

519 A SimWorld Simulator

520 A.1 Python Package

521 A.1.1 Layout Generation

522 The pipeline of our procedural generation pipe is shown in Figure 7. We have three generation stages
523 which support generating road, building and detail elements in scene graph of city. As Algorithm 1
524 shows, we use the quad-tree data structure manage the hierarchical structure of scene graph which
525 enables us to support space retrieval function in the language control environment module and support
526 generated different tasks easily.

Algorithm 1: Layout Generation Algorithm with QuadTree

Input: Configuration parameters

Output: Final merged QuadTree representing the city layout

```
1 Function RoadMapGeneration(config):  
2   Initialize road network grid;  
3   Create a new QuadTree for the road map;  
4   Place primary roads and secondary roads;  
5   Smooth intersections;  
6   Insert roads into the QuadTree;  
7   return roadQuadTree  
  
8 Function BuildingGeneration(config, roadQuadTree):  
9   Create a new QuadTree for the buildings;  
10  Partition land into plots based on road network;  
11  Generate building types, shapes, heights;  
12  Insert building geometries into the QuadTree;  
13  return buildingQuadTree  
  
14 Function DetailGeneration(config, roadQuadTree, buildingQuadTree):  
15  Create a new QuadTree for the details;  
16  For each building, add facade details (windows, doors, decorations);  
17  For each road, add lanes, crosswalks, signals;  
18  Add environmental elements (trees, benches, lights);  
19  Insert details into the QuadTree;  
20  return detailQuadTree  
  
21 Function MergeQuadTrees(roadQuadTree, buildingQuadTree, detailQuadTree):  
22  Initialize a new QuadTree for the complete city;  
23  Merge roadQuadTree into city QuadTree;  
24  Merge buildingQuadTree into city QuadTree;  
25  Merge detailQuadTree into city QuadTree;  
26  return finalQuadTree  
  
27 Main Procedure;  
28 roadQuadTree  $\leftarrow$  RoadMapGeneration(config);  
29 buildingQuadTree  $\leftarrow$  BuildingGeneration(config, roadQuadTree);  
30 detailQuadTree  $\leftarrow$  DetailGeneration(config, roadQuadTree, buildingQuadTree);  
31 finalQuadTree  $\leftarrow$  MergeQuadTrees(roadQuadTree, buildingQuadTree, detailQuadTree);  
32 return finalQuadTree
```

528 A.1.2 Waypoint System

529 To facilitate robust agent navigation and path planning within the simulated environment, SIMWORLD
530 incorporates a customizable waypoint system (Figure 8b) that serves as the foundational representation
531 of navigable space. This system is initially constructed using a coarse-grained waypoint skeleton
532 derived from the output of the **Layout Generation** pipeline. Specifically, the pipeline produces
533 high-level geometric features of the environment, such as road centerlines and intersection points,
534 which are then used to define the topological structure of the navigation graph.

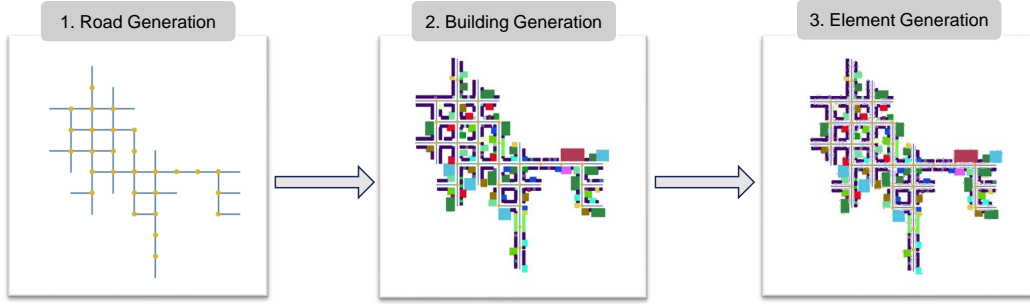


Figure 7: **Generation Pipeline for Static World Initialization.** The process consists of four stages: (1) road network generation, (2) procedural building placement and (3) insertion of street-level elements.

Building upon this base structure, we procedurally generate additional navigable elements, including traffic lanes, sidewalks, and crosswalks. These elements are instantiated by applying spatial offsets relative to the underlying road geometry, enabling the delineation of separate navigation channels for different agent types (e.g., vehicles vs. pedestrians). The result is an initial set of discrete waypoints that capture the essential connectivity of the environment. At this stage, all waypoints are categorized as “intersection” waypoints, reflecting their origin from key structural nodes in the road network.

To improve the granularity and realism of agent motion, we further enrich the navigation graph by interpolating fine-grained waypoints between each pair of adjacent intersection waypoints. These interpolated waypoints are labeled as “normal” waypoints and serve to densify the navigation graph, thereby allowing agents to traverse smoother and more continuous paths. The density of these fine-grained waypoints is configurable through parameters such as interpolation step size and spatial offset magnitude, which can be tuned to suit different simulation scenarios.

This hierarchical waypoint system provides a flexible trade-off between simulation performance and trajectory fidelity. By adjusting the level of waypoint granularity, users can optimize for either computational efficiency or motion realism, depending on the specific demands of downstream tasks such as autonomous navigation, traffic simulation, or human-agent interaction modeling.

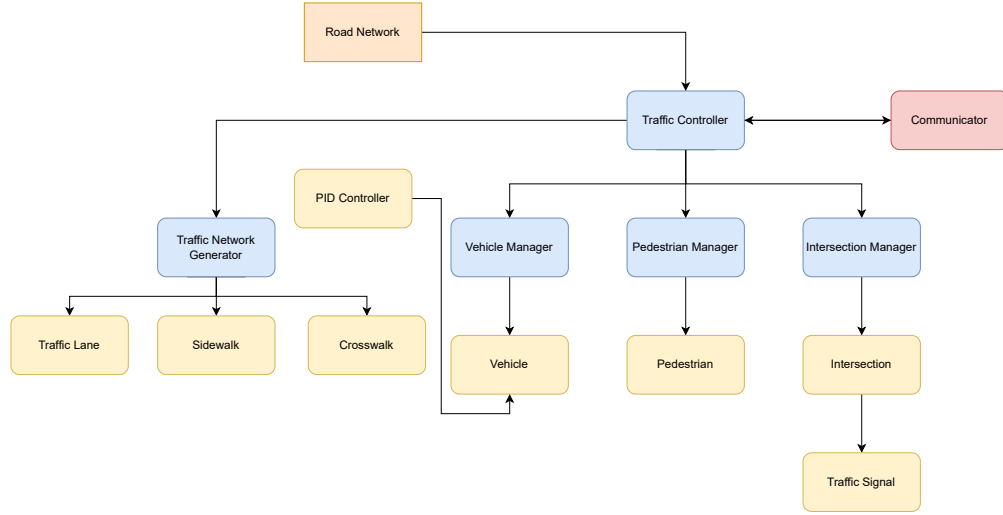
A.1.3 Traffic System

The traffic system is a core component of our simulator, responsible for simulating dynamic road usage by both vehicles and pedestrians. It enables the representation of realistic traffic flow, including vehicle generation, path planning, intersection control, and pedestrian behavior. By managing road interactions and traffic signals, this system supports complex urban scenarios such as congestion, pedestrian crossings, and traffic light coordination, providing a critical foundation for evaluating urban infrastructure and mobility policies.

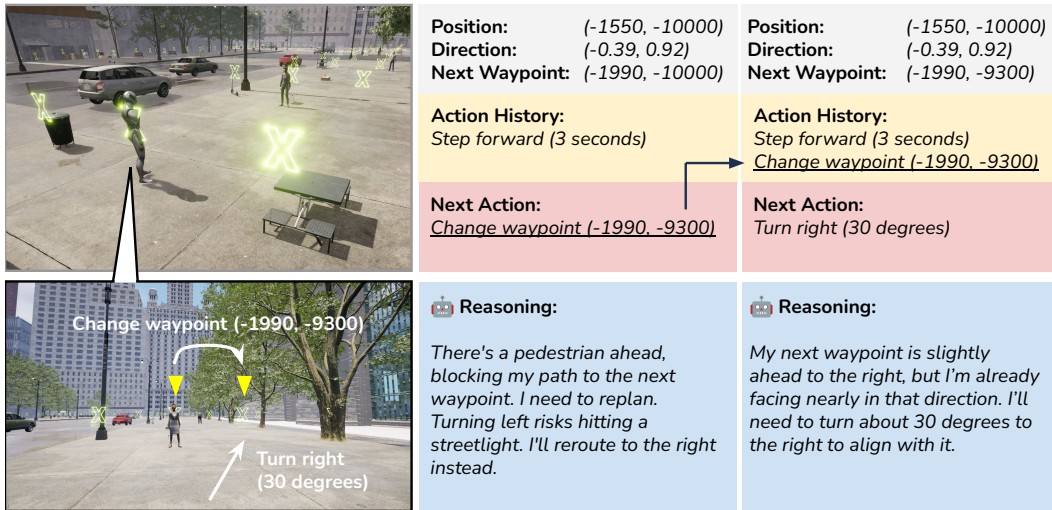
The traffic system operates on a layout graph provided by the **Layout Generation**, designed to support traffic simulation for any city layout. Based on the coarse-grained waypoints and detailed traffic routes, including traffic lanes, sidewalks, and crosswalks, provided by **Waypoint System**, we sample vehicles, pedestrians and traffic signals in the city. Three managers handle populating the traffic elements:

- **VehicleManager**: Samples positions along the traffic lanes to spawn vehicles, assigning them predefined or dynamically generated routes through the network.
- **PedestrianManager**: Places pedestrians along sidewalks and controls their movement patterns, including crossing decisions at intersections.
- **IntersectionManager**: Identifies intersections within the route network and installs traffic signal agents that control the right-of-way based on configurable timing policies or adaptive logic.

Together, these components form a fully functional traffic simulation pipeline, allowing the virtual city to support realistic and reactive mobility behavior.



(a) Architecture of Traffic System.



(b) Waypoint System and Local Action Planner. The waypoint system provides spatial guidance by specifying the positions and orientations of intermediate navigation subgoals. Based on this information, agents can perform high-level reasoning to plan their behavior. A local action planner operates in conjunction with the waypoint system, generating low-level action suggestions—such as path following or obstacle avoidance—that help agents navigate between waypoints effectively.

Figure 8: Overview of Traffic and Waypoint Systems.

As shown in Figure 8a, the traffic system adopts a modular architecture centered around a top-level **TrafficController**, which coordinates the behavior of three specialized managers: **VehicleManager**, **IntersectionManager**, and **PedestrianManager**.

To enable real-time simulation, the system integrates with Unreal Engine through the **Communicator**, which provides bidirectional communication between the traffic logic and the simulation environment. This allows the system to send control signals and receive state updates from virtual actors within the engine.

A.1.4 Local Action Planner

Recent research on AI agents incorporate action planning modules into their agent’s architecture. In our work, we integrate such a module into the simulator itself, providing a useful interface for LLM agents to use and assist the agents’ high-level long-horizon reasoning. LLM-Planner [39]

uses visual information of the environment for decomposing high level plan into low level actions. CoELA [52] uses a rule-based only executor for execution. Compare to their similar module, ours are more comprehensive in that we include rule-based and LLM as executor; we receive both information in the form of natural language as well as in the form of visual; our local action planner module also extendable to corresponding with our extendable simulator. Our local action planner module is divided into two part, parser and executor. Parser simply receive the high-level plan from user LLM and parse the high-level plan into low-level executable actions, while executor is responsible of executing the parsed low-level actions one by one given the environment information in the simulator. Considering the variety of needs from different researchers, we implement local action planner executor module using two different ways: rule-based and visual based. For perceiving the environment of the simulator: rule-based local action planner will receive abstract city layout information, and visual-based local action planner will capture the visual information from the environment directly, which allows the module to be adapted to different visual language models and further improve the comprehensiveness. For non-atomic action like navigation, we integrate a route planner based on the Dijkstra’s algorithm[8] for rule-based local action planner, which provides ground-truth solutions for navigation tasks. This route planner serves as a supporting tool for the local action planner module, ensuring that natural language descriptions of activities can be reliably converted into executable low-level action sequences. For visual-based local action planner on the other hand, we simply feed the visual information to backbone visual language model and let the model decide what to do next to test the capability of visual language model.

