EMBODIEDSCOPE: BENCHMARKING AGENTIC CAPABILITIES IN INTERACTIVE EMBODIED TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in general-purpose multimodal large language models (MLLMs) are converting them from passive perception systems into embodied AI agents that can perceive, reason, and act through interaction. Reliable execution of complex tasks requires agents to acquire missing evidence, retain task-relevant history, reason about physical constraints and action consequences, and respond to execution feedback. To assess the capabilities of embodied AI agents, we introduce EmbodiedScope, a diagnostic benchmark built on BEHAVIOR with 240 closed-loop household tasks under a humanoid setting. Embodied AI agents receive egocentric visual observations and act through a shared intermediate interface that abstracts fine-grained motor control while preserving task-level decisions. Controlled variants manipulate task-relevant evidence, interaction history, physical conditions, and execution outcomes, while tasks span basic interaction, capability-focused tasks, and structured multi-capability tasks. We evaluate representative MLLMs, including paired base and harnessed configurations for selected backbones. The best-performing system is comparatively stronger on Active Information Acquisition and Physical Reasoning, but weaker on Memory and Verification and Recovery. Performance drops sharply on L2, where multiple demands must persist and interact within the same episode. Agent-system support improves performance on selected capability subsets, but only partially mitigates the degradation on structured multi-capability tasks. EmbodiedScope bridges capability-level diagnosis and integrated embodied execution, enabling systematic studies of how capabilities that appear reliable in focused settings persist, interact, and fail as task demands become increasingly integrated.

1 INTRODUCTION

General-purpose multimodal large language models (MLLMs) are increasingly used as reasoning backbones for embodied AI agents. In this work, an evaluated configuration is an agentic system comprising an MLLM backbone and a harness that manages model interaction and the observation–decision–action loop. This progress makes it increasingly plausible to use them as the high-level reasoning core of embodied AI agents, where the model must interpret egocentric observations, decide what information is needed, select actions, and adapt its behavior from interaction feedback (Li et al., 2026; Zhang et al., 2026b). This shift changes the central question for embodied evaluation. It is no longer sufficient to ask whether a model can recognize objects, answer spatial questions, or solve an isolated planning problem; we need to determine whether the evaluated configuration can sustain the task-relevant reasoning required to complete an interactive task under a consistent embodied interface. Doing so requires evaluation that preserves the agent's responsibility for information acquisition, grounding, task-state maintenance, physical decision making, and response to execution outcomes, while avoiding both privileged environment access and overly high-level tools that solve these decisions on its behalf.

This carry-over is challenging because embodied tasks do not simply require multiple capabilities independently. The agent's own actions can change the conditions for later decisions: information acquired at one point may need to be remembered later, physical actions can change what is visible, reachable, or feasible, and unexpected outcomes can alter the task state and create new information needs. A capability that performs well in a focused setting must therefore remain effective as these

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: **Comparison with closely related interactive embodied-agent benchmarks.**

Work	Main focus	World-state goals	RGB only	Local actions	Current-frame reference
ESI-Bench (Hong et al., 2026)	Spatial intelligence	–	✓	✓	–
HumanCLAW (Li et al., 2026)	Whole-body action	✓	✓	✓	–
REAL (Mi et al., 2026)	Open-world manipulation	✓	–	–	✓
LongAct (Zhu et al., 2026)	Long-horizon planning	✓	–	✓	–
VisualAgentBench (Liu et al., 2025)	Visual-agent evaluation	✓	–	–	–
EmbodiedScope	**Agentic capability diagnosis**	✓	✓	✓	✓

demands persist and interact throughout an episode. We refer to the ability to maintain goal-directed task execution under such changing and interacting demands as *sustained embodied agency*.

