-20k** training dataset, detailing how we generate fine-grained supervision signals for multimodal learning. Next, we elaborate on the baseline models used in our experiments, including their implementation pipelines and prompting strategies. Finally, we provide an in-depth analysis of notable failure cases observed during evaluation, such as incorrect 3D spatial reasoning and other representative errors.

C.1 Detailed Task Settings

Observation Space In addition to the multimodal instruction, the robot is equipped with a compass that indicates its current facing direction. To facilitate navigation, the robot also receives its egocentric RGB image, segmentation image, and depth image, which together provide a rich perception of the surrounding environment.

Action Space The robots action space consists of two categories: **movement actions** and **task-related actions**. In the movement category, the robot can choose from the following options: move forward, move backward, move left, move right, turn left (-90°), turn right ($+90^\circ$), or stay still. In the task-related category, the robot can select the action `evaluate`, indicating that it believes the current task has been completed. If the evaluation is correct, the robot receives the next subtasks language instruction and visual hint; otherwise, the episode terminates as a failure. Our simulator allows active perception like looking up by providing egocentric observation from different fields of view.

C.2 Metric Detail

We evaluate Single-Agent Instruction Following using three metrics: Success Rate (SR), Subtask Success Rate (SSR), and Distance Progress (DP).

Success Rate This metric measures the proportion of navigation tasks in which the agent successfully reaches the final destination. A trial is counted as successful only if the agent stops near the correct landmark building and is oriented towards it at the end of the trajectory.

Subtask Success Rate This metric evaluates the proportion of correctly completed subtasks within a navigation episode. Let N denote the total number of subtasks in a given instruction, and n_c the number of subtasks successfully completed by the agent. The Subtask Success Rate (SSR) for that episode is computed as:

$$\text{SSR} = \frac{n_c}{N} \tag{1}$$

We report the average SSR across all test episodes.

Distance Progress This metric quantifies how much closer the agent is to the goal at the end of the navigation compared to the beginning. Let d_0 be the initial distance to the goal and d_T the final distance. The Distance Progress (DP) is computed as:

$$\text{DP} = \max\left(\frac{d_0 - d_T}{d_0}, 0\right) \tag{2}$$

DP values are also averaged across all tasks to obtain the final score. All distances are computed using the Manhattan metric.

In the more challenging environment containing obstacles, pedestrians, and vehicles, we additionally measure three metrics to assess the robot's social navigation capabilities: the average number of static collisions (i.e., collisions with buildings or static obstacles), the average number of dynamic collisions (i.e., collisions with pedestrians or vehicles), and the number of actions that violate traffic light rules during navigation.

C.3 Procedural Task Generation

Algorithm 1: Task Generation for SimWorld-MMNav

Input: Navigation path $P = \{n_1, n_2, \ldots, n_k\}$, orientations $\{\theta_1, \theta_2, \ldots, \theta_k\}$
Output: Ordered list of subtasks $\mathcal{T} = \{(I_1, V_1), (I_2, V_2), \ldots\}$

Initialize subtask list $\mathcal{T} \leftarrow [\,]$;
$V_1 \leftarrow$ capture visual cue at n_1 with orientation θ_1;
$I_1 \leftarrow$ generate **Orientation Alignment** instruction using compass heading θ_1 and nearby landmark;
Add (I_1, V_1) to $\mathcal{T}$;
for $i \leftarrow 2$ **to** $k - 1$ **do**
    if $\theta_i \neq \theta_{i-1}$ **then**
        $V_{\text{move}} \leftarrow$ capture visual cue at n_{i-1} with orientation θ_{i-1};
        $I_{\text{move}} \leftarrow$ generate **Move Along the Road** instruction using a landmark near n_{i-1};
        Add $(I_{\text{move}}, V_{\text{move}})$ to $\mathcal{T}$;
        $V_{\text{turn}} \leftarrow$ capture visual cue at n_i with new orientation θ_i;
        $I_{\text{turn}} \leftarrow$ generate **Turn at Intersection** instruction based on the relative angle between θ_{i-1} and θ_i;
        Add $(I_{\text{turn}}, V_{\text{turn}})$ to $\mathcal{T}$;
$V_{\text{goal}} \leftarrow$ capture visual cue at n_k with orientation θ_k;
$I_{\text{goal}} \leftarrow$ generate **Reach Destination** instruction using goal landmark description;
Add $(I_{\text{goal}}, V_{\text{goal}})$ to $\mathcal{T}$;
return $\mathcal{T}$

To support diverse multimodal navigation scenarios, we construct the SIMWORLD-MMNAV benchmark under two difficulty levels: *easy* (without obstacles) and *hard* (with obstacles such as vehicles and pedestrians). We generate 200 city maps using our procedural city generation pipeline, with 100 maps allocated to each setting. For the map under hard setting, streetside obstacles are additionally generated. The task generation process is detailed in Algorithm 16.

On each generated map, we randomly sample a pair of points, denoted as P_{start} and P_{goal}. For each point, we locate the nearest landmark building and extract the front-door location, referred to as L_{start} and L_{goal} respectively. The robot is spawned at L_{start} and tasked with navigating to L_{goal}.

We then use the A* algorithm to compute an optimal path between L_{start} and L_{goal} over the city-wide waypoint graph.

This path is decomposed into a sequence of subtasks that reflect the robots expected behavior along the route. These subtasks are:

Orientation Alignment Each episode begins with an **orientation alignment** subtask. The robot is provided with a target compass direction (e.g., North, South, East, West) and a nearby landmark description to assist alignment, such as: Face north. You will see a modern building with light blue

glass on your left. An image is captured at the initial location with the correct orientation to serve as the visual cue for this step.

Move Along the Road As the robot follows the computed path, we iterate through consecutive waypoints. When the robot is expected to travel straight between two intersections, we define a **move along the road** subtask. Specifically, we identify a prominent landmark near the intersection where the robot is expected to turn, and use it to generate a descriptive instruction, such as: Move along the road and stop at the intersection when you see a large building with glass panels and a light brick base on the opposite side. A visual cue is captured at the intersection with the robots current orientation to assist visual grounding.