For example, when local action planner receives a user plan such as “go to the nearest chair and sit down,” it decomposes the instruction into an action list: *navigate*, *agent_sit_down*. For the *navigate* action, the rule-based mode first computes the shortest path from the agent’s current position to the nearest chair, which results in a sequence of navigation actions such as *navigate*(0, 1), *navigate*(1, 10), and *navigate*(10, 10), where (10, 10) is the location of the chair. Once the agent reaches the chair, local action planner proceeds to the next action, *agent_sit_down*, and completes execution as the action list becomes empty.

609 A.1.5 LLM-based Environment Editing

SIMWORLD supports real-time, language-driven environment editing, enabling the creation of open-ended, generative worlds through natural language. Users interact via a Python interface, issuing instructions such as “add a tree next to the hospital” or “place a red sports car near the fountain.” To handle such commands, we introduce a dual-stage pipeline: (1) asset retrieval and placement, and (2) text-to-3D asset generation.

For asset retrieval and placement, given an instruction like “Add a bench near the cafe; I can see a museum and a skyscraper nearby,” an LLM first parses the target asset, spatial anchor (e.g., “the cafe”), and contextual landmarks (“museum” and “skyscraper”). Candidate reference assets are located via semantic search on the scene graph and ranked by comparing the described surroundings with actual neighbors using embedding-based similarity. The top-ranked reference point determines the target location. The desired asset is then retrieved from the library using CLIP-based[37] matching and placed accordingly.

If no suitable asset is found in the database, we fall back to a text-to-3D generation module. Given a prompt such as “a red sports car,” the system generates a new asset using a generative model, converts it into a usable format, and integrates it into the scene in real time.

This approach allows for flexible, scalable environment editing that is both semantically grounded and spatially coherent—laying the foundation for open-ended simulation and interactive world modeling.

627 A.2 Communicator and Seneraios Creation

628 A.2.1 Communicator

Inspiring by UnrealCV[36], the Communicator module serves as the bridge between Python-based simulation logic and Unreal Engine’s real-time 3D environment. Implemented in both Python and C++, it uses UnrealCV to establish a TCP connection, enabling asynchronous communication between the two sides.

633 We extended the basic communication protocol by defining a custom set of commands for scene
634 control, actor manipulation, and data querying. These commands are encoded in JSON format and
635 passed through the TCP channel. For instance, Python can instruct UE to spawn an actor at a specific
636 location, retrieve the current pose of a pedestrian, or trigger motion events on a robot.

637 The Communicator plays a central role in the simulation loop: Python uses it to update the virtual
638 world, while UE reports back environmental feedback or visual state. This design allows us to
639 decouple logic computation from rendering, achieving both flexibility and modularity.

640 A.2.2 Task Suite Workflow



(b) **Workflow for Constructing a Task Suite.** We implement a delivery task as a representative example within the SIMWORLD simulator. The full task suite includes five key components: (1) Dataset, (2) Simulator, (3) LLM, (4) Platform, and (5) Evaluator.

Figure 9: SIMWORLD Assets and Task Suite Workflow.

Action	Category	Description
Move Forward	Navigation	Keep moving in the current direction
Step Forward/Backward	Navigation	Step forward/backward for a fixed time
Rotate	Navigation	Turn the body to face a new direction
Stop	Navigation	Stop moving
Look Up/Down	Observation	Adjust the gaze upward/downward by a degree
Focus	Observation	Adjust the field of view
Pick Up	Object Interaction	Grasp and lift an object
Drop Off	Object Interaction	Release a held object at the target location
Sit Down	Object Interaction	Transition to a seated position
Stand Up	Object Interaction	Rise from a seated position
Open Door	Object Interaction	Interact to open a door
Enter Car	Object Interaction	Get into a vehicle
Exit Car	Object Interaction	Leave a vehicle
Carry Heavy Object	Object Interaction	Transport a heavy object
Put Down Heavy Object	Object Interaction	Place a heavy object on the ground
Ride Scooter	Object Interaction	Control and ride a scooter
Have Conversation	Social Action	Exchange verbal communication
Point Direction	Social Action	Gesture to indicate direction
Wave Hand	Social Action	Signal or greet with a hand wave
Discuss	Social Action	Engage in dialogue or explanation
Argue with Body Language	Social Action	Express disagreement using gestures
Dance	Social Action	Perform rhythmic movement with the body
Make Phone Call	Social Action	Simulate or engage in a phone conversation

Table 4: **Comprehensive List of Actions with Categories and Descriptions.**

A.3 Unreal Engine Integration

A.3.1 Action Space

Table 4 presents the complete set of low-level actions supported by SIMWORLD.

A.3.2 Assets

Our simulator provides a rich collection of city-scale assets, designed to support realistic and diverse urban simulations. These assets include buildings, trees, street furniture, vehicles, pedestrians, and robots (Figure 9a). All assets are sourced from the Unreal Engine Marketplace to ensure high visual fidelity and performance.

Below is a selection of the assets currently available in our simulator:

- **Buildings:** A variety of architectural styles, including residential, commercial, and industrial structures.
- **Trees:** Multiple tree species with seasonal variations to enhance environmental realism.
- **Street Furniture:** Items such as benches, streetlights, mailboxes, and trash bins to add detail and immersion.
- **Vehicles:** A range of vehicles including cars, buses, trucks, and bicycles, each with accurate scale and animations.
- **Pedestrians:** Human characters with diverse appearances and animations to simulate crowd behavior.
- **Robots:** Humanoid robots for testing autonomous navigation and interaction.

These assets collectively enable the creation of complex, dynamic, and realistic city scenes for simulation, visualization, and research purposes.

In addition to the curated asset library, we provide an **Asset Generation Pipeline** that leverages Text-to-3D models[41] to generate 3D assets directly from natural language descriptions. This tool streamlines the content creation process by translating user prompts into usable Unreal Engine assets, thereby significantly lowering the barrier to customizing city environments.

B Case Study 1 Details

B.1 Formulation

To evaluate the embodied reasoning and physical interaction capabilities of agents within SIMWORLD, we introduce a vision-based navigation task in urban environments. Agents are tasked with reaching a specified destination while avoiding both static obstacles (e.g., trees, benches) and dynamic agents (e.g., pedestrians), necessitating strong spatial reasoning and multimodal perception under uncertainty. The task requires agents to process real-time visual input and make sequential decisions to avoid collisions, respect traffic rules, and reach the goal efficiently.

To analyze the interplay between high-level planning and low-level reactive control, we consider two settings: *with* and *without* a Local Action Planner. In the Local Action Planner setting, agents are guided by a precomputed global route generated via A* search, which omits obstacle information. Agents must navigate from one waypoint to the next along this route while handling local interactions. In the planner-free setting, agents receive only the goal position and are expected to autonomously plan and execute a safe trajectory.

B.2 Action Space

Each agent may choose from the following actions at each decision step:

- **Do Nothing:** Remain stationary.
- **Step:** Move forward or backward for a fixed duration.
- **Turn:** Rotate left or right by any angle (continuous range).
- **Change Next Waypoint:** (Local Action Planner only) Switch to a different candidate waypoint.

B.3 City Map and Task Generation

- **Coarse-grained Map:** Each sidewalk is divided into three waypoints (start, midpoint, end), forming a navigation graph to generate tasks.
- **Fine-grained Map:** Each sidewalk is sampled with ~ 70 evenly spaced waypoints. These serve both as candidate motion targets and obstacle anchor points.
- **Obstacles:** Static objects (trees, trash bins, etc.) are randomly placed on fine-grained waypoints. Dynamic pedestrians follow sidewalk paths at moderate speed with randomized starting positions.

Figure 10 illustrates the task generation pipeline. We begin with a basic road network provided by SimWorld, which includes sidewalks, crosswalks, and intersection nodes. To construct the coarse-grained navigation graph, we interpolate a midpoint for each sidewalk segment, yielding a simplified map structure suitable for high-level planning.

Tasks are generated by randomly sampling start and goal nodes from this graph, with constraints on the number of segments between them to control difficulty. For fine-grained navigation, we densify the original sidewalk geometry by interpolating evenly spaced waypoints. Each sidewalk is divided into four lateral lanes, each containing 17 waypoints, while each crosswalk contains 8 waypoints.

To simulate realistic urban clutter, we place static obstacles such as trees and street furniture along these waypoints. Trees are sampled on the innermost lane (nearest to buildings) of each sidewalk, while other street elements (e.g., benches, trash bins) are distributed across the remaining three lanes. Importantly, for each group of four laterally aligned waypoints (i.e., across the four sidewalk lanes at the same longitudinal position), at least one is guaranteed to remain obstacle-free. This ensures that there is always a traversable option at every step along the sidewalk.

We sample 40 tasks spanning the four difficulty levels:

- Easy (10 tasks), Medium (10 tasks), Hard (10 tasks), Dynamic (10 tasks).
- Each task is defined by a sequence of waypoints $T_i = [W_1, W_2, \dots, W_n]$.

B.4 Baseline Agent Protocols

- **Rule-based Agent:** Follows Local Action Planner waypoints using a fixed stepping strategy. Obey the traffic rules and does not observe visual input or adapt to obstacles.

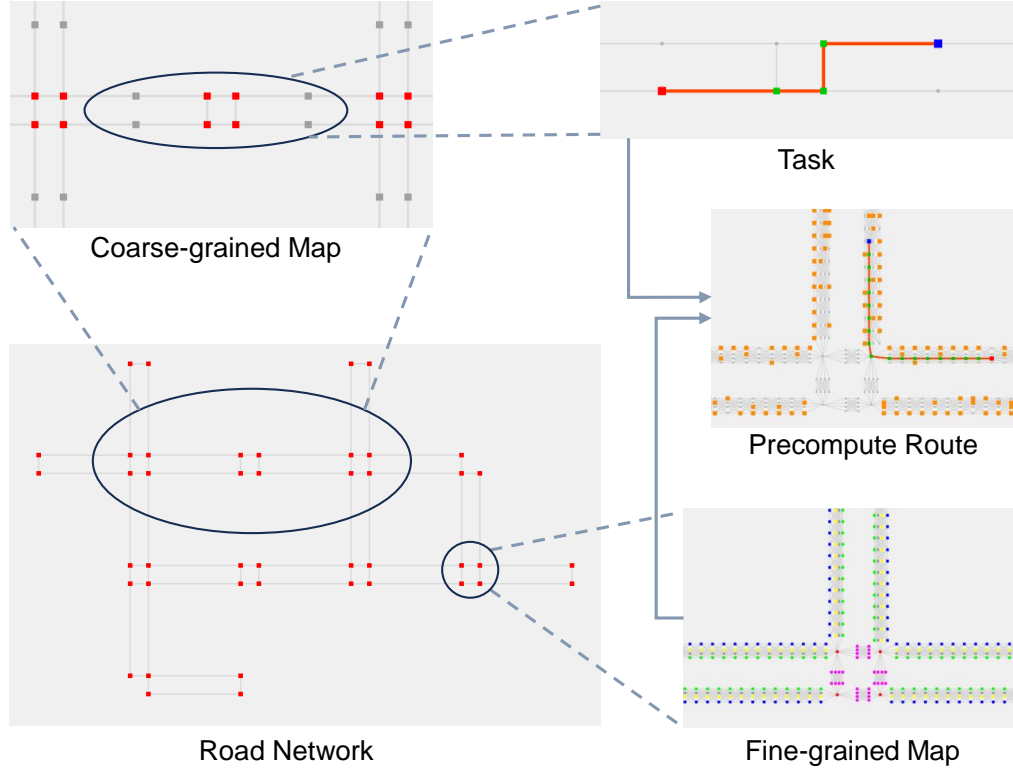


Figure 10: Task Generation Pipeline for Case Study 1.

• **MLLM Agents:**

- Input: current image, last-step image, past 3 actions and feedback, current navigation goal, relative distance and angle to current navigation goal.
- Output: action selected from the predefined space.
- Perception is simulated via RGB screenshots with 90° horizontal FOV.

B.5 Metrics

Throughout the experiment, we evaluate agent performance using the following six metrics, which together capture task success, safety compliance, and decision efficiency:

- **Success Rate (SR):** The percentage of navigation tasks in which the agent successfully reaches its destination without getting stuck, timing out, or deviating significantly from the designated route. This metric reflects the overall effectiveness of the agent’s decision-making and path-following capabilities.

$$SR = \frac{N_{\text{success}}}{N_{\text{total}}}$$

where N_{success} is the number of successfully completed tasks, and N_{total} is the total number of tasks.