To study these questions, we introduce **EmbodiedScope**, a diagnostic benchmark built on BEHAVIOR for evaluating embodied AI agents powered by general-purpose MLLMs on closed-loop household tasks with a humanoid embodiment. Agents receive egocentric visual observations and act through a shared intermediate interface that abstracts fine-grained motor execution while leaving task-level spatial and physical decisions to the agent. Rather than assigning capability labels after task generation, EmbodiedScope constructs controlled tasks by varying task-relevant evidence, interaction history, physical conditions, and execution outcomes. These variations target Active Information Acquisition, Memory, Verification and Recovery, and Physical Reasoning. The benchmark organizes tasks from basic interaction and capability-focused settings to structured multi-capability tasks, allowing the same agents to be evaluated as these capability demands persist and interact within increasingly integrated tasks.

Across 240 tasks, we evaluate representative multimodal large language models, including paired base and harnessed configurations for selected backbones. Our evaluation reveals three main findings. First, **capability-focused evaluation exposes uneven strengths and weaknesses**: GPT-6 Astra in the base configuration remains comparatively strong on Active Information Acquisition and Physical Reasoning, but performs worse on Memory and Verification and Recovery. Second, **strong capability-focused performance does not ensure equally strong multi-capability execution**: GPT-6 Astra with Codex achieves 85.0% average SR across L1, compared with 55.0% on L2. Third, **agent-system support provides substantial but selective gains**: harnessed configurations improve performance on specific capability demands, but do not yield consistent gains on structured multi-capability tasks.

In summary, our contributions are threefold. **First**, we introduce **EmbodiedScope**, a closed-loop benchmark based on BEHAVIOR for evaluating embodied agent systems powered by general-purpose MLLMs. The benchmark uses a humanoid embodiment and an intermediate interaction boundary that preserves task-level spatial and physical decisions while abstracting fine-grained motor execution.

Second, we develop a **capability-oriented and controlled task hierarchy** spanning basic interaction, focused demands in Active Information Acquisition, Memory, Physical Reasoning, and Verification and Recovery, and structured two- and three-capability tasks. Controlled variation in task-relevant evidence, interaction history, physical conditions, and execution outcomes enables these demands to be studied within a shared execution setting.

Third, we provide a **systematic evaluation of 240 embodied tasks** across representative multimodal large language models and agent-system configurations. The evaluation analyzes how performance changes from capability-focused to structured multi-capability execution and when additional agent-system support improves performance.

2 Related Work

2.1 Embodied Agent Evaluation

Recent embodied-agent benchmarks increasingly evaluate MLLMs through closed-loop interaction rather than isolated perception or language-only planning. Existing benchmarks realize this setting in different ways: ESI-Bench emphasizes active evidence acquisition and spatial reasoning through

egocentric interaction (Hong et al., 2026); SpatialWorld evaluates vision-based agents across diverse interactive 3D tasks under a shared high-level interface (Gao et al., 2026); HumanCLAW focuses on high-level embodied action selection while factoring out fine-grained motion execution (Li et al., 2026); REAL studies open-world mobile manipulation with richer semantic tools (Mi et al., 2026); and LongAct extends evaluation to substantially longer interaction horizons (Zhu et al., 2026). EmbodiedScope builds on this setting by combining complete task execution with controlled task construction: capability demands are embedded in executable tasks rather than inferred only from aggregate whole-task outcomes.

2.2 Diagnostic and Controlled Evaluation

A complementary line of work seeks to make embodied evaluation more diagnostic. Existing benchmarks target active information acquisition through exploration and evidence seeking (Hong et al., 2026; Zhang et al., 2026a), memory and task-state maintenance through history-dependent interaction (Chen et al., 2026b; Yadav et al., 2025; Wang & Li, 2026), physical reasoning through feasibility and relational constraints (Gundawar et al., 2026; Yang et al., 2026), and verification and recovery through failure detection and replanning (Liu et al., 2026b; Yan et al., 2026); other work decomposes embodied behavior into skills, planning components, or capability-oriented subsets (Li et al., 2024; Qi et al., 2025; Yang et al., 2025; Han et al., 2026). Controlled interventions and matched conditions further vary task state, history, feedback, or instructions to expose changes in agent behavior (Wang & Li, 2026; Tupini et al., 2026; Liu et al., 2026c; Chen et al., 2026b), while recent long-horizon and compositional evaluations study performance as multiple requirements accumulate (Han et al., 2026; **?**; Zhu et al., 2026). EmbodiedScope organizes tasks into three levels, from basic interaction to capability-focused and structured multi-capability tasks. Within this hierarchy, controlled variants modify task-relevant evidence, interaction history, physical conditions, or execution outcomes. This design allows the same execution framework to examine both individual capability demands and how those demands persist and interact as integration increases.