Turning at Intersections When a change in orientation is detected in the pathtypically at an intersectionwe insert a **turn at the intersection** subtask immediately following the previous move along the road step. Based on the relative orientation of the next waypoint, we determine whether the robot should turn left or right. A new visual cue is captured at the next node with the updated heading, and a corresponding instruction is generated, such as: Turn left at the intersection and you should see this view.

Repeat Navigation Steps The move along the road and turn at the intersection subtasks may repeat multiple times until the robot reaches the final waypoint along the planned path.

Reach Destination At the final step, we define a **find destination** subtask. A detailed description of the goal building and its spatial relationship to the robots position is used to generate the final language instruction. A visual cue is captured with the robot facing the destination building to aid recognition.

This process completes one full navigation episode. For each map, we generate two episodes by sampling different pairs of start and goal locations, resulting in a total of 400 navigation tasks200 under the easy setting and 200 under the hard setting.

C.4 Details of SWR-20k

The SWR-20k is generated as training dataset using a distinct set of maps, referred to as training maps, which differ from those used for evaluation. The primary difference lies in the building distribution: training maps include only 66% of the building types that appear in the testing maps. The remaining 34% of building types are held out exclusively for testing, ensuring a clear separation between seen and unseen environments and promoting better generalization.

We generated 100 training maps using only the selected 66% of building assets. On each map, two tasks are sampled, and the oracle trajectories are generated by A* algorithm, resulting 200 orcale trajectories. Each trajectory contains over 100 steps, forming a training corpus of 20K steps.

At each step during training, the robot receives a synthesized observation and a corresponding sub-instruction, and is required to predict the correct next action. To enhance the robot's ability to reason about the goal and environment, we supervise not only the action prediction but also several intermediate reasoning targets. Specifically, the robot is trained to jointly predict: (1) the distance between the current observation and the target visual hint; (2) the orientation of the visual hint; and (3) the potential sequence of actions from the current location to the final goal. All of these supervisory signals are paired with ground-truth annotations and serve as multi-task learning objectives, guiding the robot toward a deeper understanding of spatial context and task intent.

C.5 Baseline Detail

Zero-shot Pipeline To reduce the mapping complexity and prevent hallucinations, we decompose step-wise action prediction in multimodal instruction-following into two modules: a context-aware perception module and a ReAct-based [57] action module as detailed in Figure 10. Throughout this process, the model autonomously maintains a working memory.

The Context-Aware Perception module receives the current observation, a visual hint (i.e., the expected view after subtask completion), the language instruction, and the current memory state. We prompt a vision-language model with the instruction and memory to generate a task-relevant

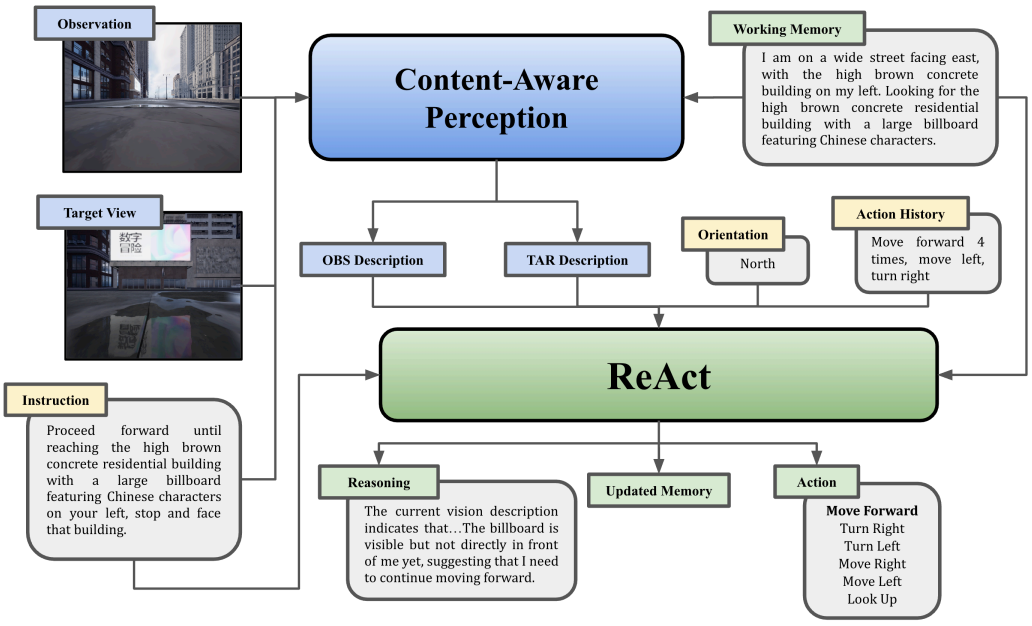

Figure 10: Zero-shot single agent multimodal instruction following pipeline

description. This description incorporates information from both the current observation and the target view, and is explicitly encouraged to compare the two images in order to assess whether the subtask has been completed.

The ReAct module operates purely in the text domain. It receives as input the perception-generated description, the original instruction, the memory, the robots current orientation, and a textualized history of past actions. ReAct is expected to first perform reasoning, then update the memory module accordingly, and finally select the next action for the robot.

We utilize segmentation images solely as auxiliary visual signals. Given the varying levels of training and understanding of depth images across different models, and considering that interpreting depth information is not the primary focus of our task, we exclude depth images from all inputs to eliminate potential confounding factors. In our experiments, we allow the model to generate a sequence of actions at each step. This design improves execution efficiency during long straight-path phases and serves as a test of the model's short-horizon planning capabilities.