- **Collision Count (CC):** The total number of collisions that occur between the agent and either static obstacles (e.g., trees, trash bins) or dynamic entities (e.g., pedestrians). A lower collision count indicates better obstacle avoidance and situational awareness. Collisions are recorded for all episodes, regardless of success or failure.

$$CC = \sum_{i=1}^{N_{\text{total}}} c_i$$

where c_i is the number of collisions in episode i .

- 732 • **Collision Count (Success Only) (CC-S)**: The number of collisions occurring only within successful episodes, which reflects if the agent really avoids the obstacles. Since the CC is possibly low if
 733 the agent get stuck at the very beginning. This metric helps assess how safely the agent completes
 734 tasks when it does reach the goal.
 735

$$\text{CC-S} = \sum_{i \in \mathcal{S}} c_i$$

736 where \mathcal{S} is the set of successful episode indices.

- 737 • **Red-light Violation Rate (RVR)**: The fraction of successful episodes in which the agent crosses an
 738 intersection during a red traffic signal, violating traffic rules. This metric evaluates the agent’s ability
 739 to obey traffic signals and is computed only over successful episodes to ensure fair assessment
 740 under comparable task completion.

$$\text{RVR} = \frac{\sum_{i \in \mathcal{S}} r_i}{|\mathcal{S}|}$$

741 where $r_i = 1$ if a red-light violation occurred in episode i and 0 otherwise, and $|\mathcal{S}|$ is the total
 742 number of successful episodes.

- 743 • **Stuck Rate (STR)**: The percentage of failed episodes in which the agent remains unable to reach
 744 the destination due to either physical immobilization (e.g., blocked by obstacles) or ineffective
 745 decision-making (e.g., repeatedly turning or repositioning without forward movement) over a
 746 period of 2 minutes. STR captures both hard failures (being physically stuck) and soft failures
 747 (behavioral indecision), reflecting challenges in planning and low-level control.

$$\text{STR} = \frac{N_{\text{stuck}}}{N_{\text{failed}}}$$

748 where $N_{\text{failed}} = N_{\text{total}} - N_{\text{success}}$ and N_{stuck} is the number of failed episodes classified as stuck.

- 749 • **Normalized Decision Count (NDC)**: The ratio between the total number of decision-making by
 750 the large model and the total number of fine-grained waypoints provided by the Local Action
 751 Planner, computed over successful episodes. NDC measures the agent’s decision efficiency, with a
 752 lower value indicating more economical use of the model.

$$\text{NDC} = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \frac{d_i}{w_i}$$

753 where d_i is the number of model decisions made in episode i , and w_i is the number of fine-grained
 754 waypoints along the successful path. $|\mathcal{S}|$ is the total number of successful episodes.

- 755 • **Decision Steps in Success (DSS)**: The average number of decision-making steps taken by the
 756 large model in episodes that were successfully completed. DSS measures the typical number of
 757 decisions required to achieve a successful outcome, with a lower value potentially indicating more
 758 efficient or direct decision-making in successful scenarios.

$$\text{DSS} = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} d_i$$

759 where d_i is the number of model decisions made in successful episode i , and $|\mathcal{S}|$ is the total number
 760 of successful episodes.

761 Together, these metrics provide a comprehensive evaluation of an agent’s navigation performance,
 762 safety behavior, and decision-making efficiency in complex urban environments.

763 B.6 Prompt Design

764 B.6.1 With Local Action Planner

System Prompt of Navigation Task with Local Action Planner

```
SYSTEM_PROMPT = """
You are an embodied agent in a 3D simulation environment, where the unit is
centimeters (cm). Your task is to navigate from your current position to a
specified destination safely and efficiently. A path to the destination is
provided as a list of waypoints. You only need to go to the next waypoint
```

765

```

if it's not blocked. Otherwise, you should choose a new next waypoint. For
this task, please only output a parsable json string inside brackets.
Please start your answer with { and end your answer with }. Don't include
any notes or explanations with the output json string.
"""

```

Figure 11: Examples System Prompt of Navigation Task with Local Action Planner.

User Prompt of Navigation Task with Local Action Planner

```

USER_PROMPT = """
Currently you are at {current_position} and your direction is {current_direction}.
Your final destination is {target_position}. Your next waypoint is {
next_waypoint}, among the possible next waypoints: {possible_next_waypoints}.
The next waypoint is {relative_distance:.2f} cm away from you. The
relative angle to the next waypoint is {relative_angle:.2f} degrees (
negative means next waypoint is to your left, positive means next waypoint
is to your right). Your walking speed is 200 cm/s.

You are given two images:
- Previous view (1 step ago): shows what you saw before your last decision.
- Current view: shows what you see now.
Use the two images to understand the changes in your surroundings and help you
make a better decision.

You have the following action history (most recent at the bottom):
{action_history}
Before making your decision, you should consider the history of actions. Avoid
choosing a new waypoint multiple times in a row. If your last action was
choosing a new waypoint, you should now try moving toward it unless it's
truly blocked. Avoid turning around multiple times in a row. You should try
to move forward a little bit before turning around.

You must:
- Avoid obstacles and pedestrians on the street.
- Obey traffic lights when at intersections: stop at red lights, proceed on
green.
- Use both current and previous visual observations to assess your surroundings.
- Only adjust your walking direction when it deviates more than 15 degrees from
the next waypoint.
- Avoid selecting a new waypoint repeatedly or turning around too frequently.
- Never walk forward continuously for more than 5 seconds; 1-5 seconds is
acceptable.

Think step by step and reason about your decision.
- If you are facing your next waypoint and it's not blocked, you should move
forward for a short time (1-5 seconds).
- If you are not facing your next waypoint roughly, you should turn around to
face the next waypoint.
- If your next waypoint is blocked, you should choose a new waypoint from the
possible next waypoints.
- If you get stuck, you can turn around and change your next waypoint or step
backward for a short time (1-5 seconds).
- If you are at an intersection, you should stop and wait for the pedestrian
light to turn green before moving.
- If you need to change your view, you can look up or set your field of view.

Now it's time for you to make a decision. You have the following options:
- 0: Do nothing.
- 1: Step. Must specify duration (max 5 sec) and direction (0 = forward, 1 =
backward).
- 2: Turn. Specify angle (0-180 degrees) and direction (clockwise = true/false).
- 3: Look up. Specify look_up_angle (-90 to 90 degrees).
- 4: Set field of view. Specify fov (0-180 degrees).
- 5: Choose a new waypoint. Must specify one from the possible next waypoints.

Output a single json object string, whose keys are 'choice', 'duration', '
direction', 'angle', 'clockwise', 'look_up_angle', 'fov', 'new_waypoint'.
"""

```

Figure 12: Examples User Prompt of Navigation Task with Local Action Planner.

772 B.6.2 Without Local Action Planner

System Prompt of Navigation Task without Local Action Planner

```

SYSTEM_PROMPT = """
You are an embodied agent in a 3D simulation environment, where the unit is
centimeters (cm). You are good at navigating in a city environment. Your
task is to navigate from your current position to a specified destination
safely and efficiently. For this task, please only output a parsable json
string inside brackets. Please start your answer with { and end your answer
with }. Don't include any notes or explanations with the output json
string.
"""

```

773
774
775 **Figure 13: Examples System Prompt of Navigation Task without Local Action Planner.**

User Prompt of Navigation Task without Local Action Planner

```

USER_PROMPT = """
You are currently at {current_position} and your direction is {current_direction
}. Your final destination is {target_position}. The destination is
approximately {relative_distance:.2f} cm away, and the relative angle to it
is {relative_angle:.2f} degrees (negative = to your left, positive = to
your right). Your walking speed is 200 cm/s.

You are given two images:
- Previous view (1 step ago): what you saw before your last decision.
- Current view: what you see now.
Use these images to understand how your environment is changing and make smart
decisions.

You have the following action history (most recent at the bottom):
{action_history}
Before making your decision, you should consider the history of actions.

The relative distance and angle only give you a rough idea of where the
destination is. You must not walk directly toward the destination without
checking the surroundings. Carefully plan your path instead.

You must:
- Avoid obstacles on the sidewalk.
- Keep walking on the sidewalk. Do not step into the roadway, crash into
buildings, or hit obstacles on your way.
- Use both current and previous visual observations to assess your surroundings.
- Never walk forward continuously for more than 5 seconds; 1-5 seconds is
acceptable.

You are required to make a decision for your next action. You have the following
options:
- 0: Do nothing.
- 1: Step. Must specify duration (max 5 sec) and direction (0 = forward, 1 =
backward).
- 2: Turn. Specify angle (0-180 degrees) and direction (clockwise = true/false).

Output a single json object string, whose keys are 'choice', 'duration', '
direction', 'angle', 'clockwise', 'new_waypoint'.
"""

```

776
777
778 **Figure 14: Examples User Prompt of Navigation Task without Local Action Planner.**

779 B.7 Main Results: Agent Planning With Local Action Planner

780 **Static Obstacle.** We evaluate three levels of task difficulty (Easy, Medium, Hard) under static obsta-
781 cles with a local action planner. Time limits are set to 15 minutes for Easy and Medium tasks, and 25
782 minutes for Hard tasks. Four MLLMs are evaluated: GPT-4o-mini, GPT-4o, Claude-3.7-Sonnet,
783 and Gemini-2.5-Pro (gemini-2.5-pro-preview-03-26). Results are shown in Table 2, with key
784 insights discussed below:

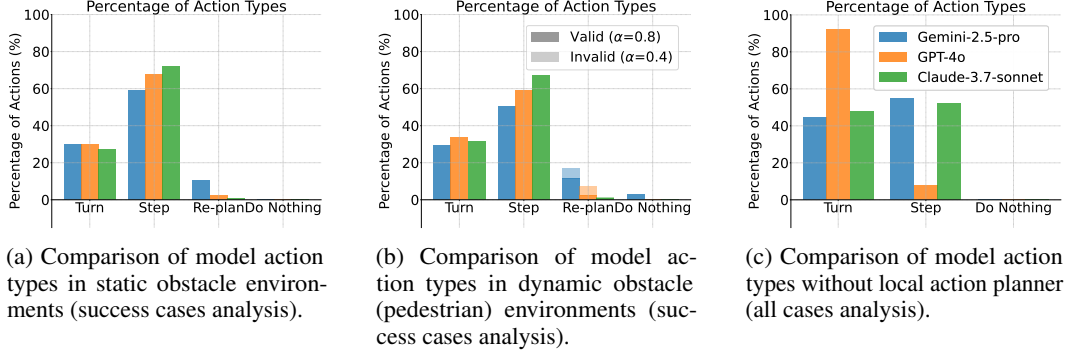


Figure 15: Distribution of Action Types for Navigation Models in Environments with/without Obstacles and Different Obstacle Types.

785 *Model capability is a decisive factor in task success.* State-of-the-art models such as GPT-4o,
786 Claude-3.7-Sonnet, and Gemini-2.5-Pro consistently outperform the rule-based baseline across
787 all three difficulty levels. These advanced MLLMs achieve higher SR, along with lower CC and
788 CC-S, indicating their ability to understand the environment and avoid obstacles more effectively. In
789 contrast, GPT-4o-mini performs even worse than the rule-based baseline, with lower SR and higher
790 CC and CC-S, suggesting that it struggles to follow task rules and may be prone to hallucinations or
791 unreasonable actions.

792 *MLLMs demonstrate limited spatial reasoning.* Although larger MLLMs attain higher SR across all
793 tasks, we do not observe a corresponding drop in CC or CC-S. This implies that while these models
794 attempt to avoid obstacles, they often fail—frequently grazing or barely touching the edges of objects.
795 This behavior suggests that current MLLMs possess strong visual recognition abilities but lack a
796 robust understanding of spatial relationships and depth.

797 *Different models exhibit distinct strengths.* Our analysis reveals that individual models have spe-
798 cialized advantages. For instance, GPT-4o and Claude-3.7-Sonnet show higher STR, indicating
799 that its failures are more often due to getting stuck rather than poor decisions, suggesting better
800 decision-making but weaker spatial adaptability. In contrast, Gemini-2.5-Pro shows a lower STR,
801 implying that it struggles more with planning and timely decision-making. These trends are consistent
802 with their respective SR, CC, and CC-S values: GPT-4o and Claude-3.7-Sonnet excel in making
803 quick and precise decisions, while Gemini-2.5-Pro shows better spatial comprehension. Figure
804 15a shows that Gemini-2.5-Pro performs "choose a new waypoint" more than other models, which
805 further explains that Gemini-2.5-Pro are more sensitive to obstacle avoidance. Additionally, lower
806 NDC supports these findings by showing which models complete tasks using fewer steps, reflecting
807 planning efficiency.

