

DeliveryBench: Can Agents Earn Profit in Simulated Worlds?

Lingjun Mao¹ Jiawei Ren¹ Jixuan Chen¹ Ziqiao Ma²

Kun Zhou¹ Lianhui Qin¹

¹University of California, San Diego ²University of Michigan

lingjun@ucsd.edu

Abstract

LLMs and VLMs are increasingly deployed as embodied agents, yet existing benchmarks largely revolve around simple short-term tasks and struggle to capture rich, realistic constraints that shape real-world decision-making. To close this gap, we propose DELIVERYBENCH, a city-scale embodied benchmark that simulates the real-world profession of food delivery. Food couriers naturally operate under long-horizon objectives (maximizing net profit over hours) while managing diverse constraints, *e.g.* delivery deadline, transportation expense, vehicle battery, and necessary interactions with other couriers and customers. DELIVERYBENCH instantiates this setting in procedurally generated 3D cities with diverse road networks, buildings, functional locations, transportation modes, and realistic resource dynamics, enabling systematic evaluation of constraint-aware, long-horizon planning. We benchmark a range of VLM-based agents across nine cities and compare them with human players. Our results reveal a substantial performance gap to humans, showing that these agents are short-sighted and frequently violate basic commonsense constraints. Additionally, we observe distinct personalities across models (*e.g.* adventurous GPT-5 vs. conservative Claude), highlighting both the brittleness and the diversity of current VLM-based embodied agents in realistic, constraint-dense environments. Our code, data, and benchmark are publicly available at [deliverybench.github.io](https://github.com/deliverybench).

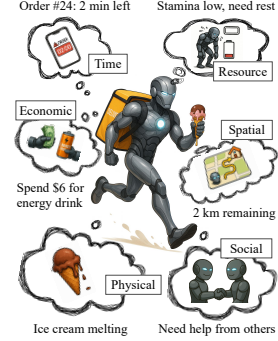
1 Introduction

Building on the strong abilities of large vision-language models (VLMs) in solving diverse real-world problems (Luo et al., 2025; Wang et al., 2025b; Anthropic, 2025b), recent research has begun exploring embodied agents that can perceive, reason, and act in physical environments (Liu et al., 2025; Hong et al., 2025; Li et al., 2024). Looking ahead, humans increasingly envision AI agents that may one day operate autonomously in the real world, helping with household tasks, participating in scientific discovery, or even earning income on our behalf. To move toward this vision, the community has developed a series of embodied-agent planning benchmarks that approximate real-world challenges through simulated environments, including 3D simulators (Yang et al., 2025a; Cheng et al., 2025; Zhong et al., 2025) and open-world games such as Minecraft (White et al., 2025; Long et al., 2024). By defining grounded tasks and modeling realistic constraints, these platforms help evaluate emerging agent abilities and provide data to guide future system design or model training.

A core capability for autonomous agents operating in the real world is to **earn profit and sustain themselves economically**. Beyond completing isolated tasks, a truly capable agent should be able to survive, adapt, and even develop a long-term career, navigating decisions that balance cost, benefit, and risk in the real world. Building and evaluating such agents requires environments that faithfully reflect the complexity of everyday life, where decisions unfold over long horizons, and outcomes depend on interacting physical, economic, resource, and social factors. To study it, a realistic benchmark should not only support embodied perception and action, but also model the incentives, constraints, and trade-offs that determine whether an agent can accumulate profit and operate sustainably.

Table 1: **Comparison of major embodied benchmarks.** Benchmarks are compared across sequence length per episode and six constraint dimensions, with DELIVERYBENCH featuring longer horizons and more comprehensive multidimensional constraints (see Section 2.2).

Benchmark	Sequence Length (action steps)	— Task Constraints —					
		Spatial	Time	Resource	Physical	Economic	Social
BEHAVIOR (Pan et al., 2024)	—	×	×	×	✓	×	×
ManiSkill2 (Gu et al., 2023)	—	×	×	×	✓	×	×
CookBench (Cai et al., 2025)	> 100	✓	✓	×	✓	×	×
ALFRED (Shridhar et al., 2020)	~12	✓	×	×	✓	×	×
ReALFRED (Kim et al., 2024)	~12	✓	×	×	✓	×	×
EB-ALFRED (Yang et al., 2025a)	~12	✓	×	×	✓	×	×
ALFWorld (Shridhar et al., 2021)	~6	✓	×	×	×	×	×
VirtualHome (Puig et al., 2018)	~9	✓	×	×	✓	×	×
ET-Plan-bench (Zhang et al., 2025a)	<20	✓	×	×	✓	×	×
EmbrACE-3K (Lin et al., 2025)	~10	✓	×	×	✓	×	×
TEACH (Padmakumar et al., 2022)	—	✓	×	×	✓	×	✓
ProcTHOR (Deitke et al., 2022)	—	✓	×	×	✓	×	×
TaPA (Wu et al., 2023)	~25	✓	×	×	✓	×	×
DELIVERYBENCH	> 100	✓	✓	✓	✓	✓	✓



However, as shown in Table 1, existing benchmarks fall short of this goal. They either overemphasize short-horizon subtasks (*e.g.* navigation, pickup-and-drop) or fail to encode the nontrivial constraints that shape real decision-making.

In this paper, we introduce a realistic embodied-agent benchmark that demands long-horizon planning while adhering to multiple real-world constraints. To minimize the gap between simulation and reality, such a benchmark must be grounded in tasks that (i) *truly exist in the real world*, (ii) *naturally involve long-term objectives*, and (iii) *require agents to simultaneously manage diverse constraints*. After surveying a variety of real-world careers, we find that food delivery provides an ideal testbed. A delivery courier operating in a city must carefully sequence routes using appropriate transportation, interleave actions (*e.g.* recharging an e-scooter or purchasing tickets), and collaborate with others when needed, all to maximize completed orders and net earnings. An example is shown in Table 1.

We develop DELIVERYBENCH, a city-scale benchmark that evaluates embodied agents under physically and socially grounded delivery scenarios. Agents act as autonomous couriers navigating procedurally generated cities to maximize long-term profit. To capture the open-ended nature of real-world operations, DELIVERYBENCH features dynamic, interactive environments populated with diverse points of interest (POIs) and multiple modes of transportation, going beyond prior urban simulators (Chen et al., 2024; Hong et al., 2025; Wu et al., 2025) that primarily offer static visual scenes. As deliveries unfold across multiple in-game hours, agents must manage resources (*e.g.* stamina depletion), adapt to changing conditions, and strategically balance efficiency, timing, and cost. When multiple agents coexist, they further encounter social dynamics such as competition and collaboration. By jointly modeling economic, physical, and social dynamics within a unified embodied environment, DELIVERYBENCH provides a realistic and action-driven setting to test whether VLM-based agents can make and execute plans that genuinely improve financial outcomes.

Using DELIVERYBENCH, we conduct extensive experiments on state-of-the-art VLMs. Our results reveal several findings: (1) frontier VLM-based agents lag far behind human players, struggling with long-horizon, constraint-aware decision-making and frequently making naïve mistakes (*e.g.* forgetting to recharge an e-scooter); (2) multi-agent performance does not scale with team size and typically peaks with two-agent teams, suggesting coordination challenges; (3) context engineering on larger models yields significant gains in profit; (4) different VLMs exhibit distinct behavioral styles—GPT-5 appears adventurous, Claude more conservative, and Gemini comparatively careless.

2 DeliveryBench

In this section, we present DELIVERYBENCH, a long-horizon planning benchmark for evaluating VLM-based embodied agents in realistic, constraint-rich settings across diverse evaluation dimensions. An overview is illustrated in Figure 1.

2.1 Profit-Earning Task

We center our benchmark on the food-delivery scenario, where an agent works in a virtual city and aims to maximize net profit by continuously completing delivery orders.

2.1.1 Task Formulation

The delivery task is formalized as a *long-horizon constrained optimization problem*, where a VLM-based agent as a courier seeks to maximize *net profit* over an operational horizon T (e.g. two virtual hours). To do so, the agent must plan and execute a sequence of delivery and supportive tasks while respecting diverse real-world constraints.

Long-term Profit Target. The agent earns income from customer orders in two forms: (i) a base salary E_{base} upon successful delivery; and (ii) rating-based rewards E_{rating} , determined by factors such as delivery punctuality, freshness, and special instructions (e.g. face-to-face delivery). Meanwhile, operational costs (C) arise from purchasing items or services (e.g. recharging, vehicle rental). The total income and net profit are therefore

$$E = E_{\text{base}} + E_{\text{rating}}, \quad P = E - C. \quad (1)$$

Constrained Decision Making. At each step, the agent receives an observation O_t and selects an action $a_t = \pi_i(O_t)$ via policy π_i . The goal is to obtain an optimal policy π_i^* that maximizes expected net profit while satisfying all constraints $\mathcal{C}$. Let $\Pi_{\mathcal{C}}$ be the set of feasible policies whose induced trajectories obey all $c \in \mathcal{C}$. Formally,

$$\pi_i^* \in \arg \max_{\pi_i \in \Pi_{\mathcal{C}}} \mathbb{E}_{\pi_i}[P]. \quad (2)$$

To achieve this objective, the agent must coordinate both delivery-related tasks that directly contribute to revenue (e.g. selecting and fulfilling orders) and supportive tasks that indirectly maintain operational feasibility (e.g. recharging, resting, or purchasing supplies).

2.1.2 Test Environment

To support realistic and versatile task execution, we simulate a high-fidelity 3D urban environment featuring diverse city layouts, interactive points of interest (POIs), multiple transportation modes, and rich physical dynamics.

Simulated 3D City. Based on SimWorld (Ren et al., 2025), DELIVERYBENCH simulates 3D cities of different scales in Unreal Engine. Each city contains realistic buildings, roads, humans, and other objects. Unreal Engine’s weather control, physics simulation, and related capabilities also allow us to flexibly vary the environment for improving realism.

Interactive Infrastructure and POIs. Across all cities, buildings are sampled as POIs with equal probability, including restaurants, customer homes, convenience stores, car rentals, and rest areas. Infrastructure such as bus stops and charging stations is placed along the road network. When an agent arrives at these POIs and infrastructures, it can trigger context-specific actions (e.g. picking up food, recharging vehicles, renting cars, or resting).

Transportation, Navigation, and Physics. The environment supports multiple transportation modes (e.g. walking, e-scooters, cars, and public transit), with different speed, cost, and stamina profiles. Because current models struggle with low-level 3D navigation (Ramrakhya et al., 2022; Song et al., 2025), we provide a waypoint-based system that follows shortest paths while still exposing motion control. Physical dynamics (e.g. temperature, collisions, odor diffusion) further affect food quality, requiring agents to adapt routing for freshness.

2.2 Multifaceted Realistic Constraints

- **Spatial constraints:** Spatial constraints specify *where* actions can be executed. For example, pickups must occur at the designated restaurant and recharging is only possible at charging stations. Therefore, agents must visit the appropriate POIs for different actions.