Content-Aware Perception System Prompt in Zero-Shot Single Agent Instruction Following

```
You are a perception module of a navigation robot in a 3D environment. The ultimate
    goal is to place yourself right in front of a particular building.

You will be given:
- A history summary.
- Your vision description at the last step.
- A single egocentric image.
- The exact expected view you will see once you complete the current subtask.
- A segmentation image. The segmentation of the entire view. Green = trees, purple =
    buildings, yellow = sidewalks/crosswalks, black = driveways.
- The subtask you are working on.
- The current cardinal direction.

Instructions:
- First, describe the observation in detail, focusing on the color, texture,
    attachment, etc. of buildings.
- Focus on potential landmarks (if needed), obstacles, or intersections. Reason about
    useful details for actions.
```

```
- Include details like whether the agent is walking straight on a sidewalk and where
    the driveway is.
- If the upper parts of the entire observation are filled with buildings, the agent is
     not walking straight along a sidewalk.
- Judge the distance by the vertical position of the object to infer how many forward
    steps can be taken or how close intersections are.
- Finally describe the expected view, also attentive to the buildings. Match the
    expected view with current view to determine if the subtask is completed.

When aligning with the last landmarks:
- When the subtask asks you to face a building, first ensure proximity to the landmark
    , not just visibility in the strip. Look for close-up details to ensure you have
    reached the right position.
- You don't need to mention the orientation, because it will be given.

Intersection:
- When approaching intersections, actively look for the expected landmark.
- Keep in mind: the landmark might not be visible at the intersection due to limited
    field of view.
- If you cannot see the landmark, only use the expected view as a reference.
- Turning once alone does not guarantee completion of the turning subtask. Always
    verify against the expected view.

Output Format:
You must return a JSON object like:
{"Reason": "The detailed description and your reasoning about what detail is useful.",
 "Description": "A useful summary of the observation"}
```

ReAct System Prompt in Zero-Shot Single Agent Instruction Following

```
You are a navigation robot in a 3D environment. The ultimate goal is to place yourself
     next to a particular building.

You will be given:
- A list of actions that have already been taken (action history).
- The action sequence you took last step.
- A description of last step's observation. You can use it to compare with the current
     observation.
- A summary of the history and current status. You can use and update the summary as a
     hint for future planning.
- The current subtask you are working on.
- Current orientation of the robot.
- A detailed description of the agent's current perception. Then a description of the
    exact expected view you will see once you complete the current subtask.

Valid actions:
-1: Subtask_completed - If you believe the current subtask is completed, the action
    sequence should be [-1].
 0: Move_forward - Move 5 meters forward in the direction the robot is facing.
 1: Rotate_left - Rotate 90ř to the left.
 2: Rotate_right - Rotate 90ř to the right.
 3: Move_left - Move 5 meters left, without rotating.
 4: Move_right - Move 5 meters right, without rotating.

Instructions:
- First, analyze the current visual observation, the instruction, current situation,
    and history, and reason about how to update the history summary and the next
    action.
- Next update and resummarize the history and current status. If you have aligned to
    the current landmark, update the landmark to the next one.
- Finally, decide the next action steps based on the previous analysis.
```

```
Alignment:
- You have the cardinal direction to help you align at the beginning of the task.

The last Alignment:
- When handling the "face the building" subtask, you must be close enough and turn to
    face the building to complete the subtask.
- If you cannot see the building after rotating, it means you are not close enough.

Intersection:
- While reaching intersections, actively look for the expected landmark. Once it's
    spotted, update your history so the next intersection is the one.
- Keep in mind: the landmark might not be visible at the intersection due to limited
    field of viewuse the expected view as your reference.
- Turning once alone does not guarantee completion of the turning subtask. Always
    verify against the expected view.

Important Rules:
- Make sure you are oriented along the sidewalk when following "move forward" commands
    .
- You can plan by outputting variable-length action sequences. For example, [0, 0, 0,
    0, 0] if the path is clear.
- Shorter sequences if obstacles/intersections ahead.
- If you only see sky and road on one side, it means you are at the map boundary.
    Rotate to face buildings.
- The subtask is done only when the current view matches the expected view.

Remember:
- If you believe the subtask is completed, output [-1]. Remind yourself in the history
     that you are starting to do the next subtask.
- You must always output at least one action. If lost, try rotating.

Output Format:
You must ALWAYS return a JSON exactly like:
{"Reason": "Your reason", "Summary": "New summary of history and current status", "
    Actions": [list of integers]}
```

Table 10: Key hyperparameters used during finetuning

Hyperparameter	**Value**
Training epochs	2
Batch size	32 (4×8)
Optimizer	AdamW
Learning rate	2e-4
Weight decay	0.01
LR Scheduler	Constant (with warmup)
Warmup ratio	0.03
Gradient Clipping	0.3
LoRA rank	8
LoRA alpha	16
LoRA dropout	0.05

Finetuning During finetuning, the model receives as input the current observation, the target image representing the expected view upon subtask completion, the instruction, the current orientation, and the ground-truth action history. Based on these inputs, the model is trained to predict the estimated distance, the anticipated final orientation, and the remaining action sequence as a form of CoT planning, followed by the prediction of the next action. We finetune the Qwen2.5-VL-7B-Instruct model using LoRA [11] applied to both the language model head and the merging projection layer. The loss is computed solely on the tokens generated by the decoder. We used four A100 GPUs with 80GB VRAM each for finetuning. The hyperparameters can be found in Table 10

Hybrid Method The hybrid system uses GPT-4o as a high-level decision maker determining whether to continue straight or turn at intersections. It then uses A* as a low-level path planner to generate and execute movement commands.

RL Method The policy is first pretrained through behavioral cloning on expert demonstrations and then finetuned with PPO. Training is conducted on two NVIDIA L40S GPUs, each running two parallel instances of SWR.