808 *MLLMs fail to function as truly embodied agents.* Although MLLM-driven agents do not achieve
809 100% RVR in the table, additional experiments on the Hard tasks—where all pedestrian signals were
810 kept red—reveal a more fundamental issue: agents crossed streets despite the red lights. To correctly
811 perceive the signal state, an agent must look up and focus on the signal. Upon detecting a red light, it
812 should repeatedly choose the *do nothing* action until the signal turns green. However, experimental
813 results show that none of the agents chose to adjust their view, and very few opted to do nothing,
814 indicating that MLLMs do not recognize their ability to adjust visual sensors. Instead, they react only
815 based on the input they are given. This suggests that LLM-based agents require substantial additional
816 components to behave like real embodied agents—the LLM alone is not sufficient.

817 **Dynamic Obstacles.** Table 5 presents the performance of different models on dynamic navigation
818 tasks. GPT-4o and Claude-3.7-Sonnet demonstrate superior performance, achieving higher SR
819 and lower NDC. This suggests that these models are capable of making faster and more decisive
820 actions, enabling them to complete navigation tasks efficiently in dynamic environments. In contrast,
821 Gemini-2.5-Pro experiences a significant drop in SR and exhibits higher NDC values—comparable
822 to those of GPT-4o-mini—while also recording the lowest values in both CC and CC-S among all
823 baselines.

824 Figure 15b reveals that Gemini-2.5-Pro frequently switches waypoints and tends to choose valid
825 ones, suggesting stronger spatial sensitivity. However, this caution often leads to hesitation or

826 stalls near moving obstacles, highlighting a trade-off between spatial adaptability and robust task
827 completion.

Table 5: **Performance Comparison under Dynamic Obstacles.** SR: Success Rate, CC: Collision Count, CC-S: Collision Count (Success Only), NDC: Normalized Decision Count. Human: Human Collision Count, Obj: Object Collision Count.

Baseline	SR↑	CC↓			CC-S↓			NDC↓
		Human	Obj	All	Human	Obj	All	
Rule-based	80.00	2.93	2.97	5.90	2.54	2.83	5.38	N/A
GPT-4o-mini	26.67	1.37	3.80	5.17	1.12	2.75	3.88	2.43
GPT-4o	86.67	1.13	2.63	3.77	1.04	2.54	3.58	2.30
Claude	93.33	1.00	2.57	3.57	1.04	2.36	3.39	2.27
Gemini	56.67	0.33	1.60	1.93	0.35	1.18	1.53	2.77

Table 6: **Performance Without Local Action Planner.** SR: Success Rate, CC: Collision Count, CC-S: Collision Count (Success Only), DSS: Decision Steps in Success. Bldg: Building Collision Count, Obj: Object Collision Count.

Baseline	SR↑	CC↓			CC-S↓			DSS↓
		Bldg	Obj	All	Bldg	Obj	All	
GPT-4o	0.00	4.50	0.00	4.50	N/A	N/A	N/A	N/A
Claude	56.67	18.50	1.63	20.13	11.71	2.18	13.88	58.71
Gemini	60.00	15.83	0.40	16.23	10.44	0.50	10.94	68.72

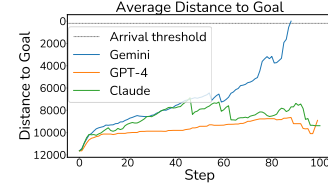


Figure 16: **Step-wise Average Destination Distance across Tasks.** GPT-4o often exhibits early stagnation, making limited goal progress, while Claude-3.7-Sonnet shows fluctuating effectiveness. Gemini-2.5-Pro stands out by demonstrating the most stable and continuous reduction in goal distance, highlighting its superior planning and navigation capabilities without a local action planner.

828 B.8 Ablation Study: Without Local Action Planner

829 To evaluate the robustness of MLLM-driven agents under more challenging conditions, we remove
830 the local action planner and re-evaluate their performance on Medium-level tasks. In this setting,
831 agents are only provided with the final destination instead of a pre-defined series of waypoints. They
832 are required to not only avoid collisions but also plan an efficient trajectory autonomously. All agents
833 are restricted to a maximum of 90 decision steps per task.

834 We compare three state-of-the-art MLLMs: GPT-4o, Claude-3.7-Sonnet, and Gemini-2.5-Pro
835 (gemini-2.5-pro-preview-05-06). As shown in Table 6, GPT-4o fails to complete any task successfully
836 (SR = 0), while Claude-3.7-Sonnet achieves a 56.67% SR and it also achieves the lowest DSS
837 of 58.71, indicating more efficient decision-making. Gemini-2.5-Pro outperforms both with the
838 highest SR (60.0%) and the lowest overall CC (16.23), including significantly fewer building and
839 object collisions.

840 Figure 16 illustrates the distance-to-goal progression for each model from a single experimental run.
841 The behavioral patterns underlying these distance trends are further elucidated by Figure 15c, which
842 shows GPT-4o and Claude-3.7-Sonnet frequently engaging in non-progressive turning actions,
843 whereas Gemini-2.5-Pro predominantly maintain forward movement.

844 B.9 Chain of Thought Example

845 This section, we showcase a representative chain of thought from Gemini-2.5-Pro as it navigated
846 the Easy level task. This demonstration is provided to give readers a transparent view of the model’s
847 deliberative process when prompted to "think step-by-step." The following transcript captures the
848 VLM’s sequential reasoning, from understanding the prompt to formulating its next action.

Example Chain of Thought from Gemini-2.5-Pro on a Successful Easy Task

Prompt: You are currently at ...
Action: Turn around
Reasoning: The relative **angle** to the next waypoint is 144.77 degrees, **which** is greater than 15 degrees. So I need to turn to face the next waypoint. The **path** to the waypoint is **clear** based on the images.

Prompt: You are currently at ...
Action: Step forward
Reasoning: I am facing the next waypoint (relative **angle** 0.77 degrees) and the **path** is **clear**. My last action was a turn, so now I should move forward. The distance to the waypoint is 948.68 cm, and my speed is 200 cm/s, so I can reach it in 4.74 seconds, **which** is within the allowed duration.

Prompt: You are currently at ...
Action: Turn around
Reasoning: The relative **angle** to the next waypoint is -16.89 degrees. Since the absolute value of this **angle** (16.89) is greater than the 15-degree threshold, I need to adjust my direction. The negative **sign** indicates the waypoint is to my left, so I will turn counter-clockwise. The current **view** shows no obstacles directly in the **path** to the waypoint. My last action was stepping forward, so a turn action is not repetitive.

Prompt: You are currently at ...
Action: Step forward
Reasoning: I am facing the next waypoint (relative **angle** -0.89 degrees). The **path** is **clear**. My last action was turning, so I should move forward now. The distance to the waypoint is 896.46 cm, and my speed is 200 cm/s, so the time to reach is $896.46 / 200 = 4.48$ seconds.

Prompt: You are currently at ...
Action: Step forward
Reasoning: The **path** to the next waypoint is **clear** and I am facing it. The relative **angle** is small (-2.55 degrees), so no turn is needed. I will step forward to **get** closer to the waypoint.

Prompt: You are currently at ...
Action: Step forward
Reasoning: The **path** to the next waypoint is **clear**, and the relative **angle** is small (-7.56 degrees). Therefore, I will step forward towards the waypoint. The time to reach the waypoint at 200 cm/s is approximately 4.5 seconds, **which** is within the allowed duration.

Prompt: You are currently at ...
Action: Turn around
Reasoning: My current target waypoint is Vector(x=-1850.0, y=-4800.0), **which** I selected in my previous action. The relative **angle** to this waypoint is 179.06 degrees. This is significantly greater than the 15-degree threshold, indicating I am facing almost directly opposite to the waypoint. The current visual information shows a tree immediately in front of me, blocking my current **path**. By turning 179.06 degrees to the right (clockwise, as the relative **angle** is positive), I will align myself with the waypoint. The tree currently in front will then be behind me. According to the guidelines, **if** the last action was choosing a new waypoint, I should attempt to move towards it. Turning to face it is the first step. This action does not violate rules against repeated waypoint selection or consecutive turns, as my action history is CHOOSE_WAYPOINT, TURN, CHOOSE_WAYPOINT, and now TURN.

Prompt: You are currently at ...
Action: Step forward
Reasoning: I am currently facing the next waypoint (Vector(x=-1850.0, y=-4800.0)) with a relative **angle** of 0.06 degrees, **which** is well within the 15-degree tolerance. The current **view** shows that the **path** to the waypoint is **clear**. My last action was a turn, and prior to that I chose a new waypoint and turned. To make progress and avoid repeated turns, I should now move forward. The next waypoint is 118.72 cm away. The instructions state that a step duration of 1-5 seconds is acceptable. Therefore, I will choose to step forward **for** 1.0 second. At a speed of 200 cm/s, this means I will move 200 cm, **which** will overshoot the current waypoint. This is acceptable given the duration constraint, and the **system** will provide a new state afterwards.

Prompt: You are currently at ...
Action: Turn around
Reasoning: The relative **angle** to the next waypoint is 77.64 degrees, **which** is greater than 15 degrees. Therefore, I need to turn to face the waypoint. Since the **angle** is positive, I will turn right (clockwise).

Prompt: You are currently at ...
Action: Choose waypoint
Reasoning: The current next waypoint Vector(x=-1550.0, y=-3900.0) is blocked by a barrier visible in the current **view**. I am choosing a new waypoint Vector(x=-1250.0, y=-3900.0), **which** is to the right of the current blocked **path**. Visual inspection suggests this **path** might be clearer.

Prompt: You are currently at ...
Action: Turn around
Reasoning: My last action was choosing a new waypoint. The current waypoint is Vector(x=-1250.0, y=-3900.0). The relative **angle** to this waypoint is -15.62 degrees. Since the absolute value of the relative **angle** (15.62 degrees) is greater than the 15-degree threshold, I need to turn to face the waypoint. The negative **angle** indicates the waypoint is to my left, so I will turn counter-clockwise by 15.62 degrees. The current **view** shows no immediate obstacles in that direction.

Prompt: You are currently at ...
Action: Turn around
Reasoning: The last action was choosing a new waypoint. The current waypoint is Vector(x=-1550.0, y=-3900.0). The relative **angle** to this waypoint is 15.64 degrees, **which** is greater than the 15-degree threshold **for** turning. The waypoint is to my right, so I need to turn clockwise. The **path** to the waypoint appears **clear** from the current **view**.

Prompt: You are currently at ...
Action: Step forward
Reasoning: I am facing the next waypoint (relative **angle** 0.64 degrees) and it is not blocked. My last action was a turn, and before that I chose a new waypoint. Therefore, I should now move towards the current waypoint. The distance is 928.38 cm and speed is 200 cm/s, so it would take about 4.64 seconds to reach it. I will step forward **for** 4.5 seconds.

Prompt: You are currently at ...
Action: Choose waypoint
Reasoning: The current next waypoint Vector(x=-1550.0, y=-3900.0) is visibly blocked by a construction barrier. I am choosing a new waypoint, Vector(x=-1850.0, y=-3900.0), **which** is to the left of the current blocked waypoint and appears to offer a **clear path** around the barrier based on the current **view**.

Prompt: You are currently at ...
Action: Choose waypoint
Reasoning: My last action was choosing a new waypoint. The current next waypoint Vector(x=-1850.0, y=-3900.0) is 19.41 degrees to my right. The current **view** shows a barrier to the right, **which** likely blocks this waypoint. Therefore, I will choose a new waypoint that is to the left of the current one and the barrier, **which** is Vector(x=-2300.0, y=-3900.0).

Prompt: You are currently at ...
Action: Step forward
Reasoning: I am facing the next waypoint (relative **angle** 0.78 degrees), and the **path** appears **clear** based on the current and previous views. My last action was a turn, preceded by choosing a new waypoint. According to the rules, I should now move towards the current waypoint. I will step forward **for** 5.0 seconds to **get** closer to the waypoint, **which** is 1046.26 cm away and my speed is 200 cm/s.

Prompt: You are currently at ...
Action: Turn around
Reasoning: My last action was choosing a new waypoint. The current waypoint is not blocked. The relative **angle** to the waypoint is -24.59 degrees, **which** is greater than 15 degrees. Therefore, I need to turn left to face the waypoint.