- **Time constraints:** Time constraints restrict *when* tasks can be performed. Each task has a feasible time window, and some must follow a fixed order (*e.g.* delivery after pickup). When windows overlap, agents may interleave tasks for efficiency, such as delivering one order while preparing another. Deadlines also matter: late deliveries reduce income.
- **Resource constraints:** Agents must manage consumable resources such as stamina, vehicle battery, and cash. Depleting a resource limits the corresponding ability (*e.g.* an e-scooter cannot be used without battery), requiring supportive actions such as recharging. Agents may also trade one resource for another, *e.g.* spending cash to restore stamina.
- **Physical constraints:** Physical constraints describe how environmental dynamics affect outcomes. Temperature, motion, and collisions can degrade food condition (*e.g.* ice cream melts), so routing and transport choices must account for fragility and perishability.
- **Economic constraints:** Economic constraints govern trade-offs between income and cost. Agents earn from successful deliveries but also incur expenses such as recharging or buying supplies, requiring them to balance immediate costs against long-term benefits.
- **Social constraints:** In multi-agent settings, multiple couriers operate in the same city, introducing additional constraints from *collaboration* and *competition*. Agents may coordinate implicitly or explicitly, for example by serving different regions, but they also compete for scarce opportunities such as high-value orders and nearby charging spots.

2.3 Benchmark Construction

2.3.1 Task Setup

Multi-level Tasks Creation. We evaluate agents on nine procedurally generated city maps across three difficulty levels: *small* (11–15 roads), *medium* (16–25 roads), and *large* (26–30 roads). Each environment maintains a fixed-size active order pool, continuously replenished as orders are accepted. For each order, the system randomly samples a restaurant and a residential building as pickup and dropoff locations, and computes the wage and time limit from travel distance with slight stochastic perturbations. A fraction of orders include special customer requirements (*e.g.* face-to-face delivery), with penalties for violations. Each episode terminates when the agent reaches either the lifetime or API call budget.

Agent State Management. At the beginning of each episode, agents are spawned at a designated starting location in the city. They start with the same initial stamina, balance, battery level, and other state variables. As agents act, stamina and battery levels decrease according to their activities. At the end of each episode, we log the complete interaction trajectory, income, and expenses, which form the basis for our evaluation metrics.

2.3.2 Single- and Multi-agent Settings

We consider both single-agent and multi-agent settings. In the **single-agent** setting, one agent operates as the sole courier in each city. In the **multi-agent** setting, we deploy eight instances of the same agent in a shared environment, where they draw from a global order pool and share infrastructure such as charging stations. To control the degree of cooperation, we group them into four team structures: 8×1 (eight independent agents, purely competitive), 4×2 (four cooperating pairs), 2×4 (two groups of four), and 1×8 (a single fully cooperative team). Within each group, agents can communicate and respond to help requests, enabling coordination such as order handoff and teammate recharging.

2.3.3 Evaluation Metrics

Our primary performance metric is the hourly net profit $\bar{P}$ achieved in a 2-hour virtual episode. We report $\bar{P}$ aggregated over episodes as the main indicator. To diagnose where agents succeed or fail, we further evaluate model behavior along the following three capability dimensions, and more details about the evaluation metrics are in Appendix E.1.

- **High-level planning.** We measure time-sensitive long-term planning via order-selection quality, on-time delivery rate, time efficiency (effective delivery time including parallel orders, normalized by episode time), and active time ratio (fraction of time spent on purposeful actions rather than idling or being incapacitated).
- **Resource management.** We assess self-sustaining behavior using hourly stamina consumption, interruption count (*e.g.* stops due to resource depletion), and proactive prevention ratio (how often agents replenish critical resources before they run out).
- **Physical/environmental adaptation.** We evaluate how well agents handle implicit physical and environmental constraints using violation rate (fraction of orders with constraint violations), food-quality rating, and customer rating (both on a 0–5 scale). These metrics capture whether agents can handle realistic constraints.

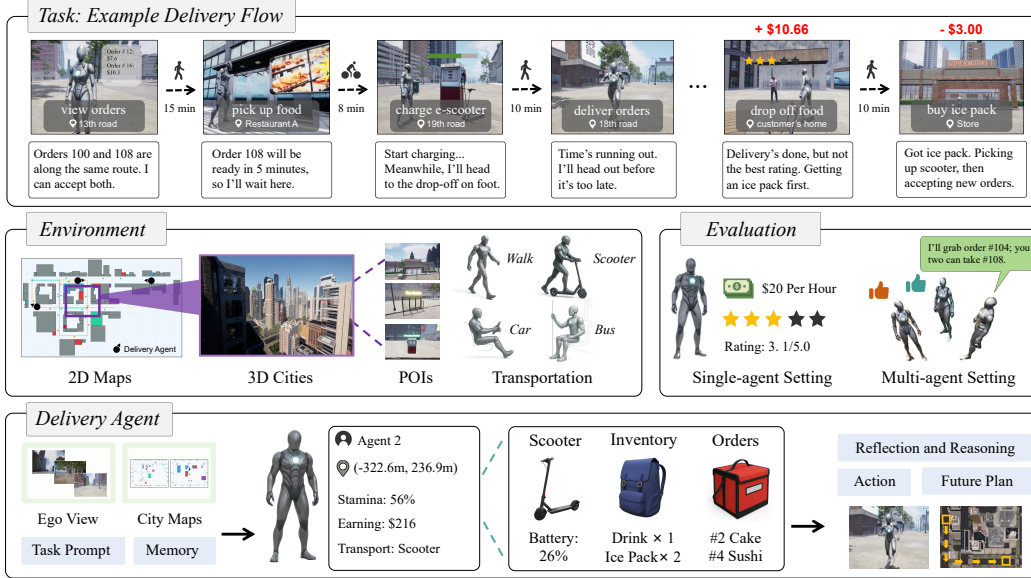


Figure 1: Overview of the DELIVERYBENCH environment and agent design. The process includes both core delivery actions (*e.g.*, accepting, picking up, and delivering orders) and supporting actions (*e.g.*, recharging e-scooters, purchasing items) for sustained delivery.

3 Agent Design

Each agent follows a perception–planning–execution loop over a rich embodied environment. At each timestep t , the agent perceives the city, reasons about its current tasks and constraints, and selects an action to update its trajectory and long-term plan.

Observation Space. The observation space aggregates multiple complementary views of the city and the agent’s operational status. A *global map* o_t^{global} shows the full city layout, including the agent’s location and major points of interest (POIs); a *local map* o_t^{local} provides finer-grained details of the nearby area; and a *first-person view* (FPV) o_t^{fpv} renders the agent’s embodied perspective, capturing streets, buildings, and surrounding objects. In addition, the agent can query *auxiliary information* o_t^{aux} via explicit actions, such as checking current orders, inventory, or public transport schedules.

Action Space. The action space in DELIVERYBENCH supports both high-level decision making and fine-grained embodied control, denoted as $\mathcal{A}$ (full details in Appendix C.3). High-level actions allow agents to delegate complex procedures to the simulator, *e.g.* using MOVE_TO with a target coordinate or POI for automatic path planning. Low-level actions provide direct control over movement and orientation, *e.g.* STEP_FORWARD and TURN_AROUND.

Interactive actions support environment manipulation and resource management, *e.g.* order pickup and dropoff, tool purchase and use, and vehicle recharging or rental.

Planning Pipeline. To model decision making over long horizons, we adopt a lightweight planning pipeline. At timestep t , the agent receives the current observation O_t and maintains a short-term memory $M_t = \{a_{t-k:t-1}\}$ of its past k actions. It also conditions on the previous plan P_t , produced at timestep $t-1$, and the most recent failure signal F_{t-1} , which indicates whether the last action or plan did not succeed as intended. The policy π_θ then outputs both the current action $a_t \in \mathcal{A}$ and an updated plan P_{t+1} :

$$(a_t, P_{t+1}) = \pi_\theta(O_t, M_t, P_t, F_{t-1}), \text{ where } O_t = \{o_t^{\text{global}}, o_t^{\text{local}}, o_t^{\text{fpv}}, o_t^{\text{aux}}\}.$$

Through this iterative update mechanism, the agent can continuously refine its future plan while reacting to new observations and failures in the environment, enabling more stable and adaptive behavior over long time horizons.

4 Experiments

4.1 Experimental Setup

Baseline Models. We test six closed-source models (GPT-5 (OpenAI, 2025), GPT-4o (OpenAI, 2024), Claude-3.7-Sonnet (Anthropic, 2025a), Grok-4 (xAI, 2025), Gemini-2.5-Flash (Comanici et al., 2025), and Gemini-2.5-Pro (Google DeepMind, 2025)) and three open-source models (Qwen2.5-VL-72B, Qwen2.5-VL-32B, and LLaMA-3.2-90B-Vision (AI, 2024)). For GPT-5, we use the “minimal” reasoning effort setting. We fix a temperature of 0 and a maximum length of 512 tokens. VLMs are accessed via the OpenRouter¹.

Human Baseline. To establish a meaningful reference for single-agent performance, we recruit three participants to independently complete the same delivery tasks. Each participant interacts via a custom GUI and follows the same evaluation protocol as the models. Interface details and screenshots are provided in Appendix D.1. We also record their delivery trajectories for subsequent supervised fine-tuning experiments.

Table 2: Global performance of different models across city sizes, measured by average hourly net profit $\bar{P}$ (\$/h), with detailed breakdown into base earnings (E_{base}), rating-based bonuses or penalties (E_{rating}), and expenses (C).

Model	Small City				Medium City				Large City			
	$\bar{P}$	E_{base}	E_{rating}	C	$\bar{P}$	E_{base}	E_{rating}	C	$\bar{P}$	E_{base}	E_{rating}	C
GPT-5	\$27.4	\$31.1	\$11.5	\$15.2	\$26.5	\$32.9	\$7.6	\$14.0	\$20.4	\$25.6	\$8.3	\$13.4
GPT-4o	\$10.4	\$23.6	\$6.8	\$20.0	\$13.9	\$25.4	\$4.9	\$16.3	\$11.9	\$20.6	\$5.3	\$13.9
Claude-3.7-Sonnet	\$31.3	\$30.1	\$14.8	\$13.6	\$31.2	\$35.7	\$10.5	\$14.9	\$25.8	\$30.1	\$13.0	\$17.2
Grok-4	\$10.4	\$40.1	\$10.1	\$39.8	\$13.9	\$29.7	\$14.6	\$30.4	\$16.7	\$33.9	\$10.8	\$28.0
Gemini-2.5-Flash	\$30.4	\$34.8	\$10.7	\$15.0	\$29.0	\$32.3	\$8.3	\$11.5	\$23.9	\$27.2	\$9.0	\$12.3
Gemini-2.5-Pro	\$45.1	\$56.7	\$19.5	\$31.1	\$42.1	\$53.2	\$13.5	\$24.6	\$38.2	\$48.0	\$10.3	\$20.1
Qwen2.5-VL-72B-Ins	\$5.4	\$15.1	\$3.8	\$13.5	\$6.3	\$15.6	\$3.3	\$12.6	-\$2.7	\$6.4	\$1.1	\$10.3
Qwen2.5-VL-32B-Ins	\$9.8	\$15.7	\$5.5	\$11.4	\$4.4	\$11.5	\$4.5	\$11.5	-\$0.1	\$8.7	\$2.3	\$11.1
LLaMA-3.2-90B-Vision-Ins	\$6.0	\$9.7	\$2.0	\$5.7	\$2.5	\$11.6	\$2.3	\$11.4	-\$0.9	\$7.0	\$1.3	\$9.3
Human	\$63.6	\$77.8	\$24.4	\$38.6	\$51.5	\$73.6	\$12.8	\$34.9	\$55.4	\$74.3	\$12.8	\$31.6

4.2 Single-Agent Planning Results

Global Performance. Table 2 summarizes the net profits earned over a 2 virtual-hour episode across models and city sizes. *Closed-source models consistently achieve higher net profit than open-source models*, with Gemini-2.5-Pro achieving the highest net profit across all city sizes. Its relatively better performance in large cities reflects an advantage

¹<https://openrouter.ai/>

in handling long-horizon tasks, which involve longer delivery routes and more complex routing decisions. In contrast, many open-source models even incur losses in these cities. We also observe that closed-source models tend to have higher expenses, but much of this reflects *strategic investment* for future deliveries (e.g., tool purchases), ultimately yielding higher profits. Nonetheless, *humans still outperform all models by a wide margin* across all city sizes. On average, they earn over \$50/hour, whereas most models earn less than \$30/hour. We analyze this gap via a multi-dimensional breakdown.