System Prompts in Finetuned Single Agent Instruction Following

```
You are a navigation robot in a 3D environment.

You will be given:
- An current egocentric image.
- An expected view image (what you should see when the subtask is completed).
- A textual description of the subtask.
- Your current cardinal direction (e.g., "North").
- A history of previously taken actions.

You must:
1. Determine whether the current subtask is already completed by comparing the current
     and expected views.
2. Deduce the expected orientation when the subtask is completed.
3. Deduce the distance from the current position to the expected position.
4. Plan the remaining actions to complete the subtask based on current and expected
    views.
5. If the subtask is not completed, output the action you will take in this step. If
    the subtask is completed, output -1.

Valid actions:
-1: Subtask_completed
0: Move_forward - Move 5 meters forward in the direction the robot is facing.
1: Rotate_left - Rotate 90ř to the left.
2: Rotate_right - Rotate 90ř to the right.

Output Format:
Only return a JSON object like:
{"Expected_Orientation": "The Orientation", "Remaining_Distance": "The Distance", "
    Remaining_Actions": "Textual Plan of the Actions", "Next_Action": integer}
```

C.6 More Quantitative Results

Ablation Here we provide the ablation study of our ReAct baseline with GPT-4o as base model, tested on a 50-task subset of SimWorld-MMNav, as Table 11 shows.

Table 11: Ablation study with key components.

Configuration	Explicit Reason	Separate Perceive/Act	Depth	Segment	Strip	Subtask Success Rate (%)
Our setting	✓	✓	–	✓	–	34.38
Merged call	✓	–	–	✓	–	33.54
w/ depth	✓	✓	✓	✓	–	33.39
w/o explicit ReAct	–	✓	–	✓	–	32.21
w/o segmentation	✓	✓	–	–	–	31.90
w/ stripping	✓	✓	–	✓	✓	31.22

Our setting requires the model to explicitly reason before acting through multi-turn interaction: it first describes the observation, then reasons and decides the action. The input includes the observation and its segmentation mask, without depth or stripped visual parts.

The perception-action framework simplifies the mapping process and reduces hallucinations. While the model possesses a certain implicit depth estimation capacity, directly adding depth images

yields marginal gain and can even introduce noise if the colormap is misaligned. The explicit ReAct framework notably stabilizes the reasoning process and mitigates hallucinations in complex intersections. Although GPT-4o itself lacks sufficient training on intersection-heavy scenes, the ground-truth segmentation image helps alleviate this limitation. Finally, stripping the observation into vertical chunks increases visual matching difficulty, leading to degraded performance in our simplified setting.

Statistical Significance We also provide the 95% CI of our main results, as Table 12 shows. For success rate and subtask success rate, we use binomial proportion confidence interval [55] and display both lower and upper bound in the table.

Table 12: Experimental results on the SimWorld-MMNav benchmark (easy task set) with confidence intervals

Models	SR%↑	Subtask SR% ↑	Distance Progress% ↑
Proprietary Models			
GPT-4o	0 [0, 3.85]	33.07 [24.71, 43.24]	15.60 (±6.97)
Gemini 2.5 Flash	0 [0, 4.32]	37.06 [28.09, 48.27]	31.29 (±8.11)
Open-sourced Models			
QwenVL 2.5 7B	0 [0, 4.14]	16.86 [10.49, 25.96]	7.82 (±2.45)
QwenVL 2.5 72B	0 [0, 3.89]	23.80 [17.60, 34.84]	17.50 (±6.11)
Gemma 3 27B	0 [0, 3.85]	15.36 [9.70, 24.19]	6.83 (±5.44)
InternVL 3 78B	0 [0, 4.28]	18.31 [11.79, 28.11]	9.34 (±4.04)
Fine-tuned Models			
QwenVL2.5 7B$_{ft}$	4.0 [1.08, 13.22]	52.45 [39.52, 65.95]	53.63 (±10.88)

D SIMWORLD-MRS

To address the limitations of existing benchmarks in multi-robot search, we propose **SimWorld-MRS**, a new benchmark designed to evaluate collaboration, localization, and communication among multiple robots in large-scale, photo-realistic urban environments. SimWorld-MRS simulates realistic challenges such as partial observability, dynamic environments, and natural language coordination. In the following subsections, we detail the procedural task generation, baseline implementations, prompting strategies, and provide case studies to illustrate key behaviors like localization and communication.

D.1 Detailed Task Settings

Observation Space The observation space for the follower robot closely mirrors that of the single-agent setting, with the key difference being that its instructions are exclusively derived from inter-agent communication. In contrast, the guide robot has access to the complete map and landmark information of the simulated city, enabling it to generate an oracle path plan and identify specific landmarks for rendezvous.

Action Space Compared to the single-agent setting, the follower robot is additionally capable of initiating communication, while the guide robot is responsible for route planning and conveying instructions to the follower. Both robots are also able to pause and check whether the other is within their field of view, facilitating coordinated rendezvous.

D.2 Metric Detail

We report two metrics for evaluating performance on the SimWorld-MRS task: Collaborative Success Rate (CSR) and Task Progress (TP).

Collaborative Success Rate This metric estimates the probability that the two robots successfully meet up across different maps. A meet-up is considered successful if at least one robot executes the `check_task_complete` action and detects the other robot dog in its observation via ground-truth segmentation.

Task Progress This metric quantifies the relative reduction in distance between the two robots by the end of the task. Let D_0 denote the initial distance between the two agents, and D_T the distance when the task terminates. The Task Progress (TP) is defined as:

$$\text{TP} = \max\left(\frac{D_0 - D_T}{D_0}, 0\right) \tag{3}$$

D.3 Procedural Task Generation

To construct the SimWorld-MRS benchmark, we procedurally generate 100 unique city maps, each covering a large-scale urban environment. For each map, we instantiate one multi-robot search task, resulting in a total of 100 evaluation tasks. The task generation process follows Algorithm 11.