Prompt: You are currently at ...
Action: Choose waypoint
Reasoning: The current waypoint Vector(x=-1850.0, y=-3900.0) is blocked by a construction barrier visible in the current **view**. Therefore, a new waypoint needs to be chosen. Vector(x=-1550.0, y=-3900.0) is selected as it is to the right of the barrier, requires a moderate turn (+19.3 degrees), and progresses towards the final destination. The right side appears **more** open based on the **image**.

Prompt: You are currently at ...
Action: Turn around
Reasoning: My last action was choosing a new waypoint. The relative **angle** to the current waypoint is -19.36 degrees, **which** is greater than 15 degrees. Therefore, I need to turn to face the waypoint. The obstacle in the **view** is distant and does not immediately block the **path for** turning.

Prompt: You are currently at ...
Action: Turn around
Reasoning: The last action was choosing a new waypoint. The current waypoint is not blocked based on the visual information. The relative **angle** to the next

waypoint is -16.62 degrees, **which** is greater than the 15-degree threshold **for** turning. Therefore, I will turn left by 16.62 degrees to align with the next waypoint.

Prompt: You are currently at ...

Action: Step forward

Reasoning: The next waypoint is 441.28 cm away and the relative **angle** is -1.47 degrees, **which** is small. The **path** appears **clear** from the current and previous views. The last action was stepping forward, so continuing to move towards the waypoint is appropriate. The calculated duration to reach the waypoint is 2.2 seconds, **which** is within the acceptable range.

Prompt: You are currently at ...

Action: Turn around

Reasoning: The relative **angle** to the next waypoint is 47.34 degrees, **which** is greater than 15 degrees. Therefore, I need to turn to face the waypoint. The waypoint is to my right, so I will turn clockwise.

Prompt: You are currently at ...

Action: Step forward

Reasoning: My last action was to turn towards the waypoint. I am now facing the waypoint (relative **angle** 0.34 degrees). The **path** to the waypoint appears **clear** in both current and previous views. The distance to the waypoint is 939.89 cm. At a speed of 200 cm/s, it would take about 4.7 seconds. I will step forward **for** 4.5 seconds, **which** is within the 1-5 second limit and will **get** me **close** to the waypoint.

Prompt: You are currently at ...

Action: Step forward

Reasoning: The relative **angle** to the next waypoint is -10.52 degrees, **which** is within the 15-degree tolerance, so I am roughly facing it. The current **view** shows a **clear path** to the waypoint. My last action was stepping forward, so I should continue moving towards the waypoint. The distance to the waypoint is 1347.75 cm. Stepping forward **for** 4.0 seconds at 200 cm/s will cover 800 cm, bringing me closer to the waypoint without exceeding the maximum step duration of 5 seconds.

852

853

854

Figure 17: Example Chain of Thought from Gemini-2.5-Pro on a Successful Easy Level Task.

C Case Study 2 Details

C.1 Problem Formulation

In Case Study 2, we formalize the scenario as a sequential decision-making problem within a multi-agent environment. Specifically, we focus on evaluating the decision-making capabilities of *Delivery Agents*. *Customers* and *Stores* are managed by a unified rule-based manager, while each *Delivery Agent* autonomously decides actions to maximize profit.

C.1.1 Agent Definition

Let the set of *Delivery Agent* be represented as $\mathcal{D} = \{D_1, D_2, \dots, D_n\}$, each operating independently yet concurrently in the shared environment.

C.1.2 Decision Variables

At each discrete timestep t , an agent D_i chooses from the following set of actions:

1. Accept or reject available orders.
2. Plan routing along waypoints defined by a provided map structure.
3. Purchase transportation equipment to enhance delivery efficiency, subject to available financial resources.
4. Adjust travel speed $v_t \in [v_{\min}, v_{\max}]$ to balance energy consumption and delivery speed.

C.1.3 Constraints

Each *Delivery Agent* D_i operates under the following constraints:

- **Energy Constraint:** Agent D_i has an energy level $E_t \geq 0$. Movement at higher speeds consumes more energy, and energy can be replenished by purchasing beverages at supply points.
- **Financial Constraint:** Each agent starts with an initial amount of money M_0 . Actions such as purchasing transportation equipment or beverages require spending money. The agent's financial status at time t is represented by $M_t \geq 0$.
- **Spatial Constraint:** Movements must follow a predetermined waypoint map $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where \mathcal{V} is the set of waypoints, and \mathcal{E} represents feasible paths.

C.1.4 Objective Function

The primary objective for each *Delivery Agent* D_i is to maximize total money M_T accumulated by the end of simulation period T :

$$\max \quad M_T = M_0 + \sum_{t=1}^T (R_t - C_t) \quad (1)$$

where:

- R_t represents revenue earned at timestep t from successful deliveries.
- C_t represents costs incurred at timestep t , including equipment purchase costs, energy replenishment costs, and penalties from delayed or failed deliveries.

C.1.5 Agent Dynamics

Agent dynamics at timestep t are governed by:

$$E_{t+1} = E_t - f(v_t) + \Delta E_{\text{replenish}}, \quad (2)$$

$$M_{t+1} = M_t + R_t - C_t, \quad (3)$$

$$X_{t+1} = \text{WaypointUpdate}(X_t, a_t), \quad (4)$$

where:

- $f(v_t)$ denotes energy consumption as a function of travel speed.

- $\Delta E_{\text{replenish}}$ is the energy replenished through beverages at supply points.
- X_t denotes the agent's position at timestep t , updated based on actions a_t following the waypoint system.

This formulation offers a comprehensive framework for evaluating long-horizon decision-making strategies of *Delivery Agents* within the multi-agent environment of SIMWORLD.

C.1.6 Metrics Formulation

Aligned with our hierarchical evaluation framework, we define the following metrics for each agent D_i operating over T timesteps:

High-level: Total Profit

$$P_i = M_i(T) - M_i(0),$$

where $M_i(t)$ is the agent's monetary balance at time t .

Mid-level: Operational Effectiveness

$$\text{Order Success Rate: } SR_i = \frac{|\mathcal{O}_i^{\text{succ}}|}{|\mathcal{O}_i^{\text{bid}}|}, \quad (5)$$

$$\text{Delay Rate: } DR_i = \frac{|\mathcal{O}_i^{\text{delay}}|}{|\mathcal{O}_i^{\text{succ}}|}, \quad (6)$$

$$\text{Energy Efficiency: } EE_i = \frac{P_i}{\sum_{t=1}^T e_i(t)}, \quad (7)$$

$$\text{Order-Sharing Number: } SH_i = |\mathcal{O}_i^{\text{shared_succ}}|, \quad (8)$$

$$\text{Investment Number: } IN_i = |\mathcal{A}_i^{\text{buy_bike_succ}}|, \quad (9)$$

where

- $\mathcal{O}_i^{\text{bid}}$ is the set of orders on which D_i bid,
- $\mathcal{O}_i^{\text{succ}} \subseteq \mathcal{O}_i^{\text{bid}}$ are those bids that led to successful deliveries,
- $e_i(t)$ is energy consumed by D_i at timestep t ,
- $\mathcal{O}_i^{\text{delay}} \subseteq \mathcal{O}_i^{\text{succ}}$ are deliveries completed after the deadline,
- $\mathcal{O}_i^{\text{shared_succ}}$ denotes successfully shared orders.
- $\mathcal{A}_i^{\text{buy_bike_succ}}$ demotes successfully buy bike actions.

Low-level: Instruction and Rule Compliance

$$IDR_i = \frac{I_i}{D_i}, \quad IDR_{i,a} = \frac{I_{i,a}}{D_{i,a}} \quad \forall a \in \{\text{bid, pickup, deliver, beverage, shared, meet, cancel, speed, nothing}\},$$

where

- D_i is the total number of actions taken by D_i ,
- I_i is the total number of those actions deemed invalid,
- $D_{i,a}$ and $I_{i,a}$ are the counts of attempts and invalid attempts for action a .

For each metric $m \in \{P, SR, ADS, DR, EE, SH, IDR, IDR_a\}$, we report group-level aggregates:

$$\bar{m} = \frac{1}{n} \sum_{i=1}^n m_i, \quad \text{median}(m), \quad \sigma(m) = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (m_i - \bar{m})^2}.$$

These statistics are written to our CSV outputs for each model grouping. And we show the part of metric in the Table 3

914 C.2 Action Space

915 The low-level action space comprises fine-grained executable commands in Unreal Engine, such as
 916 move forward, rotate, pick up, and drop off. In contrast, high-level actions are determined by the LLM.

Table 7: **Hierarchical Action Space Design in Case Study 2.** High-level actions are given to language models to make decision, which correspond to strategic decisions, while low-level actions are only exposed to local action planner module to execute concrete movements and interactions.

Action Level	Action Name	Description	Invocation Method
High-Level Actions	Bid Order	Offer a price to a new order on platform to compete with other Model Generations	Model Generation
	Pick Up Order	Navigate to the pick-up point of order	Model Generation
	Deliver Order	Navigate to the delivery point and complete the order	Model Generation
	Share Order	Publish the order for multi-model generation cooperation	Model Generation
	Cancel Share Order	Cancel a shared order that has been published	Model Generation
	Go to Meet-point	Navigate to the meet point for the shared order	Model Generation
	Purchase Scooter	Buy and use a scooter	Model Generation
	Purchase Drinks	Buy consumables to restore energy	Model Generation
	Adjust Speed	Adjust travel speed	Model Generation
Low-Level Actions	Move Forward	Basic movement action	Local Planner
	Stop	Stop moving	Local Planner
	Rotate	Adjust the facing direction	Local Planner
	Change Speed	Adjust walking speed	Local Planner
	Drive Scooter	Control a scooter for movement	Local Planner

917

918 C.3 Prompt Design

System Prompt of Long-Horizon Delivery Task

```

SYSTEM_PROMPT = """
You face the following considerations:
1. Accepting orders along your route can increase earnings efficiency by
   reducing empty travel time and grouping deliveries.
2. Risks include unpredictable road conditions, inaccurate time estimates, and
   potential delays leading to penalties and lower ratings.
3. Late deliveries incur negative ratings and monetary penalties, affecting your
   overall performance.
4. For each order, evaluate its fit to your current route, delivery time promise
   , and balance efficiency with service quality.
5. You can buy a bike for <PRICE_OF_BIKE> dollars to increase speed and reduce
   energy consumption.
6. You can change your speed (<DELIVERY_MAN_MIN_SPEED> to <
   DELIVERY_MAN_MAX_SPEED> cm/s); higher speeds consume more energy.
7. You may share a picked-up order with another delivery man by specifying a
   meeting point; both delivery men share earnings and penalties if late. This
   can bring both of you more money and save your energy.
8. You can cancel a shared order if no one accepts it after a long wait. That
   means you decide to do it by yourself, rather than waiting for too long
   which may lead to a penalty.
9. If low on energy, go to a supply point to buy a beverage (cost <
   COST_OF_BEVERAGE> dollars, recovers <DELIVERY_MAN_RECOVER_ENERGY_AMOUNT>
   energy).
10. You need to finish order in time to make money, do not bid a order and not
    delivery it.

Important:
- All your decisions are based on your persona and the considerations.
- Provide only the final action in JSON, the JSON format would be given in each
  round of request. Do not add explanations or repeat given examples.
- There are other delivery men in the city, try to compete with them or
  cooperate with them according to the your persona and the situation.
- If you can not make the most money at the end of game, you will die. So try
  your best to use the rules to make the most money.
"""

```

919

Figure 18: Examples System Prompt of Long-horizon Delivery Task.