Table 3: Fine-grained evaluation results on DELIVERYBENCH. $\uparrow$ denotes that higher values are better, while $\downarrow$ denotes that lower values are better.

Model	Planning				Resources			Physical & Env.		
	Order $\uparrow$	OnTime $\uparrow$	TimeEff $\uparrow$	Active $\uparrow$	Stamina $\downarrow$	Interrupts $\downarrow$	Prevention $\uparrow$	Violations $\downarrow$	Food $\uparrow$	Cust $\uparrow$
GPT-5	3.38	0.34	0.89	0.56	1.13	1.17	0.75	0.72	3.93	3.96
GPT-4o	3.36	0.38	0.54	0.58	1.28	1.61	0.66	0.69	3.82	3.94
Claude-3.7-Sonnet	3.51	0.44	0.91	0.59	1.02	1.04	0.79	0.62	4.09	4.02
Grok-4	3.56	0.47	0.97	0.45	1.05	1.22	0.80	0.77	3.95	3.98
Gemini-2.5-Flash	3.31	0.27	0.98	0.54	1.24	1.42	0.62	0.75	3.93	3.86
Gemini-2.5-Pro	3.54	0.50	1.21	0.49	0.94	0.95	0.88	0.58	4.18	4.02
Qwen2.5-VL-72B-Ins	3.12	0.17	0.40	0.53	1.38	1.50	0.53	0.70	4.10	3.73
Qwen2.5-VL-32B-Ins	3.43	0.16	0.48	0.47	0.98	1.05	0.74	0.65	3.87	3.48
LLaMA-3.2-90B-Vision-Ins	3.31	0.04	0.54	0.53	1.39	1.66	0.59	0.69	3.98	3.45
Human	3.09	0.51	2.90	0.94	2.39	0.91	0.91	0.61	4.29	4.06

Fine-grained Analysis. Table 3 presents the results of the trajectory-level analysis.

- **Agents struggle to exploit temporal overlap compared with humans.** Agents fail to utilize their 2-hour window efficiently, often idling between actions (e.g., waiting to charge an e-scooter) instead of performing tasks concurrently (e.g., picking up food while charging), thereby wasting considerable time. They tend to deliver orders sequentially rather than leveraging spatiotemporal alignment to complete multiple deliveries in parallel. Consequently, their active-time and time efficiency remain substantially lower than those of humans.
- **Agents remain less self-sustaining, often neglecting resource management and preventive actions.** Most agents experience more than one interruption per hour due to stamina or battery depletion, and their proactive prevention ratios remain far below human results. Even stronger models, such as Gemini-2.5-Pro, often over-replenish when resources are sufficient and fail to act when depletion is imminent.
- **Agents struggle to handle implicit, environment-dependent constraints.** They often overlook many implicit rules in delivery, choosing improper placement or transport methods that degrade food quality and trigger customer complaints (e.g., placing ice cream with hot food, causing it to melt). These constraint violations remain frequent, with both food and customer ratings staying relatively low, ultimately reducing their income.

4.3 Multi-Agent Planning Results

We further test VLM-based agents in multi-agent settings, where competition and collaboration naturally emerge.

Global Performance. We report each model’s average net profit across all multi-agent group configurations on the medium-20roads map, as shown in Table 4. Most models show a decline in profit when transitioning from the single-agent setting (without any competition or coordination) to multi-agent conditions. Notably, GPT-4o exhibits the steepest drop. Compared to the purely competitive setup, all models except GPT-5 benefit from small-team cooperation, though their performance still remains well below the single-agent case.

Impact of Team Size. We analyze how team sizes affect coordination and interaction. As shown in Table 4, *most models perform best in pairs, but some show declines as team size grows*, especially in the four-agent setting. Although interaction events (e.g. messaging or help requests) rise with team size, they also increase coordination overhead, as agents must manage more potential help requests alongside their own tasks, making it harder to prioritize effectively (e.g., accepting help requests but forgetting to act). The detailed change in interaction frequency is provided in Appendix F.2.

Table 4: Multi-agent average hourly net profit ($\bar{P}$) under five settings: single-agent (1×1), fully competitive (8×1), and three cooperative structures. Underlines mark the best multi-agent setting.

Model	Per-Agent Hourly Net Profit ($\bar{P}$, \$/h)				
	(1×1)	8×1	4×2	2×4	1×8
GPT-5	\$27.3	<u>\$20.5</u>	\$19.5	\$8.7	\$16.5
GPT-4o	\$16.9	\$5.3	\$5.5	\$5.0	<u>\$6.9</u>
Claude-3.7-Sonnet	\$31.7	\$14.2	<u>\$22.6</u>	\$10.4	\$9.6
Grok-4	\$14.5	\$8.4	<u>\$13.6</u>	\$11.6	\$8.1
Gemini-2.5-Flash	\$28.4	\$21.2	<u>\$24.3</u>	\$12.6	\$15.1
Gemini-2.5-Pro	\$40.6	\$27.5	<u>\$36.4</u>	\$30.2	\$24.5
Qwen2.5-VL-72B-Ins	\$10.1	\$4.5	\$7.0	<u>\$8.7</u>	\$5.8
Qwen2.5-VL-32B-Ins	\$6.0	\$3.0	<u>\$4.6</u>	\$3.4	\$1.4
LLaMA-3.2-90B-Vision-Ins	\$1.4	\$1.4	<u>\$2.0</u>	\$1.3	\$1.5

4.4 Agent Planning-Style Analysis

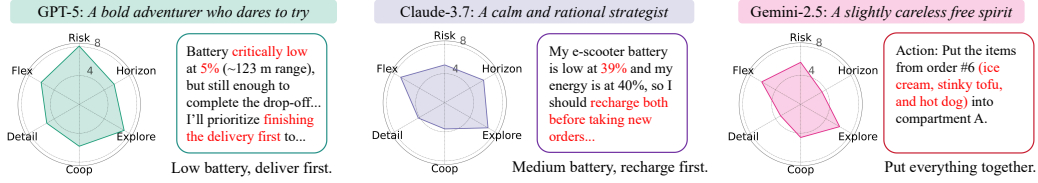


Figure 2: Comparison of model planning styles across six behavior dimensions, with example outputs provided as case studies.

During both single- and multi-agent evaluations, we observe distinct decision-making styles across models. For example, Claude behaves more cautiously, often heading to a charging station once battery is low, whereas GPT-5 is more aggressive and may continue deliveries even with a nearly depleted battery. To analyze such differences, we randomly sample delivery trajectories from each model and pair them with outcomes. GPT-4o then evaluates each decision step across six dimensions on a 0–10 scale, including *Risk* (how aggressive the decision is), *Horizon* (preference for long-term planning or short-term gains), *Explore* (tendency to try new strategies), *Coop* (willingness to cooperate with others), *Detail* (attention to operational and contextual factors), and *Flex* (frequency of plan adjustments). Dimensions irrelevant to a given step are skipped. Figure 2 presents representative models with their planning styles and example outputs, and the full set of model evaluations, including action patterns, transportation modes, and spending distributions, is provided in Appendix F.3.

4.5 Effects of Context Engineering, SFT, and RL

We evaluate several widely used strategies for improving agent performance, including Context Engineering, Supervised Fine-tuning (SFT) with human demonstrations, and Reinforcement Learning (RL), along with a baseline where the model outputs raw actions without explicit planning for reference. All evaluations in this section are conducted on the medium-20roads map.

Context Engineering. Context Engineering aims to enhance model reasoning through self-reflection on prior experience and environmental feedback. We evaluate two methods: Agentic Context Engineering (ACE (Zhang et al., 2025b)) and Dynamic Cheatsheet (DC (Suzgun et al., 2025)). Each model undergoes a 4-hour warm-up phase, during which it updates an internal memory by summarizing key patterns from its past trajectories. This memory is then frozen for evaluation. As shown in Table 5, context engineering consistently improves performance for GPT-5 and Claude-3.7-Sonnet, while the weaker open-source model Qwen2.5-VL-72B benefits little, with ACE even leading to a decline. Example memory summaries are provided in the Appendix F.4.

Supervised Fine-tuning and Online Reinforcement Learning. We study whether post-training can improve embodied agents in DeliveryBench through SFT and online RL. Starting from LLaVA-OneVision-8B (An et al., 2025), we fine-tune the model on a small set of human delivery trajectories, using either raw actions or GPT-4o-annotated actions augmented with *reasoning*, *reflection*, and *future plans*. Fine-tuning on raw actions induces more human-like behaviors (e.g., attempting to rent a car) but leads to worse overall performance, whereas the annotated variant performs better. For RL, following VAGEN (Wang et al., 2025a), we use trajectory-level net profit increase as the reward. RL substantially improves the model, but the learned policy still converges to a greedy single-order routine rather than discovering more advanced strategies such as parallelizing multiple orders. Detailed settings and additional analyses are provided in Appendix F.4.

Table 5: Results of planning, context engineering, SFT, and RL. Green and red indicate improvements and regressions relative to the corresponding baseline, respectively.