Algorithm 2: Task Generation for SimWorld-MRS

Input: City map with m streets; sample n landmark buildings per street
Output: main robot's memory $\mathcal{M}$, main robot spawn s_{main}, following robot spawn s_{follow}

```
// Phase 1: Build Main Robots Memory
```
Initialize memory set $\mathcal{M} \leftarrow [\,]$;
for $i \leftarrow 1$ **to** m **do**
  Sample n landmark buildings on street i: $\{L_{i1}, L_{i2}, \ldots, L_{in}\}$;
  for $j \leftarrow 1$ **to** n **do**
    $x_{ij} \leftarrow$ get front-door location of L_{ij};
    $o_{ij} \leftarrow$ compute orientation facing toward L_{ij};
    $I_{ij} \leftarrow$ capture visual cue at x_{ij} with orientation o_{ij};
    Add (I_{ij}, x_{ij}) to memory set $\mathcal{M}$;

```
// Phase 2: Sample Initial Robot Positions
```
Obtain a set of valid robot spawn points S;
Randomly sample two distinct spawn points from S: $s_{\text{main}}, s_{\text{follow}}$;
return $(\mathcal{M}, s_{\text{main}}, s_{\text{follow}})$

Specifically, we first sample n landmark buildings along each of the m streets in the city and collect their front-door images with aligned orientations. These image-location pairs serve as the main robot's memory of the city. Next, we sample two distinct and valid spawn locations as the starting positions for the main robot and the following robot. This ensures spatial diversity and supports realistic localization and communication challenges.

To evaluate the accuracy of the final meetup, we utilize the observations captured by both robots at the end of the multi-agent navigation process. Given that our system includes access to ground-truth segmentation, we determine whether the other robot appears within the field of view by inspecting the segmentation output. The presence of the counterpart robot in the observation is treated as evidence of a successful meetup.

D.4 Baseline Detail

Baseline 1 - Oracle Planner In the multi-robot oracle setting, our baseline communication pipeline consists of three key components: the follower robot's description of the current building, the main robot's retrieval of the building, and the oracle path planning to reach the follower.

The description module takes the current egocentric observation as input and feeds it into a VLM using a templated prompt. This prompt is specifically designed to guide the model towards generating goal-oriented and informative descriptions of the current scene, ensuring alignment with task requirements.

The memory retrieval module receives the generated textual description and a pre-collected landmark image dataset as input. It then uses the VLM to identify the landmark image that best matches the description. To achieve this, we implement a tournament-style elimination process: images are compared in pairs based on their semantic and visual alignment with the description, and in each round, the less relevant image is discarded. This process continues iteratively until a single most relevant image remains. To mitigate hallucinations and enhance retrieval stability, we employ

a system prompt that encourages precise and comparative reasoning. The output of this module is the position of the landmark image judged by the VLM to be the most semantically and visually consistent with the description.

The oracle path planning module receives as input the current location of the main robot and the estimated location of the follower robot, as determined by the memory retrieval module. Using full access to the global city map, the module computes the shortest collision-free path between the two locations via the A* algorithm. The resulting path is converted into a sequence of discrete navigation actions, such as moving forward or turning at intersections. These actions serve as an oracle reference and can be directly issued to the follower robot or translated into natural language instructions for communication purposes.

During the main robots execution of the oracle path, the follower robot actively rotates at each step and uses the VLM to detect whether the other robot dog appears in its field of view.

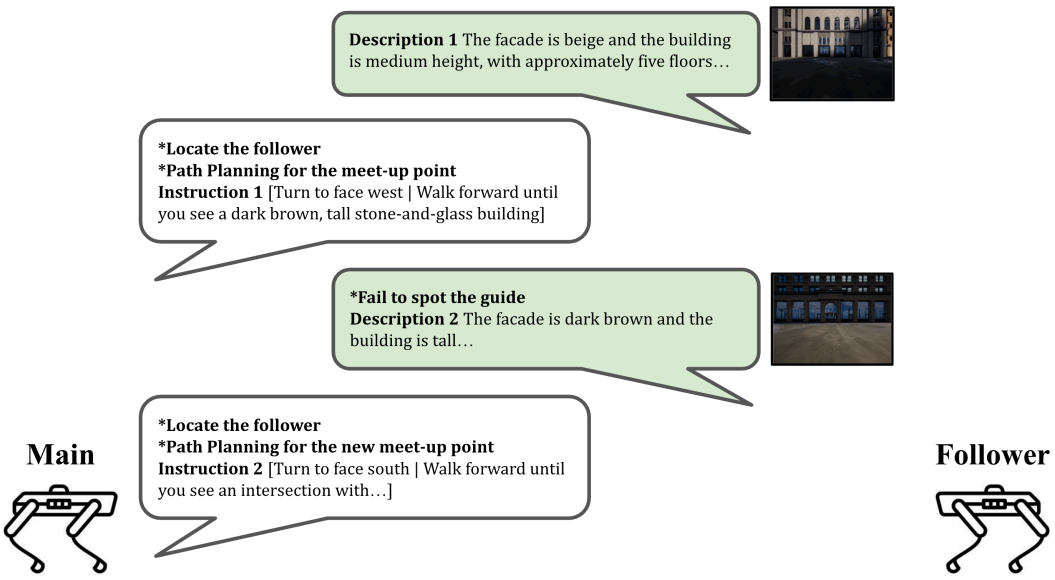

Figure 11: Example communication for ROCO baseline

Baseline 2 - ROCO The ROCO-based [33] setting extends the oracle setup by introducing collaborative planning and communication between two robots. After the two agents communicate and confirm the followers current location in terms of a landmark building, the main robot uses access to the ground-truth map to identify the landmark closest to the midpoint between both agents. It then performs path planning to compute the optimal trajectories for both agents to converge at this intermediate landmark. The main robot transforms the computed path for the follower into natural language instructions, which are communicated to the follower robot. The follower then performs multimodal instruction following to navigate toward the meeting point and rendezvous with the main robot. An example for communication in the ROCO baseline can be seen in Figure 11.

The description, memory retrieval, and path planning modules mirror those in the oracle baseline. The instruction generation module translates the planned path into natural language commands using a predefined set of sentence templates. These instructions are compact and landmark-aware, designed to guide the follower robot step-by-step without access to the full map. Each instruction encapsulates one or more discrete navigation actions, such as moving forward until reaching a visible landmark or turning at an intersection.