920

921

User Prompt of Long-Horizon Delivery Task

```
# User prompt to output the reasoning
REASONING_USER_PROMPT = """
You are now at <POSITION> in a city, where the unit is cm.
You can buy and use a beverage to recover your energy.
Your position is always at or near a node in the graph and you can move from one
node to another node only if there is an edge between the two nodes. All
your possible next waypoints are:
<POSSIBLE_NEXT_WAYPOINTS>
You have already picked up the following order in the format of Order:
<PICKED_UP_ORDER>
You already have the following orders in the format of List[Order]:
<ORDERS>
Also, you can bid for new orders from the platform in the format of List[Order]:
<ORDERS_TO_BID>
And you could notice the following notification, which is the platform wanna you
to know:
<NOTIFICATION>
For each order, you should check the attributes before making a decision:
- max_sale_price: the maximum price of the order.
- min_sale_price: the minimum price of the order.
- customer_position: the position of the customer.
- store_position: the position of the store.
- has_picked_up: whether the order has been picked up by you.
- estimated_time: the estimated time to deliver the order.
- is_shared: whether the order is shared.
- meeting_point: the meeting point of the shared order.
- spent_time: the time that has passed since the order was opened.
Note that you consume energy as long as you move. There are two ways to move:
walking and driving. You lose energy as long as you move. Walking is your
default way to move and you can choose to buy a bike to drive if you have
enough money. Your current walking speed is <SPEED> cm/s. Faster walking
speed will consume more energy. Now your moving way is <PHYSICAL_STATE>.
Currently, you have <MONEY> dollars and <ENERGY> energy.

You have the following history of your actions:
<HISTORY>
The first item of each tuple in the history is your position, and the second
item is your action.
From top to bottom, the history is sorted by time. The most recent action is at
the bottom.
To improve efficiency, avoid visiting the same position more than once unless
necessary and avoid repeating the same action except Deliver a order.

Now you need to make a decision. You have the following options:

1. Bid for an order. You can bid for an order from the platform. You can only
bid for an order on the platform. The bid price should be between the
min_sale_price and max_sale_price of the order.
2. Pick up an order. You can pick up an order from your orders. You can only
pick up an order that has not belonged to you.
3. Deliver an order. You can deliver an order from your orders. Do not deliver
the order that has not been picked up by you.
4. Buy a beverage. When your energy is low, you can buy a beverage to recover
your energy. The price of beverage start from <COST_OF_BEVERAGE> dollars
and will go up 5% every time you buy a beverage. A beverage recovers <
RECOVER_ENERGY_AMOUNT> energy.
5. Open a shared order. You can open a shared order from your orders so another
delivery man can join the shared order. This can save your time and energy,
but you need to split the earnings with another delivery man. You can only
share an order that has been picked up by you. You need to decide the
meeting point and go there to wait for another delivery man. They will
finish the rest of the order. Note that if the shared order is delivered
late, both of you will receive negative customer ratings and penalties.
6. Go to the meeting point. If you are joining a shared order, you need to go to
the meeting point and wait for another delivery man.
7. Cancel a shared order. If a shared order has not been accepted by another
delivery man for a long time, you can cancel it and do it by yourself.
8. Change walking speed. You can change your walking speed to a new walking
speed (range from 100 to 250 cm/s).
9. Buy a bike. You can buy a bike to increase your speed. A bike costs <
PRICE_OF_BIKE> dollars. But it will increase your speed and decrease your
energy consumption.

Make a reasoning about your next action based on the above information. For each
action, you have the following tips:
```

```

Explain your reasoning using one sentence with clear instructions in json format
.

Example reasoning (do not repeat):

{'reasoning': 'I have enough money to buy a bike, so I will buy a bike now.'}

Now, provide only the action JSON with keys: reasoning.
"""

# User prompt to output the action
ACTION_USER_PROMPT = """
Your reasoning: <REASONING>

Now you need to respond your action in json format. Note that you can only
choose one of the following actions:

1. Bid for an order. Respond with the index of the order you want to bid for and
the bid price.
2. Pick up an order. Respond with the index of the order you want to pick up,
the next waypoint and the target point.
3. Deliver an order. Respond with the index of the order you want to deliver,
the next waypoint and the target point.
4. Buy a beverage. Just respond with your choice.
5. Open a shared order. Respond with the index of the order you want to share
and the meeting point.
6. Go to the meeting point. Respond with the index of the shared order you want
to join and the next waypoint.
7. Cancel a shared order. Respond with the index of the shared order you want to
cancel.
8. Change walking speed. Respond with the new walking speed.
9. Buy a bike. Just respond with your choice.

Provide only the action JSON with keys: choice, index, target_point,
meeting_point, next_waypoint, new_speed.

Example action (do not repeat):

{'choice': 1, 'index': 1, 'target_point': (200, 200), 'meeting_point': (300,
300), 'next_waypoint': (400, 400), 'new_speed': 200}
"""

```

Figure 19: Examples User Prompt of Long-horizon Delivery Task.

C.4 Dataset Details

C.4.1 Simulated Environment Generation

Maps are created via a multi-stage procedural pipeline (used in both variants):

1. **Road-Grid Skeleton:** define an $N \times N$ road network ($N \in \{3, \dots, 8\}$), centered at $(0, 0)$, with uniform segment length and spacing.
2. **Element Population:** place buildings programmatically along each road; add trees and street lights for visual realism.
3. **Data Screening:** export all scene elements, then remove non-restaurant nodes (e.g., hospitals, schools) to prepare for real-world alignment.

C.4.2 Real-World Dataset (LLMDeliveryReal-100)

This variant aligns simulated blocks with actual restaurant data:

1. **Data Acquisition:** fetch Manhattan restaurant listings (latitudes 40.758–40.770, longitudes –73.995– –73.970) via Google Maps API; clean and convert to geospatial points.
2. **Neighborhood Assignment:** build polygonal blocks from street intersections; assign each restaurant via point-in-polygon tests.

941 3. **Normalization:** map real streets/avenues to grid indices $0, \dots, N - 1$. For each restaurant,
 942 compute relative position $(\alpha, \beta) \in [0, 1]^2$ within its block and obtain

$$(x_{\text{norm}}, y_{\text{norm}}) = (x_0 + \alpha, y_0 + \beta).$$

943 4. **Building Coordinate Scaling:** convert original building coordinates $(x_{\text{orig}}, y_{\text{orig}}) \in$
 944 $[-\text{offset}, \text{offset}]^2$ to $[0, N - 1]^2$ via

$$x_{\text{norm}} = \frac{x_{\text{orig}} + \text{offset}}{2\text{offset}} (N - 1), \quad y_{\text{norm}} = \frac{y_{\text{orig}} + \text{offset}}{2\text{offset}} (N - 1).$$

945 5. **Matching:** compute pairwise Euclidean distances in normalized space; match each building to its
 946 nearest restaurant within threshold 0.25, resolving conflicts by closest-pair selection.

947 6. **Metadata Injection:** embed a match flag and restaurant metadata (ID, name, category, geo-coords)
 948 into each building node.

949 C.4.3 Procedurally-Generated Dataset (LLMDeliveryRandom-1K)

950 This variant omits real-world alignment and instead samples agent and store positions randomly:

- 951 • **Generation Type:** “random procedural generation.”
- 952 • **Agent Placement:** DeliveryManager instantiates fixed counts of stores, customers and delivery
 953 agents at uniformly random building locations.

954 C.4.4 File Structure

955 Each map folder contains:

- 956 • `metadata.json`: difficulty, entity counts, generation type, file paths.
- 957 • `positions.json`: 2D coordinates of stores, customers, delivery agents.
- 958 • `progen_world.json`: full scene graph with normalized coords and match flags.
- 959 • `roads.json`: road segment list with endpoints.
- 960 • `city_visualization.png`: full-view rendering.
- 961 • `dm_visualization.png`: delivery-focused schematic.

962 C.5 Results

963 C.5.1 Main Results

964 DeepSeek-V3 (69.475 ± 16.772) and Claude-3.5-Sonnet (69.068 ± 20.685) achieved the highest
 965 mean profits, with Claude-3.5-Sonnet also leading in mean successful orders (2.733 ± 1.102) and
 966 energy efficiency (0.5411 ± 0.1981). Notably, these superior average outcomes were associated with
 967 substantial performance variability, as reflected in their respective standard deviations.

968 Conversely, Gemini-2.5-Flash, while attaining a moderate mean profit of 42.423, exhibited
 969 markedly more consistent profit generation with a standard deviation of ± 3.103 , and also demon-
 970 strated stability in successful orders (2.100 ± 0.173). Extreme variability was evident in specific
 971 metrics for certain models; for instance, sharing counts for Deepseek-Prover-V2 (7.333 ± 8.386)
 972 and Claude-3.5-Sonnet (11.333 ± 8.386) showed standard deviations exceeding their means,
 973 indicating highly unpredictable behavior in this aspect.

974 The GPT-4o-mini model consistently yielded zero values across all metrics (0.000 ± 0.000), suggest-
 975 ing it does not truly understand the goals to make reasonable decisions based on the given instructions
 976 and context information.

977 Higher investment strategies, such as those adopted by DeepSeek-V3 (8.000 ± 3.000) and
 978 Claude-3.5-Sonnet (9.000 ± 3.464), generally correlated with greater mean profit achievement
 979 but also with increased outcome volatility. These findings underscore a prevalent trade-off between
 980 optimizing for peak average performance metrics and ensuring consistent, predictable agent behavior,
 981 a critical consideration for robust deployment in dynamic environments.

982 C.5.2 Environment Configuration

983 We further investigate how different environmental configurations impact agent behavior and overall
 984 performance. Specifically, we explore two key factors: the global order availability and the agents’

985 initial financial endowment. For each factor, we conduct a series of controlled experiments to observe
 986 how variations affect agents' action distributions.

987 As shown in Figure 20a, when the total number of available orders increases, agents tend to perform
 988 fewer pickup and delivery actions and instead choose the `do_nothing` action more frequently. This
 989 suggests that in resource-rich environments, agents are more inclined to conserve energy and avoid
 990 unnecessary effort, opting to wait for optimal opportunities rather than actively pursue deliveries.
 991 Conversely, in low-resource settings, agents are more motivated to engage in delivery tasks to secure
 992 profits. Additionally, as resource abundance increases, agents demonstrate a higher tendency to
 993 initiate and complete shared deliveries, likely as a means to reduce energy costs through collaboration.

994 Figure 20b illustrates the impact of agents' initial monetary resources. As initial capital increases,
 995 agents are less reliant on aggressive bidding and instead prioritize actions such as order pickup. When
 996 funds are limited, competition intensifies, leading to more frequent bidding behavior. Furthermore,
 997 with sufficient initial capital, agents are more willing to invest in infrastructure—such as purchasing a
 998 bike—which enhances their long-term delivery efficiency.

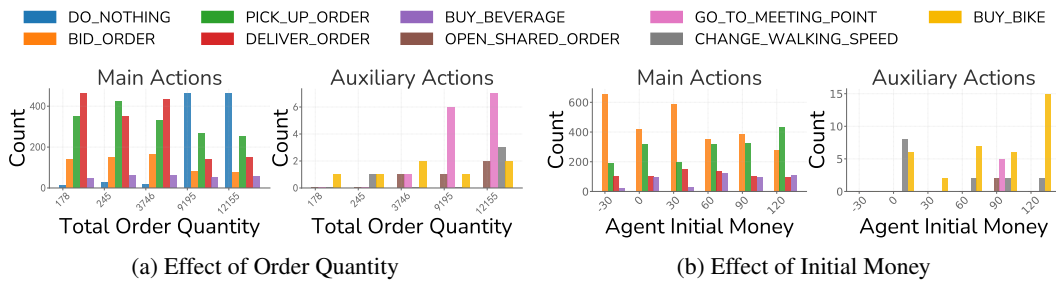


Figure 20: **Action Distribution across Environmental Settings.** (a) Shows the change of effect of global order quantity on agent behavior. (b) Shows the effect of initial money on action selection.

999 C.5.3 Influence of Persona.

1000 Personality traits significantly affect the decision-making and performance of delivery agents. As
 1001 shown in Figure 21, agents with higher Conscientiousness tend to exhibit a lower frequency of
 1002 bidding actions, a higher frequency of task-completion actions (e.g., picking up orders), and achieve
 1003 a higher bid win rate. This suggests that conscientious agents prioritize task completion over
 1004 strategic competition. Agents with higher Agreeableness are less likely to remain inactive (i.e.,
 1005 perform "do nothing" actions) and tend to achieve higher bid win rates. Conversely, agents with
 1006 lower agreeableness display higher inactivity and narrower bidding price ranges, limiting their
 1007 competitiveness. Interestingly, agents with higher Openness exhibit reduced engagement in delivery
 1008 tasks, possibly because they explore competitive or unconventional bidding strategies that divert
 1009 attention from task execution.

1010 C.6 Chain of Thought Example

1011 This section, we showcase a representative chain of thought from Claude-3.5-Sonnet. Which
 1012 shows the model's reasoning process under different circumstances.