2.3 Action Abstraction and Agent Systems

What an embodied benchmark measures also depends on where the boundary between reasoning and execution is drawn. HumanCLAW lets the model select parameterized atomic skills while delegating low-level motion generation to the execution system (Li et al., 2026); ESI-Bench similarly abstracts motor control through camera-relative navigation and object-level manipulation (Hong et al., 2026); REAL exposes higher-level semantic navigation and search tools (Mi et al., 2026); ShowHarness maps semantic action units through embodiment-specific interpreters (Chen et al., 2026c); Harness VLA places a frozen VLA beneath a higher-level planner as a local executor (Zhang et al., 2026b); and CaP-X explicitly studies action abstraction and scaffolding as evaluation variables (Fu et al., 2026). These designs illustrate a fundamental trade-off: an interface that is too low-level can entangle task-level reasoning with motor control, embodiment, and controller quality, whereas an interface that is too high-level can perform search, grounding, object selection, planning, or recovery on the agent's behalf. EmbodiedScope places its measurement boundary between these extremes, abstracting fine-grained motor execution while preserving the task-level decisions the benchmark is intended to measure. The surrounding harness is treated separately: memory, verification, recovery, planning support, and skill libraries can substantially change system behavior (Chen et al., 2026a; Liu et al., 2026a; Huang et al., 2026; Chen et al., 2026d), so different model–harness configurations are reported as distinct agent systems rather than being conflated with backbone capability.

3 EmbodiedScope

3.1 Problem Setting and Interaction Boundary

Given a natural-language task instruction g, an agent interacts with the environment through a sequence of closed-loop decision steps. Let s_t denote the underlying environment state at step t. The agent receives the current egocentric visual observation o_t together with the public interaction context h_t available to the evaluated agent system, and selects a single action