The communication module governs when to initiate a new dialogue round. After executing a received instruction, the follower robot monitors its state and triggers a new communication cycle under the condition: if it believes it has completed the instruction but, after rotating to search, does not observe the main robot. Upon reactivation, the follower generates a new scene description, enabling the main robot to re-localize its position and update the rendezvous plan accordingly.

The instruction-following module remains largely consistent with the single-agent setting, with the primary difference being that the visual hint is no longer provided.

Prompting for Multi-agent Baselines We adopt a modular prompting strategy that aligns with our multi-agent architecture. Specifically, we design two system promptsone for generating egocentric scene descriptions (used by the follower robot), and another for conducting comparative landmark retrieval (used by the main robot). These prompts are implemented via a VLM with image and text modality support.

The follower robot observes its local environment and produces a textual description aimed at helping the main robot identify its current location. The prompt guides the VLM to emphasize salient and matchable visual features such as color, height, materials, signage, and local context.

Follower Description Prompt

```
You are an expert building-description assistant.

In 200 words, describe the building so another person could match photos of it.

**Start the first sentence with the facade's MAIN COLOR and HEIGHT.**

Cover these attributes as comma-separated phrases:
- main color  dominant facade color
- height  low / medium / tall (number of floors)
- primary materials  e.g., brick, concrete, glass, steel
- window grid / pattern  shape and arrangement of windows
- ground-floor layout  doors, arches, glazing style
- signage text  exact words visible; say "no signage" if none
- sidewalk objects  lamp-posts, trees, benches, etc.
- distinctive features  murals, balconies, arches, etc.
- neighbor  immediate surrounding context (e.g., adjacent buildings or empty lots)

Keep it factual and avoid subjective opinions. Return a single descriptive paragraph
    in natural language.
```

After receiving a description from the follower robot, the main robot attempts to match it against its landmark memory using a VLM-based comparative prompt. The model is required to select the image (A or B) that better matches the textual input, focusing on distinctive visual features.

Main Robot Retrieval Prompt

```
TEXT = Natural-language description of the target building.

Two FULL-FACADE candidate photos are shown: A (first) and B (second).

Decide which matches TEXT better.

Guidelines:

- Give strong weight to facade color, signs, materials, and distinctive features.
- If the facade color clearly mismatches, that candidate must lose.
- You must compare both options and select the better one.

Reply only with A or B.
```

E Qualitative Examples

We provide qualitative examples to illustrate the common failure modes of VLMs.

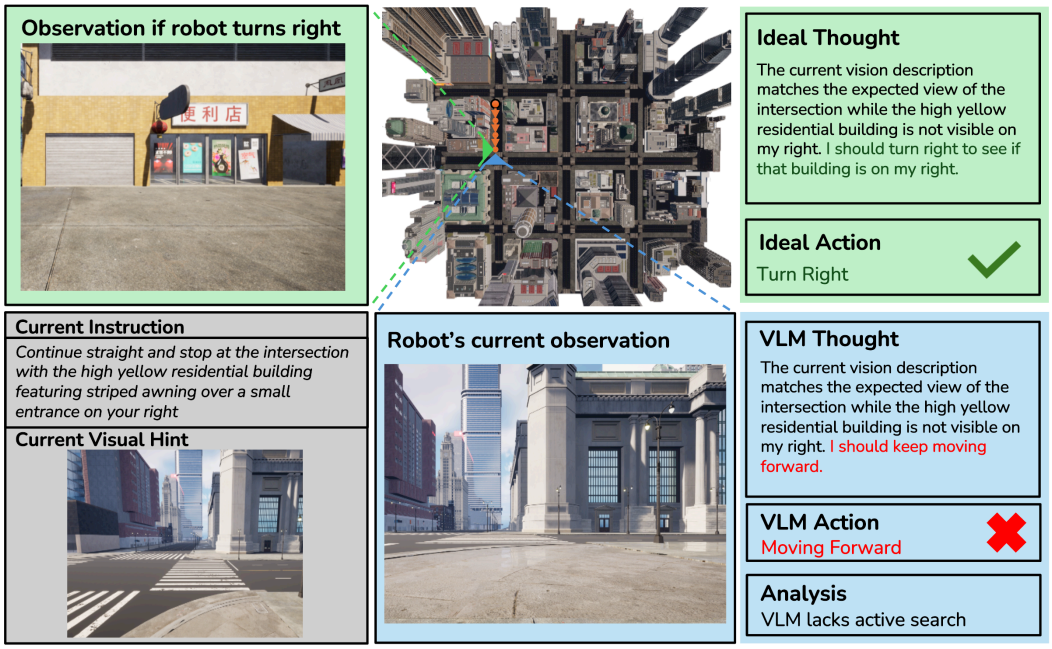

Figure 12: Qualitative result - lack of active perception

E.1 SimWorld-MMNav

Active Perception The VLM lacks initiative in active perception. As Figure 12 shows, at intersections, the robots field of view is often limited, and the landmark referenced in the instruction may not be visible from the current perspective. As a result, the robot does not recognize the location as the intended intersection, even when the visual hint has already been matched. Ideally, the robot should rotate to check its surroundings for the missing landmark. However, the VLM tends to continue moving forward, waiting for the landmark to appear directly in front of the robot rather than actively seeking it through lateral exploration.