Example Chain of Thought from Claude-3.5-Sonnet on different actions

```

Action: Do Nothing
Reasoning:
Case1= """
Since I have no orders to deliver and no new orders to bid on, I'll wait at my
current position to conserve energy and wait for new orders to appear.
"""

Case2= """
I have no orders to deliver and no money to buy anything, so I should wait for
new orders from the platform.
"""

```

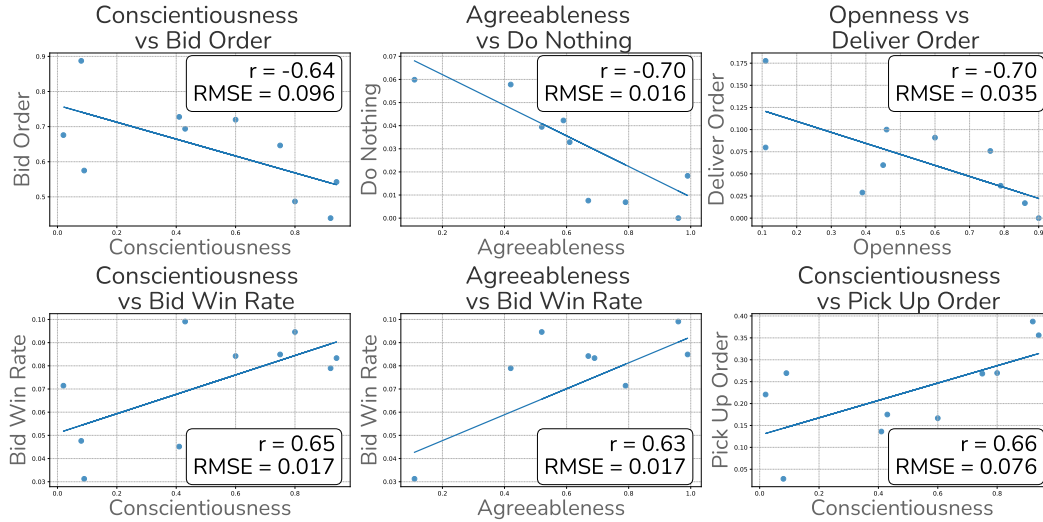


Figure 21: **Linear Regression Reveals Strong Correlations between Behaviors and Persona Traits.** The results demonstrate that agent behaviors are tightly aligned with their corresponding persona attributes, highlighting the effectiveness of trait-driven behavior modeling.

```

Action: Bidding
Reasoning:
Case1=""Since I don't have any money yet and there are good orders near me, I
will bid 142.0 for Order(customer_position=Vector(x=-18300.0, y=-1700.0),
store_position=Vector(x=-21700.0, y=1700.0)) since it's close to my current
position and has a good profit margin.
"" ""

Case2=""
Since I have no money and no active orders, I should bid on Order(max_sale_price
=146.5775036765818, min_sale_price=114.8202263455568) which starts near my
location at Vector(x=38300.0, y=-1700.0) to maximize efficiency.
"" ""

Case3=""
Since I have no orders and no money, I'll bid for Order 2 which delivers between
-58300,-1700 and -58300,1700 since it's closest to my current location and
has a short estimated time of 68 seconds.
"" ""

Case4=""
As I have 0 dollars and see other delivery men bidding 130-140 for the first
order, I should bid 143 dollars for Order 9 which has high max price and is
close to store position to start earning money.
"" ""

Case5=""
Since I already bid an order but lost to competitor who bid 130, I need to bid a
higher price of 135 for Order(customer_position=Vector(x=-18300.0, y
=-1700.0), store_position=Vector(x=1700.0, y=-21700.0)) to increase my
chances of winning.
"" ""

Action: Change Speed
Reasoning:
Case1=""
Since I have no orders to deliver and there are no new orders to bid on, I
should change my walking speed to 200 cm/s to conserve energy while still
maintaining reasonable mobility.
"" ""

Case2=""
I have an order to pick up at Vector(x=-21700.0, y=1700.0), so I need to move
through Vector(x=21700.0, y=1700.0) to get there, but I've been moving back

```

```

        and forth inefficiently, so I should change my walking speed to 200 cm/s
        to conserve energy while maintaining progress.
    """
    Action: Pick Up Order
    Reasoning:
    Case1=""
    Since I already bid for two orders and have high competition, I should move
    towards Vector(x=18300.0, y=1700.0) to get closer to my assigned order's
    store location at Vector(x=38300.0, y=-1700.0).
    """

    Case2=""
    I need to go to Vector(x=-18300.0, y=-1700.0) to compete with other delivery men
    for orders near there since there are multiple orders in that area.
    """

    Case3=""
    I am at the edge of the map and low on money, so I will move towards Vector(x
    =38300.0, y=1700.0) to be closer to the center where orders are
    concentrated.
    """

    Case4=""
    I need to move towards Vector(x=38300.0, y=1700.0) to pick up my order at the
    store since I have already won the bid for the delivery to Vector(x
    =21700.0, y=-21700.0)
    """

    Case5=""
    I have two orders to pick up and I'm closer to the first store at (-58300, 1700)
    , so I'll move towards Vector(x=-41700.0, y=1700.0) to pick up the first order.
    """

    Action: Deliver Order
    Reasoning:
    Case1=""
    I will go to Vector(x=38300.0, y=1700.0) since it's on the path to the store of
    my current order at Vector(x=-21700.0, y=1700.0).
    """

    Case2=""
    I need to deliver order at Vector(x=-58300.0, y=-1700.0) since I have already
    picked it up and I'm at the store position, so I will move to Vector(x
    =-58300.0, y=-1700.0).
    """

    Case3=""
    I have an order to deliver to Vector(x=-1700.0, y=-1700.0) and I'm at the store
    location, so I'll move to Vector(x=21700.0, y=-1700.0) which is on the
    shortest path to the customer.
    """

    Case4=""
    I should move to Vector(x=-18300.0, y=-1700.0) to deliver the order since it's
    closest and will let me earn money quickly.
    """

    Action: Buy Beverage
    Reasoning:
    Case1=""
    I am at the edge of the map with low energy and multiple nearby orders, so I
    should buy a beverage to recover energy before taking on more deliveries.
    """

    Case2=""
    My energy is getting low at 80 and I have a long delivery ahead, so I should buy
    a beverage to recover energy before heading to the store location.
    """

    Case3=""
    I have a picked up order but my energy is getting low, so I should buy a
    beverage first to ensure I can complete the delivery safely.
    """

    Action: Buy Bike
    Reasoning:
    Case1=""

```

```

I have several orders at my current location but need to pick them up from
distant stores, so I should buy a bike first to increase efficiency since I
have enough money
"""

Action: Shared Order
Reasoning:
Case1="""
DeliveryMan 1:I need to deliver my current orders but have low energy, so I
should share an order to reduce my workload and conserve energy.

DeliveryMan 4: Since DeliveryMan 1 has opened a shared order with a meeting
point at Vector(x=18300.0, y=-1700.0), I should move to Vector(x=18300.0, y
=-1700.0) to accept and join the shared delivery.

DeliveryMan 1: Since I am at the meeting point Vector(x=1700.0, y=-1700.0) and
DeliveryMan 4 wants to join my shared order, I should wait here to meet him
.

DeliveryMan 4: Since I am at the meeting point and have low energy but not
enough money for beverage, I should wait here to get my share from the
shared orders to earn money first.

DeliveryMan 1: I don't have enough money or energy to take new orders, and my
first shared order has been waiting too long, so I should cancel the shared
order Order(max_sale_price=142.4723211683914, min_sale_price
=114.17753296249396, customer_position=Vector(x=-18300.0, y=-1700.0),
store_position=Vector(x=1700.0, y=-21700.0)) and do it myself.
"""

```

1016

1017

1018

Figure 22: Example Chain of Thought from Claude-3.5-Sonnet on a Delivery Task.

1019 D Related Work

1020 **Minecraft and Web-based Agents** Recent advancements have led to the development of intelligent
1021 agents utilizing open-ended platforms such as Minecraft and the Web. Minecraft offers a procedurally
1022 generated, sandbox environment that supports long-horizon planning, multi-agent collaboration,
1023 and complex tool use. A series of benchmark efforts have emerged in this setting: Voyager [42]
1024 enables lifelong learning and autonomous skill acquisition through LLM-powered exploration; Team-
1025 Craft [27] evaluates collaborative agent behaviors grounded in multimodal perception; Odyssey [24]
1026 incorporates a comprehensive skill library with a fine-tuned LLaMA-3 model to enhance agent capa-
1027 bilities; and Optimus-2 [21] demonstrates hierarchical control using MLLMs for complex mission
1028 execution.

1029 Parallel research has extended to web-based agents, where the environment entails realistic interaction
1030 patterns, long-term planning, and diverse user-facing tasks. WebVoyager [9], for example, builds
1031 an end-to-end system using multimodal models to support browsing, form-filling, and question
1032 answering. Complementary frameworks such as BrowserGym [7], GPT-4V Web Agent [54], Skill-
1033 Weaver [55], and Real-World WebAgent [14] further push the boundary by enabling self-improvement,
1034 long-context reasoning, and programmatic action synthesis.

1035 Building on these foundational efforts, we introduce SIMWORLD and the Delivery Benchmark to
1036 explore a new frontier in agent research: LLM-driven multi-agent behavior in large-scale, dynamic,
1037 urban environments. Unlike prior setups, our benchmark places agents in a photorealistic, procedu-
1038 rally generated city and tasks them with fulfilling complex delivery jobs over extended time horizons.
1039 To meet the demands of realism and open-ended interaction, we incorporate a hierarchical control
1040 framework, persistent memory modules, in-context reasoning feedback, and personality-driven be-
1041 havioral diversity. Together, these components enable immersive and coordinated agent performance,
1042 offering a robust testbed for studying language-conditioned reasoning, cooperation, and adaptation in
1043 real-world inspired settings.

1044 **Physical and Social Simulator for AI Agents** Simulations have played a crucial role in con-
1045 structing environments for training and evaluating autonomous agents. Text-based simulators often
1046 emphasize social scenarios, such as human interaction [48], daily activities [31], and relational
1047 polarization [33]. Popular embodied simulators support a broader range of applications, particularly
1048 in embodied AI research and 2D/3D scene synthesis [23]. However, most embodied simulators
1049 remain constrained to either indoor household environments (e.g., AI2-THOR [18], Habitat [34],
1050 iGibson [19]) or outdoor driving scenarios (e.g., CARLA [10], MetaDrive [22]) or natural scenes
1051 (e.g., AirSim [38]). Most of these simulators [10, 34, 19, 38, 43, 13] often rely on a limited num-
1052 ber of manually crafted scenes, which hinders scalability and diversity. Some platforms, such as
1053 MetaUrban [46], MetaDrive [22] and AI2-THOR [18], introduce rule-based procedural generation to
1054 alleviate this issue. Nonetheless, existing embodied simulators typically lack support for dynamic
1055 multi-agent interactions in complex, outdoor environments.

1056 Recent advancements have introduced large-scale, language-driven social simulators capable of
1057 modeling complex societal behaviors. OASIS [49] simulates up to one million LLM-powered agents
1058 interacting on social media platforms, capturing phenomena such as information diffusion, echo
1059 chambers, and polarization. Casevo [17] integrates chain-of-thought reasoning, retrieval-augmented
1060 generation, and customizable memory mechanisms to simulate intricate social phenomena and
1061 decision-making processes. MineLand [51] offers a multi-agent Minecraft environment where agents,
1062 driven by physiological needs and limited multimodal perception, engage in collective behaviors,
1063 fostering ecological and detailed simulations. Project SID [2] further advances this landscape by
1064 deploying over 1,000 autonomous AI agents within a Minecraft environment to explore the emergence
1065 of AI civilizations. Utilizing the PIANO (Parallel Information Aggregation via Neural Orchestration)
1066 architecture, agents interact in real-time, developing specialized roles, adhering to and modifying
1067 collective rules, and engaging in cultural and religious transmission. These simulations demonstrate
1068 agents’ capabilities in forming complex social structures, economies, and governance systems,
1069 providing insights into large-scale societal simulations and agentic organizational intelligence.

1070 In summary, while prior simulators have demonstrated success in specific domains such as indoor
1071 navigation or autonomous driving, none have been explicitly designed to support dynamic, multi-
1072 agent interactions in outdoor, city-scale environments. SIMWORLD fills this gap by offering a scalable

1073 platform that enables multi-agent collaboration and competition, language-grounded interactions, and
1074 procedural generation of urban scenes—all essential for benchmarking advanced embodied agents.