$$a_t = \pi(g, o_t, h_t). \tag{1}$$

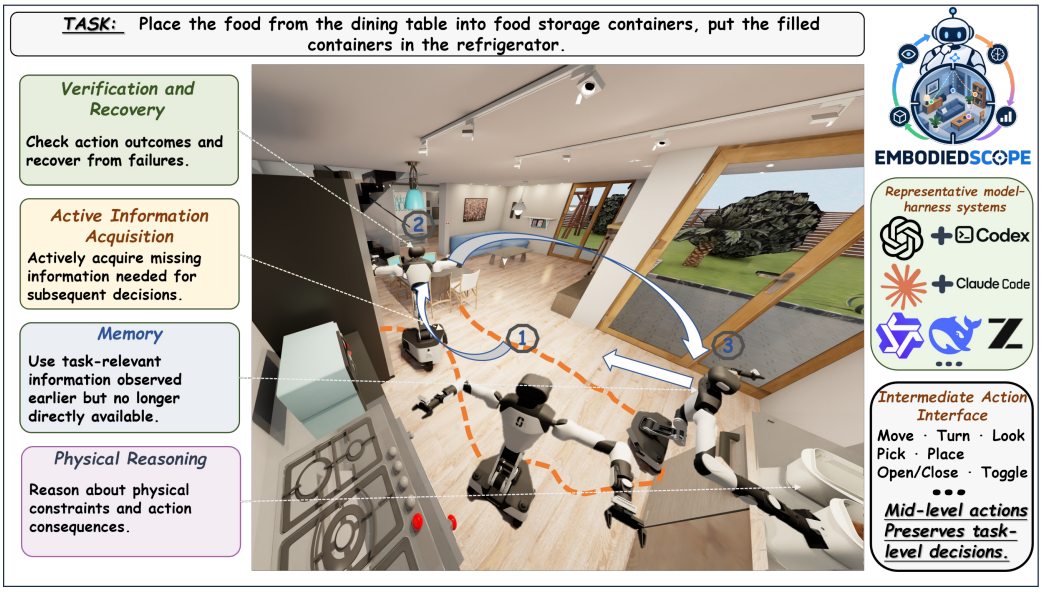

Figure 1: **Overview of EmbodiedScope.** EmbodiedScope evaluates multimodal agents on closed-loop household tasks through a shared intermediate action interface. The example episode illustrates how Active Information Acquisition, Memory, Physical Reasoning, and Verification and Recovery can arise and interact during task execution. Representative model–harness systems are evaluated under the same interaction interface, which abstracts fine-grained motor control while preserving task-level decisions.

The action is executed in the embodied environment, producing a public execution receipt ρ_t and a new environment state s_{t+1}:

$$(s_{t+1}, \rho_t) = \mathcal{E}(s_t, a_t), \qquad o_{t+1} = \mathcal{O}(s_{t+1}). \tag{2}$$

The resulting observation, action, and execution receipt become part of the public episode record before the next decision step. Task success is evaluated from the resulting world state by the environment's canonical verifier, rather than declared by the agent.

At each step, o_t is an egocentric RGB observation combining one head-camera view with two wrist-camera views. The environment directly provides the current observation together with the public interaction record from the five most recent completed steps, including their observations, actions, and execution receipts. It does not expose depth, semantic segmentation, object poses, scene graphs, verifier outcomes, or other privileged simulator state. Full proprioceptive state is also unavailable, except for the discretized motion summaries contained in public execution receipts.

In the *Base Agent Configuration*, no additional memory or retrieval mechanism is provided, so the environment-supplied recent context constitutes the agent's available interaction context. Additional model–harness configurations, described in Sec. 4, may retain or retrieve previously exposed episode information but receive no additional access to privileged environment state.

Object-directed actions use temporary visual markers rather than semantic object names or precise spatial coordinates. Allowing commands such as `pick(mug)` would let the interface resolve the intended object on the agent's behalf and weaken visual grounding. Requiring pixel locations, 3D coordinates, or precise manipulation poses, on the other hand, would introduce low-level localization and control requirements that are not the focus of our evaluation.

We therefore mark currently visible candidates with temporary labels such as `C1`, `C2`, ... The markers only provide a simple way to refer to objects in the current view; searching for objects, choosing the task-relevant instance, and keeping track of objects over time remain the responsibility of the agent.

EmbodiedScope uses an intermediate action interface to separate task-level decision making from fine-grained motor control. Requiring joint commands, continuous end-effector trajectories, or

precise motor control would make performance depend heavily on motion planning and control, which are not the main focus of our evaluation. At the other extreme, high-level tools such as `find_object`, `navigate_to(object)`, or whole-subtask skills would perform important parts of the task for the agent.

We therefore use body-relative movement, viewpoint control, and local interaction primitives such as picking, placing, opening, closing, and toggling. The agent still decides where to move and look, which object to interact with, where to place it, and how to respond to execution feedback, while the environment handles fine-grained motor execution. This preserves embodied constraints such as viewpoint, reachability, and collision without turning the benchmark into a low-level control problem.

3.2 Capability Design and Task Hierarchy

Whole-task execution can depend on different sources of task-relevant information and constraints. EmbodiedScope organizes these requirements into four capability dimensions according to the primary demand that shapes how the agent should proceed. Active Information Acquisition (A) concerns information that has not yet been obtained but is needed for later decisions. Memory (M) concerns information acquired earlier in the episode that is no longer directly available. Verification and Recovery (V) concerns feedback from current execution that reveals a difference between the expected and actual outcome. Physical Reasoning (P) concerns physical constraints and action consequences that determine which actions or plans remain feasible.