Model	$\bar{P}$	$\bar{E}$	$\bar{C}$
GPT-5 (with Plan)	\$27.3	\$38.8	\$11.5
GPT-5 (w/o Plan)	\$8.6	\$16.8	\$8.2
GPT-5 (with Plan + ACE)	\$33.2	\$46.1	\$12.9
GPT-5 (with Plan + DC)	\$36.2	\$47.3	\$11.2
Claude-3.7-Sonnet (with Plan)	\$31.7	\$51.6	\$19.9
Claude-3.7-Sonnet (w/o Plan)	\$19.2	\$25.6	\$6.3
Claude-3.7-Sonnet (with Plan + ACE)	\$40.5	\$56.3	\$15.8
Claude-3.7-Sonnet (with Plan + DC)	\$44.5	\$57.1	\$12.6
Qwen2.5-VL-72B (with Plan)	\$2.3	\$14.0	\$11.7
Qwen2.5-VL-72B (w/o Plan)	\$2.0	\$10.8	\$8.8
Qwen2.5-VL-72B (with Plan + ACE)	\$0.1	\$14.3	\$14.2
Qwen2.5-VL-72B (with Plan + DC)	\$3.2	\$16.6	\$13.4
LLaVA-OneVision-8B (original)	-\$7.2	\$4.4	\$11.6
LLaVA-OneVision-8B (raw-action-ft)	-\$7.8	\$7.2	\$15.0
LLaVA-OneVision-8B (annotated-ft)	\$3.2	\$12.7	\$9.5
LLaVA-OneVision-8B (VAGEN-RL)	\$14.4	\$25.0	\$10.6

5 Related Works

VLM-based Embodied Agents. Recent advances in VLMs (OpenAI, 2025; Anthropic, 2025a; Comanici et al., 2025) and large-scale manipulation datasets (O’Neill et al., 2024; Bu et al., 2025) have accelerated the development of embodied agents (Zitkovich et al., 2023; Driess et al., 2023; Yang et al., 2025c; Peng et al., 2026) that map language instructions to grounded visual understanding and executable actions. While these agents perform well on short-horizon tasks, they remain challenged by complex long-horizon scenarios, motivating both agentic workflows (Wang et al., 2023; Mu et al., 2023) and training-based approaches (Zitkovich et al., 2023; Driess et al., 2023; Yang et al., 2025b). Agentic workflows improve adaptivity through mechanisms such as memory (Lei et al., 2025), reflection (Huang et al., 2023; Yang et al., 2025b), and feedback-driven correction (Yang et al., 2025d; Kumar et al., 2025), whereas training-based methods emphasize end-to-end (Intelligence et al., 2025) or distilled learning (Sumers et al., 2023). Yet, it remains unclear how these designs perform on tasks that capture the true long-horizon complexity of real-world settings.

Embodied Agent Benchmarks. Existing embodied benchmarks vary widely in abstraction level and planning horizon. Low-level control benchmarks such as BEHAVIOR (Srivastava et al., 2021), iGibson (Shen et al., 2021), SAPIEN (Xiang et al., 2020), and ManiSkill2 (Gu et al., 2023) emphasize fine-grained motor control and physical realism, often relying on high-fidelity simulators such as MuJoCo (Todorov et al., 2012) and PyBullet (Coumans & Bai, 2016). In contrast, long-horizon benchmarks such as ALFRED (Shridhar et al., 2020), ReALFRED (Kim et al., 2024), and TEACH (Padmakumar et al., 2022) focus on multi-step instruction following and structured task planning, while later extensions (Deitke et al., 2022; Lin et al., 2025; Puig et al., 2018; Shridhar et al., 2021; Zhang et al., 2025a) increase scene diversity, interaction complexity, or reasoning demands. However, most existing benchmarks overlook multidimensional constraints such as economic, resource, and social factors, and fall short in capturing open-ended long-horizon decision-making.

6 Conclusion

We introduced DELIVERYBENCH, an embodied benchmark to evaluate VLM-based agents under realistic, long-horizon delivery scenarios. In the grounded food-delivery profession,

agents must maximize long-term profit while simultaneously handling spatial, temporal, resource, physical, economic, and social constraints. By instantiating these demands in simulated 3D cities with diverse layouts, multiple transportation modes, and both single- and multi-agent regimes, DELIVERYBENCH provides a more faithful and diagnostic testbed for studying constraint-aware planning. Our experiments across nine cities with a range of state-of-the-art VLMs reveal a substantial gap between current agents and human couriers, with agents exhibiting short-sighted behavior and frequent violations of basic common-sense constraints. Moreover, different models display distinct behavioral personalities, highlighting both diversity and brittleness in current VLM-based agents.

Looking ahead, DELIVERYBENCH opens avenues for developing agents with stronger temporal abstraction, hierarchical planning, and self-sustaining awareness, as well as for exploring richer memory architectures and training regimes tailored to long-horizon control.

References

- Meta AI. Llama 3.2 vision (90b) model card. <https://huggingface.co/meta-llama/Llama-3.2-90B-Vision>, 2024. Accessed 2025-10-25. 6
- Xiang An, Yin Xie, Kaicheng Yang, Wenkang Zhang, Xiuwei Zhao, Zheng Cheng, Yirui Wang, Songcen Xu, Changrui Chen, Chunsheng Wu, et al. Llava-onevision-1.5: Fully open framework for democratized multimodal training. *arXiv preprint arXiv:2509.23661*, 2025. 9
- Anthropic. Claude 3.7 sonnet and claude code. <https://www.anthropic.com/news/claude-3-7-sonnet>, 2025a. Accessed: 2025-02-24. 6, 9
- Anthropic. Claude code: Best practices for agentic coding, 2025b. URL <https://www.anthropic.com/engineering/claude-code-best-practices>. 1
- Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. Agibot world colosseum: A large-scale manipulation platform for scalable and intelligent embodied systems. *IROS*, 2025. 9
- Muzhen Cai, Xiubo Chen, Yining An, Jiaxin Zhang, Xuesong Wang, Wang Xu, Weinan Zhang, and Ting Liu. Cookbench: A long-horizon embodied planning benchmark for complex cooking scenarios. *arXiv preprint arXiv:2508.03232v1*, 2025. 2
- Feng Chen et al. Embodiedcity: A benchmark platform for embodied agent in real-world city environment. *arXiv preprint arXiv:2410.09604*, 2024. 2
- Zhili Cheng, Yuge Tu, Ran Li, Shiqi Dai, Jinyi Hu, Shengding Hu, Jiahao Li, Yang Shi, Tianyu Yu, Weize Chen, et al. Embodiedeval: Evaluate multimodal llms as embodied agents. *arXiv preprint arXiv:2501.11858*, 2025. 1
- Gheorghe Comanici, Eric Bieber, Mike Schaeckermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025. 6, 9
- Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning, 2016. 9
- Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. Proctor: Large-scale embodied ai using procedural generation. *NeurIPS*, 2022. 2, 9
- Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. *ICML*, 2023. 9

- Google DeepMind. Gemini 2.5 pro model card. <https://deepmind.google/models/model-cards/>, 2025. Gemini 2.5 Pro entry updated Jun 27, 2025; accessed 2026-03-30. 6
- Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. Maniskill2: A unified benchmark for generalizable manipulation skills. *ICLR*, 2023. 2, 9
- Yining Hong, Rui Sun, Bingxuan Li, Xingcheng Yao, Maxine Wu, Alexander Chien, Da Yin, Ying Nian Wu, Zhecan James Wang, and Kai-Wei Chang. Embodied web agents: Bridging physical-digital realms for integrated agent intelligence. *NeurIPS*, 2025. 1, 2
- Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *CoRL*, 2023. 9
- Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi 0.5$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2505.21906*, 2025. 9
- Taewoong Kim, Cheolhong Min, Byeonghwi Kim, Jinyeon Kim, Wonje Jeung, and Jonghyun Choi. Realfred: An embodied instruction following benchmark in photo-realistic environments. *ECCV*, 2024. 2, 9
- Nishanth Kumar, William Shen, Fabio Ramos, Dieter Fox, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Caelan Reed Garrett. Open-world task and motion planning via vision-language model inferred constraints. *ICRA*, 2025. 9
- Mingcong Lei, Ge Wang, Yiming Zhao, Zhixin Mai, Qing Zhao, Yao Guo, Zhen Li, Shuguang Cui, Yatong Han, and Jinke Ren. Clea: Closed-loop embodied agent for enhancing task execution in dynamic environments. *IROS*, 2025. 9
- Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Erran Li Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking llms for embodied decision making. *NeurIPS*, 2024. 1
- Mingxian Lin, Wei Huang, Yitang Li, Chengjie Jiang, Kui Wu, Fangwei Zhong, Shengju Qian, Xin Wang, and Xiaojuan Qi. Embrace-3k: Embodied reasoning and action in complex environments. *arXiv preprint arXiv:2507.10548*, 2025. 2, 9
- Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, et al. Visualagentbench: Towards large multimodal models as visual foundation agents. *ICLR*, 2025. 1
- Qian Long, Zhi Li, Ran Gong, Ying Nian Wu, Demetri Terzopoulos, and Xiaofeng Gao. Teamcraft: A benchmark for multi-modal multi-agent systems in minecraft. *arXiv preprint arXiv:2412.05255*, 2024. 1
- Xiaoliang Luo, Akilles Rechartd, Guangzhi Sun, Kevin K Nejad, Felipe Yáñez, Bati Yilmaz, Kangjoo Lee, Alexandra O Cohen, Valentina Borghesani, Anton Pashkov, et al. Large language models surpass human experts in predicting neuroscience results. *Nature human behaviour*, 2025. 1
- Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhai Wang, Mingyu Ding, Jun Jin, Bin Wang, Jifeng Dai, Yu Qiao, and Ping Luo. Embodiedgpt: Vision-language pre-training via embodied chain of thought. *NeurIPS*, 2023. 9
- OpenAI. Gpt-4o mini. <https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/>, 2024. Accessed: 2024-07-18. 6
- OpenAI. Gpt-5 system card. <https://openai.com/index/gpt-5-system-card/>, 2025. Accessed 2025-10-25. 6, 9

- Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. 2024. [9](#)
- Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramithu, Gokhan Tur, and Dilek Hakkani-Tur. Teach: Task-driven embodied agents that chat. *AAAI*, 2022. [2](#), [9](#)
- Yingqian Pan, Yufan Zhou, Zang Feng, Wenbo Hu, Xiaotian Deng, Junyi Yang, Yali Liu, Guangjun Liu, Jie Hu, Guangtao Yu, Ruijie He, Hong Liu, Yang Zhang, Qihuan Wu, Jianye Hao, Wenxue Wang, Jun Guo, and Yang Liu. Large language models overcome the machine penalty when acting fairly but not when acting selfishly or altruistically. *arXiv preprint arXiv:2410.03724*, 2024. [2](#)
- Bo Peng, Pi Bu, Keyu Pan, Xinrun Xu, Yingxiu Zhao, Miao Chen, Yang Du, Lin Li, Jun Song, and Tong Xu. How foundational skills influence vlm-based embodied agents: A native perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 8322–8330, 2026. [9](#)
- Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. *CVPR*, 2018. [2](#), [9](#)
- Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. *CVPR*, 2022. [3](#)
- Jiawei Ren, Yan Zhuang, Xiaokang Ye, Lingjun Mao, Xuhong He, Jianzhi Shen, Mrinaal Dogra, Yiming Liang, Ruixuan Zhang, Tianai Yue, et al. Simworld: An open-ended realistic simulator for autonomous agents in physical and social worlds. *NeurIPS*, 2025. [3](#)
- Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne Tchapmi, et al. igibson 1.0: A simulation environment for interactive tasks in large realistic scenes. 2021. [9](#)
- Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. 2020. [2](#), [9](#)
- Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *ICLR*, 2021. [2](#), [9](#)
- Xinshuai Song, Weixing Chen, Yang Liu, Weikai Chen, Guanbin Li, and Liang Lin. Towards long-horizon vision-language navigation: Platform, benchmark and method. *CVPR*, 2025. [3](#)
- Sanjana Srivastava, Chengshu Li, Michael Graf, Unnat Aneja, Fei Xia, Gokul Demir, Roberto Martín-Martín, Joe Su, N. Hudson Lang, Jiajun Wu, et al. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. *CoRL*, 2021. [9](#)
- Theodore Sumers, Kenneth Marino, Arun Ahuja, Rob Fergus, and Ishita Dasgupta. Distilling internet-scale vision-language models into embodied agents. *ICML*, 2023. [9](#)
- Mirac Suzgun, Mert Yuksekogunul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. *arXiv preprint arXiv:2504.07952*, 2025. [8](#)
- Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. 2012. [9](#)
- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *TMLR*, 2023. [9](#)

- Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Chi Wan, Yiping Lu, Zhengyuan Yang, et al. Vagen: Reinforcing world model reasoning for multi-turn vlm agents. *arXiv preprint arXiv:2510.16907*, 2025a. 9
- Ke Wang, Juntong Pan, Linda Wei, Aojun Zhou, Weikang Shi, Zimu Lu, Han Xiao, Yunqiao Yang, Houxing Ren, Mingjie Zhan, et al. Mathcoder-vl: Bridging vision and code for enhanced multimodal mathematical reasoning. *ACL Findings*, 2025b. 1
- Isadora White, Kolby Nottingham, Ayush Maniar, Max Robinson, Hansen Lillemark, Mehul Maheshwari, Lianhui Qin, and Prithviraj Ammanabrolu. Collaborating action by action: A multi-agent llm framework for embodied reasoning. *arXiv preprint arXiv:2504.17950*, 2025. 1
- Wayne Wu, Honglin He, Jack He, Yiran Wang, Chenda Duan, Zhizheng Liu, Quanyi Li, and Bolei Zhou. Metaurban: An embodied ai simulation platform for urban micromobility. *ICLR*, 2025. 2
- Zhenyu Wu, Ziwei Wang, Xiuwei Xu, Jiwen Lu, and Haibin Yan. Embodied task planning with large language models. *arXiv preprint arXiv:2307.01848*, 2023. 2
- xAI. Grok 4. <https://x.ai/news/grok-4>, 2025. Accessed: 2026-03-30. 6
- Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapient: A simulated part-based interactive environment. *CVPR*, 2020. 9
- Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, et al. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *ICML*, 2025a. 1, 2
- Yi Yang, Jiaxuan Sun, Siqi Kou, Yihan Wang, and Zhijie Deng. Lohovla: A unified vision-language-action model for long-horizon embodied tasks. *arXiv preprint arXiv:2506.00411*, 2025b. 9
- Zhejian Yang, Yongchao Chen, Xueyang Zhou, Jiangyue Yan, Dingjie Song, Yinyao Liu, Yuting Li, Yu Zhang, Pan Zhou, Hechang Chen, et al. Agentic robot: A brain-inspired framework for vision-language-action models in embodied agents. *arXiv preprint arXiv:2505.23450*, 2025c. 9
- Zhutian Yang, Caelan Garrett, Dieter Fox, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Guiding long-horizon task and motion planning with vision language models. 2025d. 9
- Lingfeng Zhang, Yuening Wang, Hongjian Gu, Atia Hamidizadeh, Zhanguang Zhang, Yuecheng Liu, Yutong Wang, David Gamaliel Arcos Bravo, Junyi Dong, Shunbo Zhou, et al. Et-plan-bench: Embodied task-level planning benchmark towards spatial-temporal cognition with foundation models. *IROS*, 2025a. 2, 9
- Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*, 2025b. 8
- Fangwei Zhong, Kui Wu, Churan Wang, Hao Chen, Hai Ci, Zhoujun Li, and Yizhou Wang. Unrealzoo: Enriching photo-realistic virtual worlds for embodied ai. *ICCV*, 2025. 1
- Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *CoRL*, 2023. 9

A Future Research Directions

DELIVERYBENCH simulates real-world food-delivery task, which naturally involves long-horizon objectives (*e.g.* maximizing net profit) intertwined with diverse physical, social, and economic constraints, providing a testbed that more faithfully reflects the complexity of real-world decision-making. As a next step, we aim to further extend this platform in several important directions:

Real-time reasoning. In the current setup, the simulator pauses the environment whenever the model is “thinking”: order timers, battery levels, food freshness, and other dynamic states are frozen. In contrast, real-world decision-making unfolds in a continuously evolving environment, where time keeps progressing and other entities (*e.g.* couriers, pedestrians, customers) act in parallel. We plan to support real-time planning in future versions, where agents must reason within this dynamic setting and adapt to ongoing temporal and environmental changes (*e.g.*, adjusting their trajectory in real time to avoid pedestrians).

Learning from interaction data. Although DELIVERYBENCH currently serves primarily as an evaluation benchmark, the platform naturally supports collecting rich interaction data at scale. This makes it a useful testbed for studying how different learning paradigms, including supervised fine-tuning, reinforcement learning, and memory-augmented agents, adapt to long-horizon delivery tasks. As shown in Section 4.5, we present preliminary results with several representative approaches, but there remains substantial room for further investigation. For example, current RL mainly enables the model to acquire basic delivery workflows, but does not spontaneously induce more advanced strategies such as recharging proactively or parallelizing multiple orders by exploiting their temporal and spatial overlap. How to improve learning in such an open-ended decision space remains an important direction for future work.

B DELIVERYBENCH Details

We provide additional details of DELIVERYBENCH, including map construction, transportation and POI design, and several task-specific mechanisms (*e.g.* food categories).

B.1 City Maps and Spatial Layout

We construct nine city maps spanning three difficulty levels: *small* (11–15 roads), *medium* (16–25 roads), and *large* (26–30 roads), with three maps in each category. Every map contains a diverse set of POIs distributed across the road network, sampled under a uniform spatial density such that larger maps naturally include more POIs. For each city, we select the largest inscribed loop as the bus route, evenly place bus stops along it, and deploy a single bus that continuously travels on this route. The overall spatial layouts of the maps are illustrated in Figure 3, and the POI statistics for each map are summarized in Table 7.

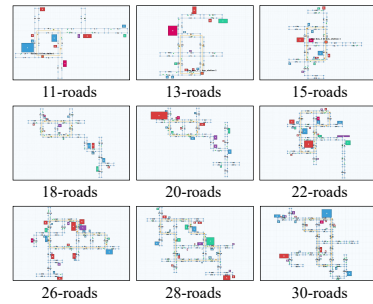


Figure 3: Overview of the nine procedurally constructed city maps used in our experiments.

B.2 Transportation Modes

We provide multiple transportation modes, including e-scooter, walking, driving, and public transit such as buses. These modes differ in speed, stamina consumption, and additional costs (*e.g.*, bus fares, car rental fees), requiring the model to make context-dependent trade-offs. A summary of these transportation modes is provided in Table 6.

Table 6: Different transportation modes in DeliveryBench.

Mode	Speed (m/s)	Stamina (%/m)	Extra Cost
walk	2.0	0.08	–
e-scooter	6.0	0.01	battery 0.04%/m
drag e-scooter	1.5	0.10	–
car	12.0	0.008	rental \$1.0/min
bus	10.0	0.006	\$1 fare

B.3 Points of Interest

Our constructed city includes various POIs, each serving distinct functions. Agents must navigate the city and interact with these POIs to accomplish different subtasks.

Table 7: Counts of points of interest (POIs) on each DELIVERYBENCH map.

Size	#Roads	Restaurant	Store	Rest Area	Car Rental	Hospital	Charging Station	Bus Station	Bus Route
small	11	4	4	1	1	1	10	4	1
	13	5	4	1	2	1	15	6	1
	15	4	5	2	2	1	18	6	1
medium	18	6	7	2	3	1	20	6	1
	20	5	7	3	3	1	24	6	1
	22	7	7	3	3	1	22	8	1
large	26	7	9	4	4	1	29	8	1
	28	8	11	3	4	1	29	8	1
	30	9	9	4	3	1	24	8	1

Restaurant. Restaurants serve as the pickup locations for delivery orders. Once an order is accepted, the restaurant begins food preparation. When the meal is ready, its state (e.g., temperature or freshness) starts changing over time, and the agent can visit the restaurant to collect it.

Store. Stores provide agents with access to purchasable items, including energy drinks, e-scooter batteries, and food-preservation tools such as ice packs and heat packs. The prices and functions of these items are listed in Table 8.

Table 8: Prices and functions of store items.

Item	Price (\$)	Function
Energy Drink	6	Restore 50% of stamina
E-Scooter Battery	10	Fully recharge e-scooter battery
Ice Pack	3	Cool food temperature
Heat Pack	3	Heat food temperature

Rest Area. Rest areas provide couriers with a place to recover stamina, allowing agents to restore 10% of their stamina per minute at no cost while resting.

Car Rental. Car rental stations allow agents to rent and return cars. An agent can pick up a car at any rental station and return it to any other. Rental fees are time-based and cost \$0.5 per minute, even when the vehicle is not in use.

Hospital. Hospitals handle agent recovery when stamina is fully depleted. An agent who collapses is automatically sent to a hospital for a 30-minute recovery process, during which no actions can be performed and a \$5 service fee is charged. All environment dynamics, such as order timers and food freshness, continue to progress normally. After recovery, the agent resumes work starting from the hospital.

Charging Station. Charging stations provide recharging services for agents’ e-scooters, with each station able to serve only one scooter at a time. The charging cost is \$0.05 per unit of battery, and the charging speed is 10 units per minute. Agents may stop charging and retrieve their e-scooters at any time.

Bus Station. Bus stations allow agents to wait for the arriving bus and board it when it reaches the stop. Upon arrival, agents may pay a \$1 ticket fee and ride the bus to any other station on the route.

B.4 Food Attributes

We simulate 22 food types, each with a preparation time and several quality-related attributes. These attributes influence how the food evolves during delivery and influence the agent’s strategy. The main factors include temperature dynamics, fragility, and odor sensitivity.

Temperature Dynamics. Temperature is the most influential factor affecting food quality. After preparation, a food item’s temperature evolves according to a lightweight thermodynamic model that simulates heat exchange with its surroundings. Each item has a temperature T_i and heat capacity C_i , while each storage compartment has an air node with temperature T_a and a small heat capacity C_{ab} . Items outside the insulated bag exchange heat with ambient air, whereas items inside the bag primarily exchange heat with others in the same compartment. We update temperatures using a discrete heat-exchange rule with timestep Δt :

$$S = \sum_i C_i (T_i - T_a), \quad (3)$$

$$T_a^{\text{new}} = T_a + \alpha \frac{S}{C_{ab}}, \quad (4)$$

$$T_i^{\text{new}} = T_i + \alpha (T_a - T_i), \quad (5)$$

where S denotes the net heat flow from the food items to the air node. The coefficient $\alpha = \Delta t / \tau_{\text{ex}}$ controls the exchange rate and is clipped to $\alpha \leq 0.5$ for numerical stability, while τ_{ex} determines the effective speed of heat transfer.