1075 **Benchmarks for Foundation Model Agents** With the rise of foundation models, a wide range
1076 of agent frameworks have been developed that leverage large language models (LLMs)[5, 57] or
1077 multimodal large language models (MLLMs)[47, 26, 30] to perform complex tasks. Numerous
1078 benchmarks have emerged to evaluate LLM-based agents’ capabilities across domains such as
1079 decision-making[25], tool use[35], scientific reasoning[45], and software development[45, 16]. For
1080 MLLM-based agents, common benchmarks often involve embodied tasks such as vision-language
1081 navigation (VLN)[40], robotic manipulation[53, 28, 56], and interactive household activities[20, 4].
1082 And some benchmarks focus on playing games, especially Minecraft[51]. However, most of these
1083 benchmarks present relatively short-horizon or goal-directed tasks with limited action spaces—often
1084 consisting of just a few discrete options. Agents are typically instructed to win a game[5] or complete
1085 a predefined task[47, 26, 30], which oversimplifies real-world reasoning and coordination challenges.
1086 Moreover, most settings ignore the complexity introduced by multiple agents operating in the same
1087 environment[47, 26, 25, 35, 6], where cooperation, competition, negotiation, and resource contention
1088 are vital aspects of intelligent behavior.

1089 In contrast to benchmarks that focus on short-horizon, single-agent tasks with limited action spaces,
1090 our proposed Delivery Benchmark introduces a dynamic, multi-agent urban simulation where agents
1091 must compete, cooperate, and plan over extended time horizons. The open nature of the environment
1092 and the need for real-time resource management provide a more realistic and rigorous test bed to
1093 evaluate reasoning, coordination, and adaptation in agents based on LLM and MLLM.

1094 **E Conclusion**

1095 In this work, we introduce SIMWORLD, a next-generation simulator designed to bridge the gap
1096 between high-level cognitive reasoning and low-level embodied execution for LLM/VLM-powered
1097 agents. SIMWORLD uniquely combines realistic physical dynamics, socially grounded interactions,
1098 and language-controllable procedural generation to create open-ended, urban-scale environments.
1099 Through two comprehensive case studies—urban physical navigation and multi-agent social delivery
1100 economy—we demonstrate the platform’s capability to support complex physical and social reasoning,
1101 long-horizon planning, and strategic interaction under diverse environmental conditions.

1102 Our evaluations reveal that while frontier LLMs exhibit promising planning and interaction abilities,
1103 they also manifest limitations in perceptual grounding, rule-following, and consistent decision-making.
1104 Importantly, SIMWORLD enables a fine-grained analysis of agent behavior under varying resource
1105 conditions, competitive pressures, and personality traits, offering new insights into the nature of agent
1106 adaptability and emergent strategies.

1107 We believe SIMWORLD lays the foundation for a more holistic and rigorous development of embodied
1108 intelligence. By supporting scalable simulation, rich agent interfaces, and systematic evaluation, it
1109 opens up new research opportunities across AI, robotics, social simulation, and multi-agent systems.
1110 We release SIMWORLD as an open platform and invite the community to use, extend, and build upon
1111 it toward the goal of general-purpose intelligent agents capable of thriving in real-world physical and
1112 social environments.

1113 F Broader Impacts

1114 SimWorld aims to establish a new paradigm for simulating realistic, interactive, and open-ended
1115 environments that support large language model (LLM)-driven agents. Its design bridges the gap
1116 between traditional simulators and the growing need for multimodal, embodied reasoning and
1117 decision-making capabilities. The broader impacts of SimWorld span multiple domains:

1118 **Advancing AI Safety and Alignment.** SimWorld offers a controlled yet complex testbed for
1119 studying agent behavior, ethics, and alignment in real-world inspired scenarios. By supporting
1120 language-based control and social decision-making, it allows researchers to systematically evaluate
1121 how agents interpret ambiguous instructions, negotiate trade-offs, and adhere to human values — key
1122 to advancing safe and aligned AI systems.

1123 **Democratizing Embodied AI Research.** Built on top of Unreal Engine and designed with modular
1124 extensibility, SimWorld lowers the entry barrier for researchers and educators working on embodied
1125 agents, human-AI interaction, and reinforcement learning. It enables experimentation with low-cost
1126 models while supporting high-fidelity simulation, which makes embodied AI research more accessible
1127 and inclusive.

1128 **Benchmarking and Transparency.** SimWorld includes standardized multi-agent tasks and bench-
1129 mark protocols, allowing fair comparison across models and strategies. This promotes reproducibility
1130 and transparency in a field where rapid progress has often outpaced rigorous evaluation.

1131 **Real-World Applicability.** By simulating city-scale logistics, physical interaction, and social
1132 coordination tasks, SimWorld directly contributes to domains such as autonomous driving, ser-
1133 vice robotics, and digital urban planning. These applications can have significant societal impact,
1134 improving safety, accessibility, and efficiency in public infrastructure.

1135 **Risks and Mitigations.** As with any simulation platform, there is a risk of overfitting models to
1136 simplified environments. We mitigate this by supporting real-world data integration and promoting
1137 diverse task settings. Moreover, the misuse of SimWorld in developing deceptive or manipulative
1138 agents is a concern; we encourage the community to adopt and share responsible usage practices, and
1139 we plan to release only non-malicious agent baselines by default.

1140 In summary, SimWorld provides a generalizable, ethical, and forward-looking testbed that supports
1141 the next generation of interactive AI systems. Its broader impact lies in enabling the research
1142 community to move beyond static benchmarks toward dynamic, real-world-relevant evaluation and
1143 development.

1144 **NeurIPS Paper Checklist**

1145 **1. Claims**

1146 Question: Do the main claims made in the abstract and introduction accurately reflect the
1147 paper's contributions and scope?

1148 Answer: [\[Yes\]](#)

1149 Justification: The abstract describes the key features of our work and include the reason why
1150 our work is distinguish with other simulators, which is the key point of this paper.

1151 Guidelines:

- 1152 • The answer NA means that the abstract and introduction do not include the claims
1153 made in the paper.
- 1154 • The abstract and/or introduction should clearly state the claims made, including the
1155 contributions made in the paper and important assumptions and limitations. A No or
1156 NA answer to this question will not be perceived well by the reviewers.
- 1157 • The claims made should match theoretical and experimental results, and reflect how
1158 much the results can be expected to generalize to other settings.
- 1159 • It is fine to include aspirational goals as motivation as long as it is clear that these goals
1160 are not attained by the paper.

1161 **2. Limitations**

1162 Question: Does the paper discuss the limitations of the work performed by the authors?

1163 Answer: [\[Yes\]](#)

1164 Justification: We discussed limitations of the work on Appendix F.

1165 Guidelines:

- 1166 • The answer NA means that the paper has no limitation while the answer No means that
1167 the paper has limitations, but those are not discussed in the paper.
- 1168 • The authors are encouraged to create a separate "Limitations" section in their paper.
- 1169 • The paper should point out any strong assumptions and how robust the results are to
1170 violations of these assumptions (e.g., independence assumptions, noiseless settings,
1171 model well-specification, asymptotic approximations only holding locally). The authors
1172 should reflect on how these assumptions might be violated in practice and what the
1173 implications would be.
- 1174 • The authors should reflect on the scope of the claims made, e.g., if the approach was
1175 only tested on a few datasets or with a few runs. In general, empirical results often
1176 depend on implicit assumptions, which should be articulated.
- 1177 • The authors should reflect on the factors that influence the performance of the approach.
1178 For example, a facial recognition algorithm may perform poorly when image resolution
1179 is low or images are taken in low lighting. Or a speech-to-text system might not be
1180 used reliably to provide closed captions for online lectures because it fails to handle
1181 technical jargon.
- 1182 • The authors should discuss the computational efficiency of the proposed algorithms
1183 and how they scale with dataset size.
- 1184 • If applicable, the authors should discuss possible limitations of their approach to
1185 address problems of privacy and fairness.
- 1186 • While the authors might fear that complete honesty about limitations might be used by
1187 reviewers as grounds for rejection, a worse outcome might be that reviewers discover
1188 limitations that aren't acknowledged in the paper. The authors should use their best
1189 judgment and recognize that individual actions in favor of transparency play an impor-
1190 tant role in developing norms that preserve the integrity of the community. Reviewers
1191 will be specifically instructed to not penalize honesty concerning limitations.

1192 **3. Theory assumptions and proofs**

1193 Question: For each theoretical result, does the paper provide the full set of assumptions and
1194 a complete (and correct) proof?

1195 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 open source our code, configuration and the prompt we used to make main experimental results reproducible. However, because we using third-party API server to conduct our experiment, it is normal to have some error. But the error would not affects our claims. Please refer to Appendix B and Appendix C for comprehensive details of the experiments.

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 have included example prompt in the supplementary material. And we open source our code.

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 include all information needed in Section 3, Section 4, Appendix B and Appendix C.

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: The paper reports statistically meaningful results by providing mean and standard deviation over multiple runs. Please see Section 3, Section 4, Appendix B and Appendix C.

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: Please see Appendix B and Appendix C for the setup of experiments.

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: [Yes]

Justification: We discuss the broader impact in Appendix F.

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, we cited the original owners of assets in our works and we use these assets follow the provided licences.

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.

1404 • If assets are released, the license, copyright information, and terms of use in the
1405 package should be provided. For popular datasets, `paperswithcode.com/datasets`
1406 has curated licenses for some datasets. Their licensing guide can help determine the
1407 license of a dataset.

1408 • For existing datasets that are re-packaged, both the original license and the license of
1409 the derived asset (if it has changed) should be provided.

1410 • If this information is not available online, the authors are encouraged to reach out to
1411 the asset’s creators.

1412 **13. New assets**

1413 Question: Are new assets introduced in the paper well documented and is the documentation
1414 provided alongside the assets?

1415 Answer: [Yes]

1416 Justification: We introduced our SIMWORLD simulator in this paper. Please refer to
1417 Section 2 and Appendix A.

1418 Guidelines:

1419 • The answer NA means that the paper does not release new assets.

1420 • Researchers should communicate the details of the dataset/code/model as part of their
1421 submissions via structured templates. This includes details about training, license,
1422 limitations, etc.

1423 • The paper should discuss whether and how consent was obtained from people whose
1424 asset is used.

1425 • At submission time, remember to anonymize your assets (if applicable). You can either
1426 create an anonymized URL or include an anonymized zip file.

1427 **14. Crowdsourcing and research with human subjects**

1428 Question: For crowdsourcing experiments and research with human subjects, does the paper
1429 include the full text of instructions given to participants and screenshots, if applicable, as
1430 well as details about compensation (if any)?

1431 Answer: [NA]

1432 Justification: We do not involve any research with human subjects.

1433 Guidelines:

1434 • The answer NA means that the paper does not involve crowdsourcing nor research with
1435 human subjects.

1436 • Including this information in the supplemental material is fine, but if the main contribu-
1437 tion of the paper involves human subjects, then as much detail as possible should be
1438 included in the main paper.

1439 • According to the NeurIPS Code of Ethics, workers involved in data collection, curation,
1440 or other labor should be paid at least the minimum wage in the country of the data
1441 collector.

1442 **15. Institutional review board (IRB) approvals or equivalent for research with human
1443 subjects**

1444 Question: Does the paper describe potential risks incurred by study participants, whether
1445 such risks were disclosed to the subjects, and whether Institutional Review Board (IRB)
1446 approvals (or an equivalent approval/review based on the requirements of your country or
1447 institution) were obtained?

1448 Answer: [NA]

1449 Justification: We do no involve crowdsourcing nor research with human subjects.

1450 Guidelines:

1451 • The answer NA means that the paper does not involve crowdsourcing nor research with
1452 human subjects.

1453 • Depending on the country in which research is conducted, IRB approval (or equivalent)
1454 may be required for any human subjects research. If you obtained IRB approval, you
1455 should clearly state this in the paper.

- 1456 • We recognize that the procedures for this may vary significantly between institutions
1457 and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the
1458 guidelines for their institution.
- 1459 • For initial submissions, do not include any information that would break anonymity (if
1460 applicable), such as the institution conducting the review.

1461 16. Declaration of LLM usage

1462 Question: Does the paper describe the usage of LLMs if it is an important, original, or
1463 non-standard component of the core methods in this research? Note that if the LLM is used
1464 only for writing, editing, or formatting purposes and does not impact the core methodology,
1465 scientific rigorousness, or originality of the research, declaration is not required.

1466 Answer: [Yes]

1467 Justification: We describe in detail how we deploy LLMs powered agents in our work in
1468 experiment section. Please refer to Section 2, Section 3, Section 4, Appendix A, Appendix B
1469 and Appendix C.

1470 Guidelines:

- 1471 • The answer NA means that the core method development in this research does not
1472 involve LLMs as any important, original, or non-standard components.
- 1473 • Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>)
1474 for what should or should not be described.