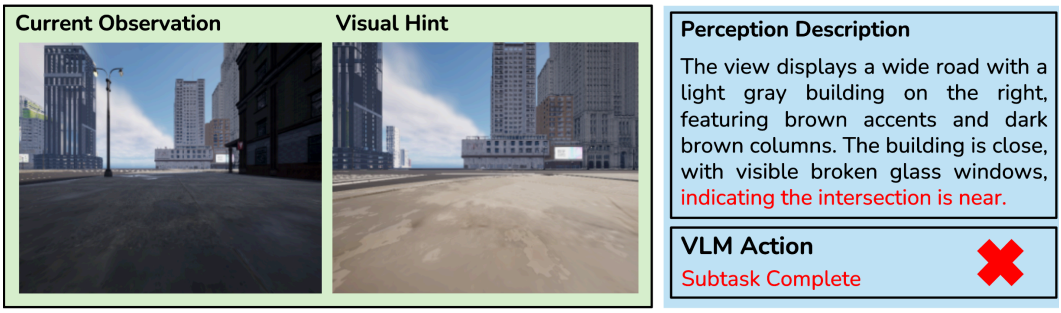

Figure 13: Qualitative result - lack of distance grounding

Spatial Reasoning The VLM exhibits limitations in reasoning about spatial relationships, particularly in estimating distance, maintaining spatial continuity, and interpreting alternate perspectives. In one failure case, the current observation partially resembles the visual hint, leading the VLM to prematurely assume arrival at the intersection, despite the robot still being far from the crosswalk, as Figure 13 shows. This example also indicates that the recognition of intersection lacks robustness and relies heavily on referential landmark buildings.

In another instance, as Figure 14 shows, during an orientation alignment task, the robot is initially facing a landmark. After turning right, a characteristic part of the landmark disappears from view.

Figure 14: Qualitative result - lack of embodied reasoning

Given a working memory, an embodied agent would robustly infer that it has aligned accordingly. However, the VLM fails to make this inference, indicating a lack of embodied spatial understanding.

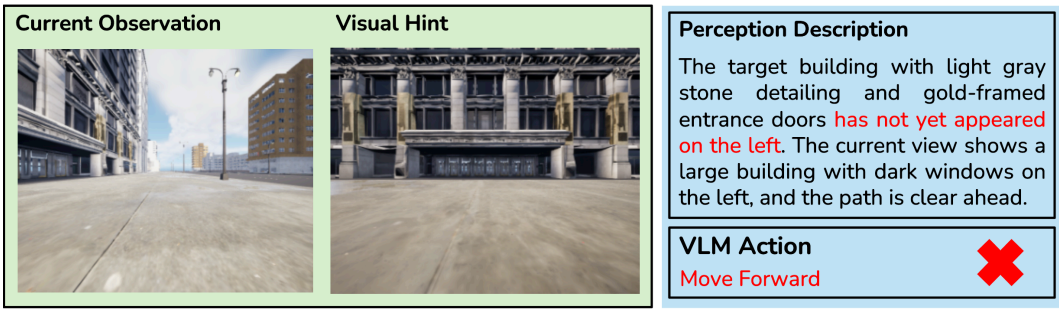

Figure 15: Qualitative result - lack of perspective-adaptive matching

These limitations also manifest when matching buildings from different perspectives. The target building is provided as a frontal image, but during navigation, only a side view may be visible. The perception module often fails to associate the side and frontal views and provides insufficient information, causing the robot to overlook the destination, as Figure 15 shows.

Even when a correct match is made, the VLM may still fail to reorient. For example, given the thought, "Currently, I am on a wide street facing east with the high brown concrete building and billboard on my left. The billboard is not directly in front of me yet, indicating that I need to continue moving forward until it is," the model chooses to proceed rather than rotate, failing to reason from an embodied perspective.

Pragmatic Reasoning The VLM also struggles with interpreting pragmatic intent in natural instructions. When given the instruction "turn right at the intersection," it often treats the subtask as complete after a single turning action. In real-world scenarios, such a turn typically involves multiple steps, like moving forward, turning, and possibly crossing the street. The VLMs overly literal interpretation leads to partial execution and deviation from the intended path.

E.2 A Success Case of Finetuned Baseline

As Figure 16 shows, finetuning facilitates Qwen2.5-VL-7B-Instruct's spatial reasoning capabilities. When the agent is relatively close to the subtask target, the model is able to accurately infer the remaining distance and the corresponding sequence of actions, while also correctly predicting the final orientation. This enables a strong foundation for chain-of-thought (CoT) planning and improves the reliability of next-step action prediction. Furthermore, in turning scenarios, the finetuned model demonstrates, for the first time, the ability to reason about intermediate stepssuch as the need to cross the streetin order to align with the target image. This allows the model to successfully complete the illustrated navigation task.