These dimensions are not intended to isolate mutually exclusive abilities. A task associated with one capability may still require perception, navigation, planning, memory, or other forms of reasoning to complete. Instead, each capability-focused task is constructed so that one demand is made especially salient and becomes the primary object of diagnosis. Because multiple demands can also persist and interact within the same episode, we further organize tasks by increasing levels of capability integration.

Capability dimensions. We define each capability dimension by the role it plays in determining how the agent should continue a task.

Active Information Acquisition (A). An A demand arises when the information currently available is insufficient to decide how to proceed. The agent must actively acquire additional evidence, for example by changing its viewpoint, moving to a new position, or interacting with the environment, and use that evidence to guide later actions.

Memory (M). An M demand arises when later actions depend on task-relevant information that was available earlier in the episode but is no longer directly available in the current observation. The agent must retain, update when necessary, and later use this information as the task unfolds.

Physical Reasoning (P). A P demand arises when choosing how to proceed depends on physical constraints or the likely consequences of actions. These may include reachability, containment, support, clearance, articulation, and dependencies between actions that determine which strategies are physically feasible.

Verification and Recovery (V). A V demand arises when the agent must determine whether an action produced the expected outcome and use the resulting observation or execution feedback to decide what to do next. When the actual outcome differs from expectation, the agent may need to retry, revise its plan, or restore a prerequisite before continuing.

Levels of integration. We organize tasks into three levels according to how capability demands are introduced and coordinated within an episode. The hierarchy begins with basic interaction, then progressively increases the extent to which capability demands must shape and coordinate task execution.

L0: Basic Interaction. L0 establishes the execution substrate of the benchmark. These tasks require the agent to complete basic closed-loop household interactions through the shared interface, without deliberately introducing a particular capability demand.

L1: Capability-Focused Tasks. Building on this substrate, L1 makes one capability demand from A, M, P, or V the primary object of diagnosis. Other abilities may still be needed to complete the task, so L1 is capability-focused rather than perfectly isolated.

L2: Structured Multi-Capability Tasks. L2 moves from a single primary demand to two or three capability demands that must be handled within the same episode. The defining property is not their mere co-occurrence: the task is constructed to make the demands interact and jointly shape task execution, rather than appearing as independent subtasks placed in sequence.

3.3 Controlled Task Construction

EmbodiedScope constructs capability-oriented tasks from executable household configurations rather than assigning capability labels after task generation. Each task family begins from a base configuration that specifies the scene, natural-language instruction, shared action interface, and canonical goal and verifier contract. Controlled variants modify a designated task-relevant condition while preserving the task goal, interaction setting, and non-target conditions where possible.

We introduce the four capability demands through corresponding changes to the task configuration. For Active Information Acquisition, information needed later in the task is initially unavailable and must be actively obtained through interaction. For Memory, task-relevant information is available earlier in the episode but later leaves the environment-provided context while remaining necessary for subsequent decisions. For Physical Reasoning, physical conditions such as reachability, containment, support, clearance, or dependencies between actions are changed so that the feasible course of action changes. For Verification and Recovery, the outcome of an action is changed so that subsequent behavior must depend on the resulting observation or public execution feedback.

When a task-relevant condition can be changed without altering the underlying task goal, we construct matched variants that differ in that condition while keeping the remaining task setting as similar as possible. This provides a controlled way to examine whether the intended capability demand changes how the agent must complete the task.

The unmodified base configurations provide L0 tasks. L1 tasks introduce one primary capability demand through the corresponding task modification. L2 tasks combine two or three demands within the same episode and are constructed so that the demands affect one another during execution. The resulting tasks are then checked for solvability, task quality, and whether the intended capability demands can be meaningfully exercised, as described in Sec. 3.4.