Fragility. Items such as cakes and soups are sensitive to movement and require gentle handling. Actions involving rapid movement (e.g., riding an e-scooter at high speed or running) introduce a risk of damaging these items. Each fragile item accumulates a fragility score when subjected to excessive vibration or acceleration. Once the accumulated damage exceeds a threshold, the food is considered ruined.

Odor Sensitivity. Strong-smelling foods (e.g. stinky tofu or durian) can affect other items stored in close proximity. When such foods are placed in the same insulated compartment as milder items, prolonged storage can lead to odor transfer. We model this using a simple odor-mixing mechanism. Each food item maintains an odor level $o_i \in [0, 1]$, and items within the same compartment gradually converge toward the highest odor level present in that compartment:

$$o_i^{\text{new}} = o_i + \alpha (o_{\text{max}} - o_i),$$

where o_{max} is the maximum odor level among items in the compartment, and α is a small timestep-based update coefficient. If $o_{\text{max}} = 0$, no odor transfers.

B.5 Order Attributes

Orders serve as the fundamental task units in our simulation. Each order specifies a designated pickup restaurant, a drop-off address, a delivery time window, and an associated wage. Some orders may also include special customer requests, which agents must carefully consider during fulfillment. Upon successful delivery, the system automatically settles the base wage and applies any additional bonuses based on customer ratings.

Delivery Methods. Agents may choose from four delivery methods: leaving the item at the doorstep, calling the customer, knocking on the door, or handing the order directly to the customer. For face-to-face delivery, the agent must first locate the customer’s actual position (e.g., “under the tree near the entrance”) and approach them to trigger the handoff.

The other methods only require reaching the designated building entrance. If the order includes no customer notes, any of the four methods is acceptable. However, if specific delivery instructions are provided, the agent must infer the most appropriate method from the context. For example, a note saying “I’m in a meeting” suggests the agent should leave the item at the door to avoid interruption, while high-value items may warrant direct handoff. Choosing an inappropriate delivery method can result in customer dissatisfaction and lower ratings.

Base Delivery Pay. Each delivery order includes a fixed base wage, which is granted in full if the agent completes the delivery within the specified time window or a short grace period (e.g. 1 minute). For late deliveries, the base pay is proportionally reduced based on the delay duration, but never falls below 30% of the original amount.

Customer Rating Bonus. Upon successful delivery, the customer provides a rating from 0 to 5 based on overall satisfaction. This rating influences the agent’s compensation through a bonus or penalty mechanism. The score reflects three main factors: total customer waiting time, food condition upon arrival, and the suitability of the chosen delivery method. If the rating exceeds 3 stars, the agent receives a bonus of up to \$3. If the rating falls below 3 stars, a fixed \$2 penalty is applied.

C Agent Input–Output Specification

In this section, we specify the delivery agent’s input and output formats, along with its action space.

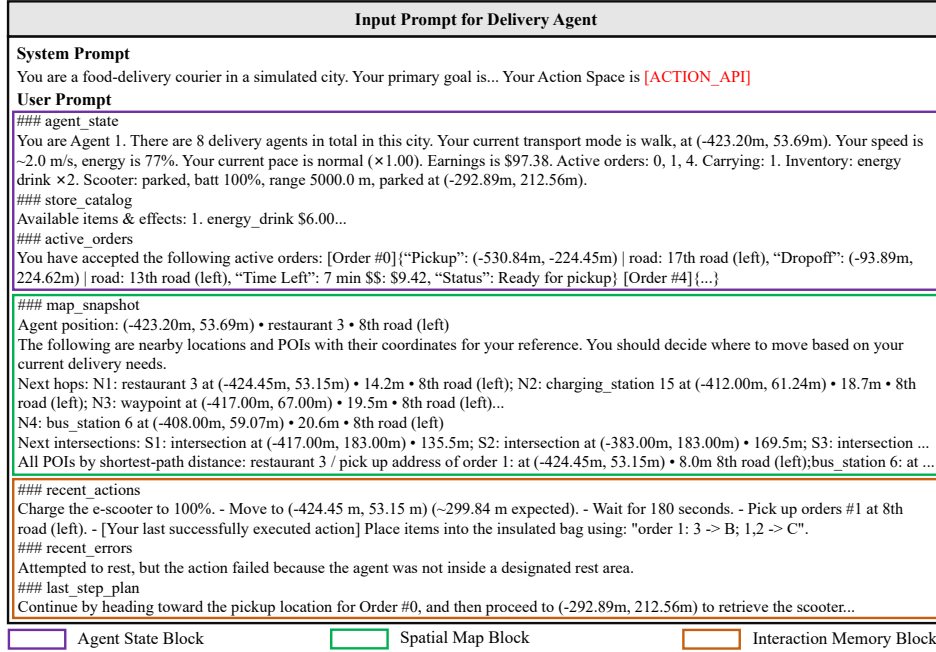


Figure 4: Overview of the input prompt used by delivery agents

C.1 Input Prompt Structure

At each decision step, the agent receives an input prompt that summarizes all information needed for planning and acting. The prompt consists of two parts: a *System Prompt*, which remains fixed throughout the episode, and a *User Prompt*, which is dynamically updated at every step. The System Prompt specifies the agent’s role in the simulated city, its primary delivery objective, and the available action space. The User Prompt then provides three additional components: (i) an *Agent State* block describing the agent’s current status, such

as its location, transport mode, speed, energy level, and active orders; (ii) a *Spatial Map* block encoding a compact map snapshot, including the next reachable waypoints, nearby intersections, and the locations of relevant POIs; and (iii) an *Interaction Memory* block recording recent actions, the previous step’s plan, and any error messages from failed actions. Sometimes the User Prompt also includes context-specific information; for example, arriving at a restaurant reveals the list of available pickups, and invoking an order-viewing action inserts the current order pool into the prompt. An example of the full prompt structure is shown in Figure 4.

C.2 Output Format

The agent follows a fixed structured format when producing its textual output. It first reflects on its recent memory and current state to formulate a *Reflection and Reasoning* paragraph that explicitly articulates the thought process behind the current decision. Based on this reasoning, the agent then outputs an *Action* specifying the concrete operation to execute. Finally, it provides a *Future Plan* describing how it intends to proceed after completing the current action.

C.3 Action Space

In DeliveryBench, the agent selects from a discrete action space of 30 actions, organized into several functional categories: (i) *Movement actions* allow the agent to navigate across the city, either through high-level navigation commands that invoke the built-in shortest-path planner or through simple low-level motion steps (e.g. stepping forward or turning around). (ii) *Order-handling actions* support core delivery operations such as browsing the order pool, accepting orders, and completing drop-offs. (iii) *Inventory and resource management actions* involve managing the agent’s internal resources, enabling it to regulate stamina, battery levels, and food conditions (e.g. resting, inspecting the bag, consuming energy drinks or battery packs). (iv) *Social and collaboration actions* facilitate multi-agent assistance, including viewing or posting help requests, accepting cooperative tasks, and simple communication. (v) *Transportation actions* allow the agent to switch transportation modes, rent or return vehicles, or use the public bus system.

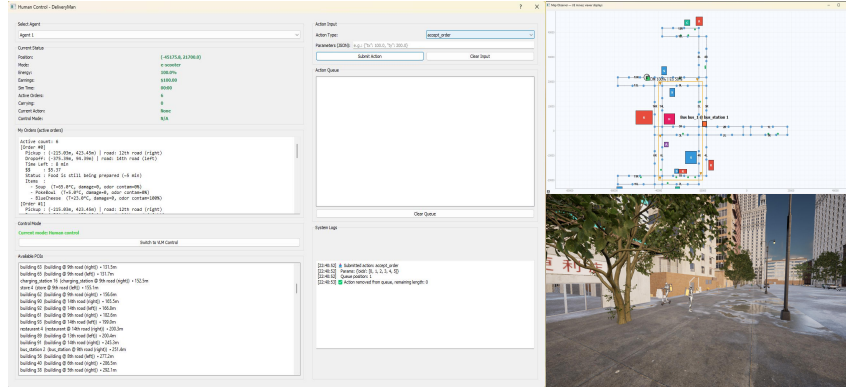


Figure 5: Human interaction GUI.

D Human Data Collection

To obtain a reasonable human performance reference and collect data for supervised fine-tuning, we recruited three human participants, each completing a two-hour delivery session independently. All experimental settings and evaluation protocols were kept identical to those used for the VLM agent. The resulting human trajectories were then augmented using GPT-4o to generate the corresponding reflection, reasoning, and future-plan annotations.

D.1 Human Interaction GUI

Human participants interacted with the environment via a custom-designed GUI that provides first-person observations, a map view, and contextual task information. Participants issued their actions directly through the interface. During delivery, the GUI displays real-time information such as the participant’s remaining stamina, current location, and accumulated earnings. All human trajectories are automatically logged by the system. A detailed illustration of the GUI is provided in Figure 5.

D.2 LLM-enhanced Annotation

Since the human trajectories only record the actions chosen at each step, we use GPT-4o to reconstruct the full chain-of-thought annotations in the same structured format described in Appendix C.2, ensuring consistency with the VLM agent’s outputs. For each human decision step, we provide GPT-4o with the corresponding observation and executed action, prompting the model to infer the underlying rationale behind the decision. We further supply the subsequent five human actions to GPT-4o, enabling it to generate the future plan aligned with those actions.

E Evaluation Details

E.1 Fine-grained Metric Definitions

Table 9: Fine-grained metrics for delivery agents; arrows indicate whether higher ($\uparrow$) or lower ($\downarrow$) values are better.

Dimension	Metric	Definition	Range
Planning	Order (Quality) $\uparrow$	Average relative quality of the orders selected by the agent, evaluated based on delivery-deadline feasibility relative to distance, reward relative to cost, and the alignment between the order’s delivery route and the agent’s current trajectory. Candidate orders are scored and ranked within the pool, with higher-ranked orders indicating higher quality.	[0, 5]
	OnTime (Rate) $\uparrow$	Proportion of selected orders delivered before their deadlines.	[0, 1]
	TimeEff (Time Efficiency) $\uparrow$	Sum of effective delivery durations for all delivered orders, including periods where multiple orders are handled in parallel, divided by the total episode time. Values greater than 1 indicate that the agent frequently handles multiple orders in parallel, values close to 1 indicate that the agent is almost continuously engaged in deliveries, and values below 1 indicate substantial idle time between deliveries.	[0, 1]
	Active (Rate) $\uparrow$	Fraction of time spent performing purposeful actions (<i>e.g.</i> moving, picking up, delivering, recharging), excluding waiting or incapacitated periods.	[0, 1]
Resources	StaminaUse $\downarrow$	Average stamina consumption per hour.	≥ 0
	Interrupts $\downarrow$	Number of forced interruptions per hour caused by resource depletion (<i>e.g.</i> stamina or battery exhaustion).	≥ 0
	Prevention $\uparrow$	Fraction of times the agent replenishes critical resources before they are depleted.	[0, 1]
Physical & Env.	Violations $\downarrow$	Proportion of orders that incur constraint violations, such as food-quality failures (<i>e.g.</i> melting, breakage, or odor transfer).	[0, 1]
	FoodRate $\uparrow$	Average rating of the food’s final quality upon delivery.	[0, 5]
	CustRate $\uparrow$	Average customer rating for each delivered order, reflecting overall satisfaction with factors such as waiting time, delivery behavior, and food condition.	[0, 5]

To analyze agent behavior beyond final delivery profit, we adopt a set of fine-grained metrics that capture different aspects of long-horizon delivery performance. These metrics assess high-level planning (order selection, deadline handling, time utilization), resource

management (stamina usage and proactive replenishment), and physical or environmental adaptation (food quality, constraint violations, customer satisfaction). Their formal definitions and computation methods are summarized in Table 9.