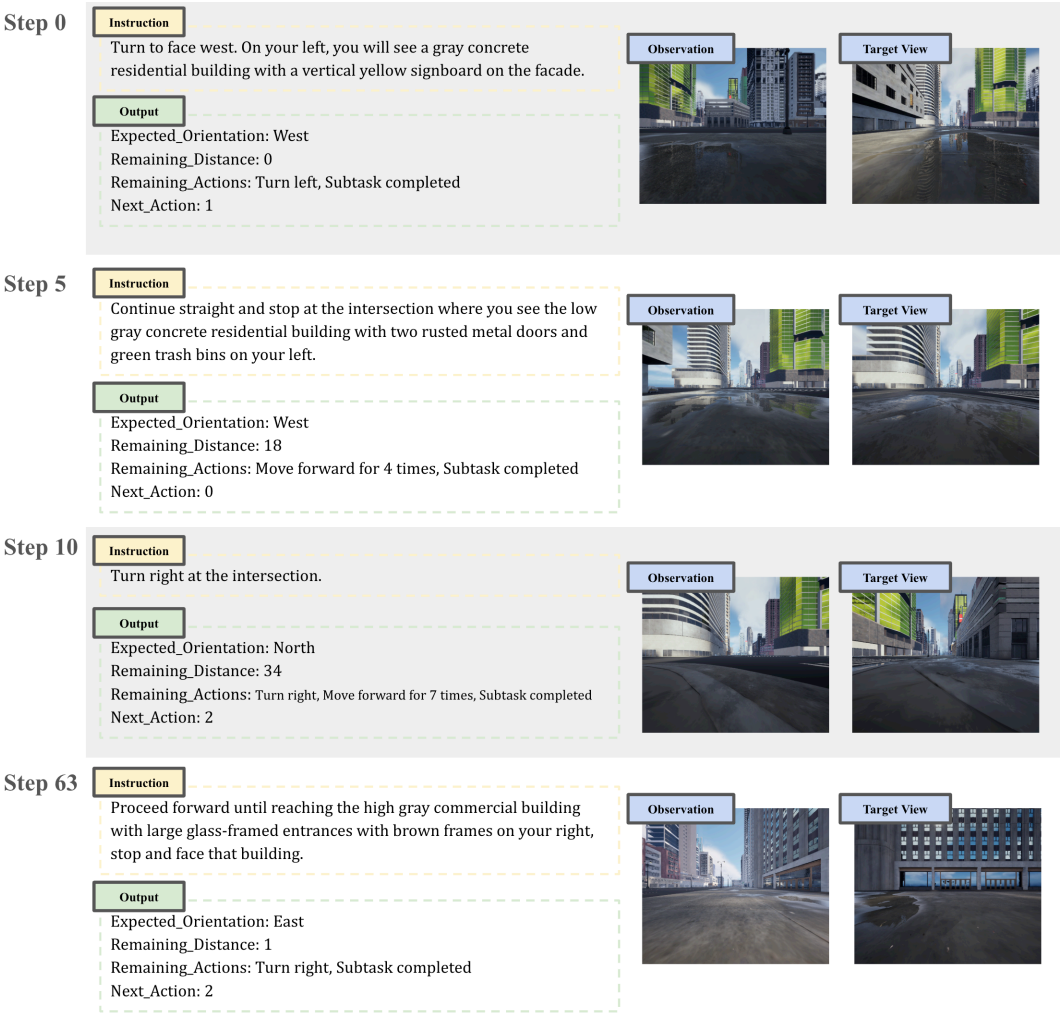

Figure 16: Qualitative result key-step VLM outputs from the finetuned model successfully completing the task

However, finetuning also exhibits certain limitations. First, when the target image corresponds to a location far from the agent's current observation, the lack of meaningful visual overlap makes it difficult for the model to reason about progress. Second, the model is finetuned solely on ground-truth action sequences and lacks robustness to error correction. As a result, if a mistake occurs during navigation, the model struggles to recover, which partially explains the still limited overall success rate.

E.3 SimWorld-MRS

Salient Feature Ignorance In several failure cases, the follower robot generates descriptions that miss highly distinctive features such as store signs, murals, or logos. Instead, the VLM focuses on general elements like a modern building with glass windows, which are insufficient for precise localization. This results in ambiguous matches and large localization errors during memory retrieval by the main robot.

Failure of Instruction Following The issues observed in single-agent instruction following often persist in this setting and are further exacerbated by the absence of a visual hint, making accurate instruction execution more challenging. Although repeated communication can partially correct navigation drift, the task may still fail if the follower agent stops at a non-landmark building, as the main robot will be unable to localize it for subsequent rendezvous planning.

Lack of Failure-Awareness While executing instructions, the follower robot often struggles to determine whether it has become lost. Even when the subtask has not been completed, and the action history indicates that the robot has already moved forward for an extended sequence, the model tends to continue moving forward until the task termination conditions are met. As a result, the robot fails to recognize its deviation in time to trigger communication for goal correction, ultimately leading to out-of-bounds behavior or exceeding the maximum allowed number of steps.

F Code Availability Statement

The implementation of our system builds on a codebase developed through multi-institutional collaboration, with components that are difficult to anonymize due to dependency structures and prior repository history. To maintain the integrity of the double-blind review process, we have withheld the release of the code at this stage. We are committed to open science and will publicly release the complete codebase, along with detailed documentation and instructions for reproduction, upon acceptance.

G LLM Usage Statement

We utilize several large multimodal models (VLMs) as core components of our study, including GPT-4o, Gemini 2.5 Flash, Gemini 2.0 Flash, Qwen2.5-VL-72B-Instruct, Qwen2.5-VL-7B-Instruct, InternVL-78B, and Gemma3-27b-it. These models are evaluated within our simulator-based benchmarks to investigate their embodied navigation capabilities and multi-agent communication performance. The LLMs are responsible for interpreting visual-linguistic instructions, reasoning about spatial environments, and generating actions or dialogue.

Furthermore, we fine-tune Qwen2.5-VL-7B-Instruct on our proposed training dataset to assess the effectiveness of task-specific supervision. Since the models play a central role in both methodology and experimental analysis, and significantly influence the reported results, we declare their usage as integral to the core of this research.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract summarizes the key features of our work, focusing on what differentiates our simulator from existing onesthe main focus of this paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed limitations in the conclusion.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We will open-source our gym environment, the agent's prompt, backend, and model. Please check the details of our two benchmarks and the baselines implementation in Appendix C and Appendix D

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will open-source our code as mentioned in Appendix Fand the details of baseline implementation can be found in Appendix C.5 and Appendix D.4.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so No is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have specified that 33 percent of buildings in the training set are excluded from testing environment.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Each experiment is run twice to account for variability, and results are reported accordingly. While limited in sample size, this provides an initial estimate of consistency.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We use on a headless machine with an AMD EPYC 9534 CPU, L40S GPU, 64GB RAM

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We follow the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: While we do not explicitly discuss societal impacts, we believe our simulator and benchmark can positively contribute to research in embodied AI, and we do not foresee any negative societal consequences from this work.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work has no misuse risk.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes. All assets used in our work were purchased from the official Unreal Engine Marketplace (fab.com), and we fully comply with their licensing terms. No assets were used for any unauthorized or additional 3D asset generation.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Yes. We introduce a 20K training dataset to support vision-language navigation tasks, and detail of it can be found in appendix C.4.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We do not involve any research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We do no involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We provide a detailed explanation in the experimental section on how large language models (LLMs) and vision-language models (VLMs) are used to conduct experiments, which form a core component in our benchmark. Details of it can be found in appendix G