3.4 Task Validation

Before entering the final benchmark, each constructed task undergoes a two-stage validation procedure. First, a privileged oracle with access to simulator state executes the task from its specified initial state. A task is retained only if the oracle can complete it and the resulting world state is accepted by the canonical verifier. Tasks whose goals are already satisfied at the start, or whose reference execution cannot be completed reliably, are rejected. Privileged state is used only for task validation and is never exposed to evaluated agents.

Tasks that pass oracle validation are then manually reviewed for task quality. We remove tasks whose difficulty is dominated by excessive navigation, whose instructions or scene configurations are unnatural or ambiguous, or whose execution requirements are inconsistent with the intended benchmark setting. We also check for obvious shortcuts that would allow the task to be completed without exercising the intended capability demand. Only tasks that pass both the oracle execution check and the human quality review are included in the final benchmark.

3.5 Benchmark Composition

The final EmbodiedScope benchmark contains 240 tasks across three levels. L0 contains 100 basic-interaction tasks. L1 contains 120 capability-focused tasks, with 30 tasks for each of A, M, V, and P. L2 contains 20 structured multi-capability tasks.

4 EXPERIMENTAL SETUP

We evaluate MLLMs and agent-system configurations on EmbodiedScope under a common observation, action, and verification boundary. This section details the task execution protocol, evaluated agent systems, and metrics used throughout our experiments.

4.1 TASKS AND EVALUATION PROTOCOL

We evaluate every agent-system configuration on all 240 EmbodiedScope tasks: 100 L0 basic-interaction tasks, 120 L1 capability-focused tasks (30 each for A, M, P, and V), 20 L2 structured multi-capability tasks.

All evaluations follow the common interaction boundary defined in Sec. 3.1. Evaluated agents receive no privileged simulator state. Task success is determined from the resulting world state by the task-specific canonical verifier, which is shared across all agent-system configurations.

For the human baseline, we recruit three participants with prior robotics experience. Each participant independently attempts every task once, resulting in three human trials per task. We report both *Human (Avg.)*, the average success rate across the three trials, and *Human (Best-of-3)*, where a task is counted as successful if at least one of the three participants completes it successfully.

4.2 MODELS AND AGENT-SYSTEM CONFIGURATIONS

We evaluate eight representative multimodal backbones across ten agent-system configurations. GPT-6 Astra and GPT-5.6 Sol are each evaluated both in our Base Agent Configuration and with Codex. Claude Opus 5.5 is evaluated with Claude Code, while Gemini-3.8-Flash, DeepSeek-V4.1-Flash, GLM-5.3-Flash, Qwen3.8-Max, and Qwen3.8-27B are evaluated in the Base Agent Configuration. We treat each model–harness combination as a distinct agent system rather than attributing its performance solely to the underlying backbone. For all models, we use the highest available reasoning setting, except Claude Opus 5.5, which is evaluated at *High* to balance reasoning capability against inference latency and total evaluation time.

Base Agent Configuration. The Base Agent Configuration uses a shared lightweight harness that directly invokes the model API and manages the observation–decision–action loop. At each step, the model receives the task instruction, current public observation, environment-provided interaction context, and shared action schema. No additional agent-side memory, retrieval, or planning mechanism is introduced.

Agent Harnesses. These harnesses may provide their own context management and planning mechanisms, but interact with EmbodiedScope through the same observation and action boundary and receive no privileged simulator state or verifier information.

Comparison Protocol. All agent systems are evaluated on the same benchmark tasks with the same environment-side action semantics and canonical verifier. Because harnesses may differ in system instructions, context management, and planning support, we report each configuration separately. Direct harness-effect comparisons are made only for backbone-matched pairs for which both Base and harnessed configurations are available. Exact model versions, reasoning settings, and harness configurations are provided in the Appendix.

4.3 METRICS

We use **Task Success Rate (SR)** as our primary metric, defined as the percentage of tasks whose final world state is accepted by the canonical verifier. We report SR separately for L0, the four L1 capability subsets (A, M, P, and V), and L2.