E.2 Planning Style Evaluation Prompts

Input Prompt for Planning Style Evaluation
You are a step-level evaluator for a delivery agent. For each evaluation step, you will receive: (1) a GLOBAL MAP image, (2) a LOCAL MAP image, and (3) TEXT containing the agent's prompt (observation/rules/context), (4) the agent's JSON output for THIS step, and (5) the resulting consequences of the chosen action. Your task is to assign 0–10 scores for the behavioral dimensions defined below. CRITICAL: Use ONLY the evidence present in THIS step's materials. If a dimension is not evidenced in this step, assign -1 for that dimension. Scoring dimensions (integers in [0,10], or -1 if not evidenced): 1) Risk (risk-taking vs conservatism): 10 = clearly high-risk behavior (e.g., preferring far or high-reward orders; continuing long-distance delivery with insufficient resources; accepting multiple orders simultaneously and attempting concurrent delivery.) 0 = strongly conservative behavior (e.g., one safe order at a time, pre-emptive charging/resting). 2) Horizon (long-term planning): 10 = explicit multi-step foresight (e.g., linking destinations, choosing spatially aligned orders; purchasing items for future benefit, or future plans extending beyond the next step). 0 = purely myopic, one-step reasoning. 3) Explore (strategy diversity): 10 = use of non-routine tools or strategies beyond the standard charge→accept→pickup→deliver loop (e.g., renting vehicles, taking buses, purchasing functional items, using alternative coordination strategies). 0 = strictly routine behavior. 4) Coop (collaboration): 10 = clear evidence of proactive collaboration (e.g., initiating coordination or dialogue with teammates; offering help, requesting assistance, jointly handling orders, or adjusting plans to support others). 0 = purely individualistic behavior focused solely on the agent's own tasks, with no attempt to cooperate. 5) Detail (attention to operational constraints): 10 = careful handling of perishables, temperature-sensitive items, fragile goods; correct drop-off methods, timing windows, melting risk, etc. 0 = clear oversight of important constraints. 6) Flex (adaptability to state changes): 10 = evident plan adjustments based on new information (e.g., noticing low battery and rerouting to charge; correcting mistakes). 0 = blindly following an outdated plan. Output policy: Judge ONLY from THIS STEP's provided materials (text, maps, action, and consequences); If the model output is malformed or missing, the step is skipped by the caller; Return JSON ONLY with EXACT keys and integer values: {"Risk": int, "Horizon": int, "Explore": int, "Coop": int, "Detail": int, "Flex": int}; No extra keys, no commentary, no markdown.

Figure 6: Prompt for planning style evaluation.

We use GPT-4o as an evaluator to assess the planning style exhibited by each model. At each evaluation step, GPT-4o is given (i) the current environment observation and (ii) the model's full output, which includes the chosen action, its chain-of-thought rationale, and the resulting consequences of that decision (e.g., whether an accepted order later times out or whether the action leads to future battery depletion). GPT-4o then scores this decision across multiple planning dimensions. The complete evaluation prompt used for scoring is shown in Figure 6.

F Additional Experimental Details and Results

In this section, we provide additional details for the experiments in the main paper, including more complete descriptions of the evaluation setup and supplementary experimental results.

F.1 Evaluation Environment and Protocol.

Our evaluation spans nine procedurally generated city maps, distributed across three difficulty levels. The order pool maintains 10 active orders, with 40% containing special customer requirements. We fix the weather to sunny with a temperature of 22°C. All VLM-based agents start with full stamina, an initial balance of \$100, and an e-scooter at 50% battery, together with basic insulation to slow food-quality degradation during transit. Agents continue acting in the virtual world until they reach either a 2-hour lifetime budget or a cap of 300 API calls. The simulation speed is set to three times that of real time. To avoid bias from model response latency, we pause each agent's lifetime clock, order timers, and food dynamics while it is reasoning. Time only advances when actions are executed. We fix random seeds to ensure identical order generation across runs. Each model is evaluated over eight independent runs per map, reporting average performance.

F.2 Interaction Frequency with Team Size

In the multi-agent setting, we evaluate how interaction frequency among models changes with team size, as shown in Figure 7. Although the communication rate tends to increase in larger teams, agents still interact only occasionally. However, this increase in communication does not improve task performance. As team size grows, coordination becomes more complex. Agents must balance maximizing their own utility with supporting their teammates, which makes effective cooperation more difficult. As a result, agents often overreact to teammate requests and abandon their own tasks, or they promise help but fail to follow through, leaving both sides stalled.

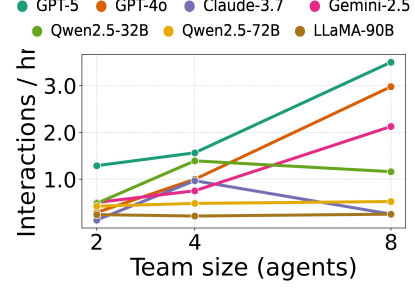


Figure 7: Interaction frequency across team sizes.

F.3 Model Behaviors and Planning Styles

In addition to the three examples of model planning styles shown in Figure 2, we evaluate the behaviors of all models, with the remaining results presented in Figure 8. We further analyze each model’s action distribution, spending patterns, and transportation choices. As shown in Figure 9, Stronger models such as GPT-5 and Claude-3.7-Sonnet exhibit broader action coverage and employ a richer set of strategies, such as renting cars or purchasing tools. In contrast, weaker open-source models such as LLaMA-3.2-90B-Vision-Ins primarily rely on simple pickup-and-delivery routines. These weaker models also end up spending more money on hospital rescues due to stamina depletion and often use less efficient transportation modes (e.g., walking or dragging scooters). Their spending patterns are summarized in Figure 10a, and their transportation preferences are illustrated in Figure 10b.

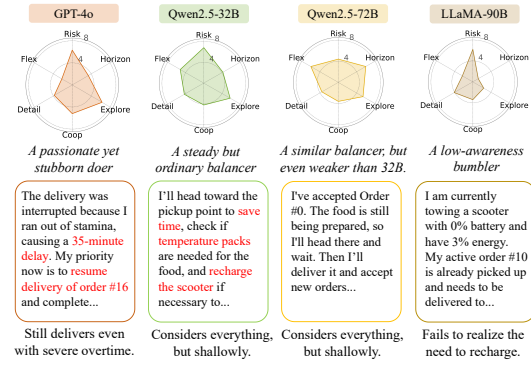


Figure 8: Planning style visualizations for the remaining four models, complementing the examples shown in Figure 2.

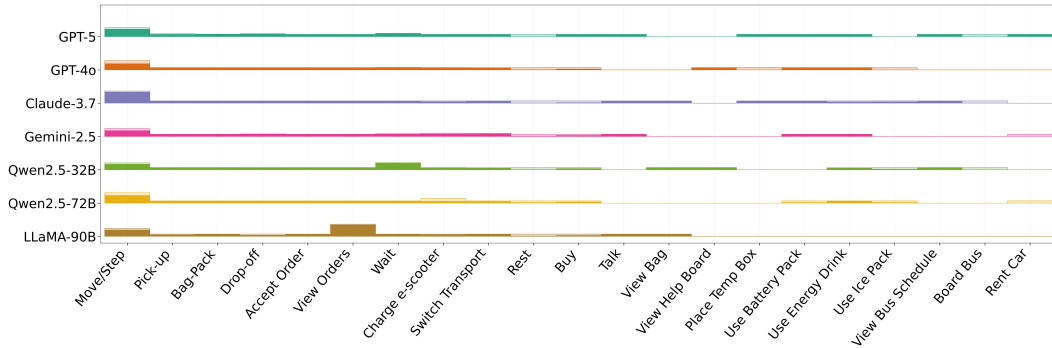


Figure 9: Action distributions of different models. For each model, the outer bars indicate the relative frequency of attempted actions, while the inner bars show the corresponding success rates.

F.4 Detailed Results for Context Engineering and Post-Training

We provide additional details that complement Section 4.5, including the experimental settings, more detailed metric breakdowns, and illustrative case studies of model-generated summaries under context engineering.

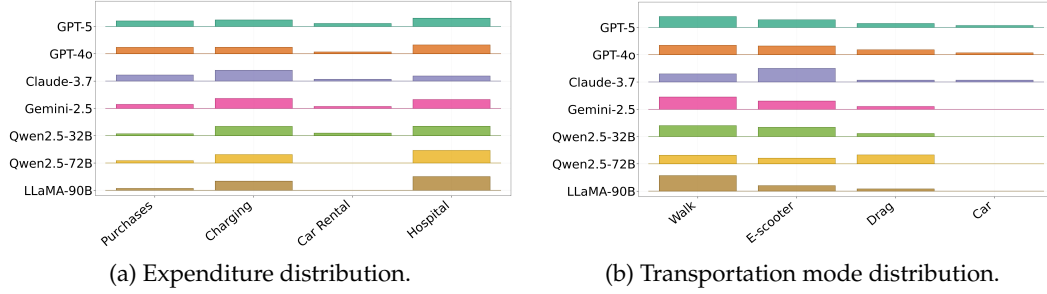


Figure 10: Model behavior statistics on spending and transportation choices.

SFT and RL Settings. For SFT, we use the trajectories that achieved the highest net profit in each city among human participants as training data, resulting in nine trajectories in total and 1,627 input-output pairs. Since the original records contain only raw human actions, we use GPT-4o to read the context associated with each action and retrospectively annotate *reasoning*, *reflection*, and *future plans* to simulate the human player’s thought process at that time. We fine-tune the model for five epochs on both the raw-action and annotated variants. For online RL, we follow the multi-turn VLM setting of VAGEN, treating each half-hour trajectory segment as one rollout and using the increase in net profit over the trajectory as the reward. We run 16 instances in parallel and train until performance no longer improves.

Fine-grained Analysis. We further analyze model performance along three dimensions: high-level planning, resource management, and physical or environmental adaptation. As shown in Table 10, context engineering generally leads to higher on-time delivery rates, better time efficiency, and a larger active-time ratio, which allow the models to complete more orders and achieve higher earnings. However, the gains in resource management and environmental handling are less substantial. For the human-trajectory fine-tuning experiments, fine-tuning directly on raw actions results in noticeable declines across multiple capabilities. In contrast, fine-tuning on annotated trajectories produces significant improvements. In particular, time-efficiency scores even exceed those of large models such as GPT-5 and Claude-3.7-Sonnet, indicating that the model successfully learns the human strategy of handling multiple orders in parallel. Moreover, compared with context engineering and SFT, online RL delivers more comprehensive improvements across all three dimensions.

Table 10: Fine-grained metrics for planning, resource usage, and physical/environmental behavior under context engineering and post-training. Green highlights improvements and red denotes regressions over the with-Plan baseline.

Model	Planning				Resources			Physical & Env.		
	Order↑	OnTime↑	TimeEff↑	Active↑	Stamina↓	Interrupts↓	Prevention↑	Violations↓	Food↑	Cust↑
GPT-5 (with Plan)	3.38	0.32	0.94	0.56	1.35	1.35	0.72	0.65	3.95	3.79
GPT-5 (w/o Plan)	3.24	0.25	0.45	0.48	1.32	1.86	0.48	0.75	3.35	3.20
GPT-5 (with Plan + ACE)	3.62	0.33	0.88	0.63	1.66	2.50	0.62	0.89	3.56	3.56
GPT-5 (with Plan + DC)	3.41	0.37	1.08	0.68	1.29	2.96	0.79	0.68	3.83	4.04
Claude-3.7-Sonnet (with Plan)	3.46	0.41	0.92	0.59	0.78	0.64	0.77	0.62	3.80	3.72
Claude-3.7-Sonnet (w/o Plan)	3.28	0.37	0.58	0.54	1.05	0.39	0.77	0.78	3.88	3.76
Claude-3.7-Sonnet (with Plan + ACE)	3.38	0.60	0.96	0.82	0.79	0.50	0.91	0.70	4.00	4.30
Claude-3.7-Sonnet (with Plan + DC)	3.41	0.52	1.06	0.77	1.22	1.06	0.54	0.72	3.92	4.16
Qwen2.5-VL-72B (with Plan)	3.12	0.17	0.40	0.53	1.38	1.50	0.53	0.70	4.11	3.73
Qwen2.5-VL-72B (w/o Plan)	3.07	0.21	0.38	0.51	1.42	2.13	0.24	0.75	3.61	3.35
Qwen2.5-VL-72B (with Plan + ACE)	2.97	0.14	0.88	0.63	1.76	3.00	0.40	1.00	3.80	3.40
Qwen2.5-VL-72B (with Plan + DC)	3.49	0.36	0.59	0.72	0.98	1.26	0.44	0.62	4.16	4.03
LLaVA-OneVision-8B (original)	3.22	0.05	0.15	0.50	2.32	2.49	0.16	0.74	3.67	3.52
LLaVA-OneVision-8B (human-ft)	3.05	0.06	0.72	0.38	2.49	2.99	0.14	0.82	3.63	3.04
LLaVA-OneVision-8B (annotated-ft)	3.36	0.16	1.51	0.88	0.64	2.38	0.47	0.58	4.02	3.96
LLaVA-OneVision-8B (VAGEN-RL)	3.44	0.35	0.81	0.65	1.14	2.42	0.26	0.52	4.06	4.01

Context Engineering Case Study. We present example notebooks generated by Claude-3.7-Sonnet and Qwen2.5-VL-72B under Agentic Context Engineering (ACE). In this setting, each model autonomously summarizes patterns from its past trajectories and maintains these summaries as persistent memory to guide future deliveries. For each model, we select the ten highest-quality examples, shown in Figure 11 and Figure 12. Both models extract principles covering multiple aspects of delivery, including time management and resource planning, and their summaries closely align with the underlying task rules. In

comparison, Claude-3.7-Sonnet produces more detailed and actionable guidelines, which in turn contributes to its larger performance improvement when ACE is applied.

Claude-3.7-Sonnet-Generated ACE Notebook
• When both scooter battery and personal energy are critically low, prioritize addressing vehicle battery first unless personal energy is below 25%, as mobility is typically the more constraining resource. • Before abandoning a scooter due to battery depletion, calculate whether purchasing a battery pack would be more time and energy efficient than walking to a charging station and back. • Maintain a continuous awareness of nearby resource restoration points (charging stations, stores, rest areas) and incorporate them into route planning before resources reach critical levels. • When managing multiple resource needs, plan a route that minimizes total travel distance by addressing needs at locations in close proximity rather than treating each need as a separate journey. • When multiple resource needs exist simultaneously (low personal energy, low scooter battery, pending orders), prioritize based on urgency and proximity to create efficient multi-objective routes. • Add a 5-minute buffer to all delivery time estimates to account for unexpected delays, traffic, or resource management needs that may arise during delivery. • Before accepting orders, calculate the total journey distance (to pickup + to delivery + return to strategic location) and verify both personal energy and scooter battery are sufficient with at least a 40% buffer. • Before accepting new orders, check the map for nearby resources (stores, charging stations, rest areas) to ensure access to necessary replenishment options during the delivery journey. • Avoid accepting orders that require significant backtracking, especially when handling overtime deliveries that require direct routing to minimize further delays. • Before accepting distant orders, evaluate all available transportation options (walking, scooter, public transit) based on current location and resource levels to determine the most efficient delivery method.

Figure 11: Example ACE notebook generated by Claude-3.7-Sonnet.

Qwen2.5-VL-72B-Generated ACE Notebook
• While waiting for orders to be prepared, use the time to scout for nearby orders or pre-plan delivery sequences to optimize efficiency. • When battery level drops below 20%, immediately prioritize moving to the nearest charging station before continuing deliveries. • Prioritize high payout-to-distance ratio orders when selecting batches to maximize efficiency and profitability. • When multiple orders are ready simultaneously, batch them based on similar drop-off directions to minimize backtracking and save time. • Regularly monitor energy levels and plan routes that include charging stations if needed, especially during long delivery sequences. • Consider the urgency of orders (overtime status) when planning delivery sequences to minimize penalties and improve customer satisfaction. • Utilize the map snapshot effectively to identify optimal paths and avoid unnecessary detours, ensuring efficient use of time and resources. • Consider alternative orders with shorter preparation times when faced with long wait times at pickup locations. • Monitor energy levels and plan breaks accordingly to maintain optimal performance throughout the shift. • Always assess the feasibility of new orders against current battery and energy levels to avoid overcommitting.

Figure 12: Example ACE notebook generated by Qwen2.5-VL-72B.

E.5 Ablation Studies

Planning Ablation. We further analyze the results reported in Table 5, which compare models that perform explicit plan-and-execute reasoning with models that directly output a single action. For GPT-5 and Qwen2.5, planning consistently improves most capability metrics and leads to higher net profit. In contrast, Claude-3.7-Sonnet earns more when planning is enabled, but its net profit decreases because of increased expenses. These additional costs mainly arise from overplanning, such as repeatedly recharging the e-scooter when the battery level is already sufficient or purchasing items that are not immediately necessary.

Waypoint Ablation. We evaluate whether VLM agents can navigate without privileged spatial priors. We remove preset waypoints and restrict them to step-by-step navigation using only low-level actions (STEP_FORWARD, TURN_AROUND) with egocentric observations. Agents fail to complete even a single order under this setting, indicating that current models struggle to translate visual understanding into embodied navigation. Explicit spatial coordinates remain a dependency for these models.

Greedy Rule-based Baseline. To better understand the difficulty of DeliveryBench and the current performance level of VLM agents, we implement a simple rule-based baseline based on greedy heuristics, such as selecting nearby high-paying orders and recharging when the battery falls below a fixed threshold. We find that this baseline achieves a non-trivial net profit, but still falls substantially behind strong VLM agents such as GPT-5 as well as human players. This result suggests that, because DeliveryBench exposes a highly open-ended action space, no single fixed strategy is sufficient to perform well across diverse situations.

Table 11: Comparison of a greedy baseline, GPT-5, and humans on medium cities.

Method	$\bar{P}$	E_{base}	E_{rating}	C
Greedy baseline	\$10.1	\$17.6	\$4.3	\$11.8
GPT-5	\$26.5	\$32.9	\$7.6	\$14.0
Humans	\$51.5	\$73.6	\$12.8	\$34.9

E.6 Variance and Stability Analysis

Robustness to Environment Conditions. To verify the stability of the task under different environment configurations, we evaluate agents on medium-sized cities under three profit conditions of increasing difficulty, *i.e.* progressively higher costs and tighter operational constraints, including higher charging fees, lower order rewards, and faster e-scooter battery consumption. As shown in Table 12, although absolute returns decrease as the environment becomes less favorable, the relative ranking of agents remains consistent across conditions. Table 12: Performance under three profit conditions of increasing difficulty on medium-sized cities. Although absolute returns vary across conditions, the relative ranking of agents remains consistent.

Setting / Model	Easy	Medium	Hard
Charging fee	\$0.050/unit	\$0.075/unit	\$0.100/unit
Avg. order reward	\$10/km	\$8/km	\$6/km
Battery drain	0.03%/min	0.04%/min	0.05%/min
Gemini-2.5-Flash	\$40.4	\$29.0	\$17.2
GPT-5	\$37.2	\$26.5	\$15.7
Qwen2.5-VL-72B	\$9.1	\$6.3	\$5.8

Run-to-run Stability. We further evaluate the stability of model performance under repeated runs. For both Gemini-2.5-Flash and Qwen2.5-VL-72B-Ins, we conduct three experimental groups, each following the same setup as the main experiment and consisting of eight independent runs along with their averaged results. As shown in Table 13, overall, both models exhibit low variance across runs, demonstrating stable and reliable performance under identical conditions.

Table 13: Mean and run-to-run variability for Gemini-2.5-Flash and Qwen2.5-VL-72B-Ins.

Metric	Gemini	Qwen
$\bar{P}$	\$28.46 $\pm$ 2.52	\$5.96 $\pm$ 2.82
E	\$37.55 $\pm$ 2.11	\$13.28 $\pm$ 2.90
C	-\$9.09 $\pm$ 1.32	-\$7.32 $\pm$ 0.96
Order	3.32 $\pm$ 0.11	3.07 $\pm$ 0.09
OnTime	0.30 $\pm$ 0.07	0.18 $\pm$ 0.05
TimeEff	0.88 $\pm$ 0.08	0.45 $\pm$ 0.04
Active	0.52 $\pm$ 0.04	0.50 $\pm$ 0.05
Stamina	1.03 $\pm$ 0.06	1.42 $\pm$ 0.09
Interrupts	1.79 $\pm$ 0.05	1.57 $\pm$ 0.10
Prevention	0.78 $\pm$ 0.06	0.55 $\pm$ 0.08
Violations	0.70 $\pm$ 0.11	0.68 $\pm$ 0.14
Food	4.08 $\pm$ 0.11	4.01 $\pm$ 0.17
Cust	3.77 $\pm$ 0.20	3.62 $\pm$ 0.25