We additionally report **Average Action Steps (AAS)**, the average number of action decisions recorded during task execution, to characterize interaction efficiency. We analyze the relationship between SR and AAS in the main text, while detailed step statistics are reported in the Appendix.

Table 2: **Overall performance across EmbodiedScope.** Task Success Rate (SR, %) is reported for $L0$, the four $L1$ capability subsets, and $L2$.

Model	L0 (100)	L1				L2 (20)
		L1-A (30)	L1-M (30)	L1-V (30)	L1-P (30)	
Task Success Rate (SR, %) ↑						
GPT-6 Astra	88.00	**90.00**	66.67	73.33	**86.67**	30.00
Codex + Codex	**95.00**	**90.00**	**90.00**	**83.33**	76.67	**55.00**
GPT-5.6 Sol	69.00	70.00	43.33	26.67	30.00	30.00
Codex + Codex	71.00	73.33	70.00	26.67	33.33	25.00
Opus 5.5 + Claude Code	89.00	**90.00**	86.67	56.67	60.00	35.00
Gemini-3.8-Flash	51.00	80.00	66.67	33.33	43.33	20.00
DeepSeek-V4.1-Flash	57.00	63.33	36.67	40.00	33.33	15.00
GLM-5.3-Flash	52.00	53.33	36.67	46.67	23.33	15.00
Qwen3.8-Max	47.00	63.33	33.33	50.00	53.33	25.00
Qwen3.8-27B	39.00	50.00	10.00	16.67	13.33	20.00
Human (Avg.)	91.00	82.22	85.56	91.11	81.11	93.33
Human (Best-of-3)	100.00	100.00	100.00	100.00	100.00	95.00

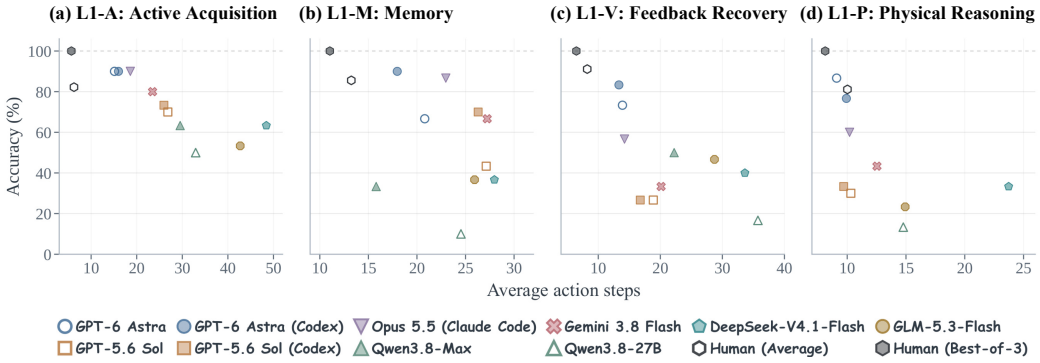

Figure 2: **Capability-level performance and execution efficiency on L1.** Task Success Rate (SR) is plotted against Average Action Steps (AAS) for the four capability-focused subsets.

5 Results

5.1 Overall and Capability-Level Performance

Table 2 reports the main results on L0, L1, and L2. GPT-6 Astra with Codex performs best, reaching 95.0% SR on L0 and 85.0% on average across the four L1 subsets. GPT-6 Astra and Opus 5.5 with Claude Code achieve 79.2% and 73.3% average L1 SR, respectively. Performance varies widely across the remaining systems, showing that strong basic interaction performance does not necessarily imply strong capability-level performance. On L2, GPT-6 Astra with Codex achieves the highest SR among the evaluated model–harness configurations at 55.0%, compared with its 85.0% average SR across the four L1 subsets.

Figure 2 further compares task success and execution efficiency across the four L1 capability subsets.

As shown in Figure 2, the four L1 subsets also reveal different strengths across agent systems. GPT-6 Astra performs better on A and P than on M and V. With Codex, its Memory score increases from 66.7% to 90.0%, and V also improves from 73.3% to 83.3%. Opus 5.5 with Claude Code shows a different pattern, with strong A and M performance but lower V and P. Human (Avg.) reaches 85.0% on average across L1, while Human (Best-of-3) reaches 100% on all L0 and L1 tasks. These results show that similar average scores can hide very different capability profiles. A task-level complement to these aggregate L1 results is provided in Table 6 in Appendix D, which reports success rates on one representative instance from each capability category.

5.2 Effect of Agent Harnesses

Agent harnesses provide substantial but selective gains. For GPT-6 Astra, Codex improves Memory from 66.7% to 90.0%, while GPT-5.6 Sol improves from 43.3% to 70.0%. In contrast, gains on other capabilities are smaller or inconsistent. This pattern is consistent with the stronger history management and recall mechanisms provided by native agent harnesses, suggesting that external context management can compensate for some memory-related failures. However, these gains do not transfer uniformly to structured multi-capability execution. On L2, Codex increases GPT-6 Astra's SR from 30.0% to 55.0%, whereas GPT-5.6 Sol decreases from 30.0% to 25.0%. This contrast suggests that harness support does not independently determine performance: its effectiveness also depends on the capabilities of the underlying model and on the demands imposed by the task. Harness effectiveness therefore reflects an interaction between external scaffolding, underlying model capability, and task structure rather than a model-independent improvement.

5.3 Behavioral Findings and Failure Analysis

To understand why agents fail, we inspect complete L0 and L1 trajectories from a subset of eight agentic-system configurations. Rather than reproducing the full error taxonomy in the main text, we highlight five recurring behavioral patterns.

Agents lose track of the evolving episode. As observations, actions, and execution receipts accumulate, models often lose a consistent account of the current world state and task progress. Their later plans can therefore rely on states that are outdated or no longer hold.

Agents show weak awareness of their own body and reachability. Models frequently misjudge their position, orientation, surrounding clearance, and distance to visible objects. This leads to collisions, repeated `out-of-reach` interactions, and difficulty leaving confined spaces after entering them.

Spatial errors propagate into later plans. An incorrect spatial interpretation can affect more than the current action. For example, misidentifying which object supports another can cause the agent to omit a necessary object or operation from its subsequent plan.

Action success is often mistaken for task completion. Agents may recover from explicit action failures, yet stop checking once an individual action returns a successful receipt. When the resulting goal state is changed or reverted later, they can terminate without noticing that the task is no longer complete.

Current appearance can override interaction history. In history-dependent tasks, agents sometimes choose an object based on its current appearance rather than its role in earlier interactions. A visually plausible choice can therefore be inconsistent with the historical reference required by the instruction.

6 Conclusion

To address the difficulty of diagnosing specific capability limitations and understanding interacting task demands from overall task success alone, we introduce EmbodiedScope, a diagnostic benchmark for embodied AI agents powered by general-purpose multimodal large language models. Built on BEHAVIOR, it combines controlled task variants with a hierarchy from capability-focused to integrated execution under a shared observation and action interface. It examines active information acquisition, memory, physical reasoning, and verification and recovery while abstracting fine-grained motor control. Experiments show that current models can perform basic interactions and capability-focused tasks but exhibit uneven capability profiles and struggle with multi-capability execution. Agent harnesses improve memory in the evaluated models, while gains in other capabilities and integrated execution vary across configurations. Trajectory analysis further reveals difficulties in maintaining task state, judging reachability, and verifying task completion. These findings highlight the need for agents to maintain an accurate account of environmental changes and use it to guide subsequent decisions. EmbodiedScope provides a common evaluation setting for studying these limitations and improving coordination among perception, memory, planning, and feedback.

AI USE STATEMENT

We used generative AI tools, including ChatGPT, to assist with manuscript drafting and editing, LaTeX formatting, and figure preparation. All AI-assisted content was reviewed and revised by the authors, who take full responsibility for the final manuscript.

ETHICS STATEMENT

This work does not involve the collection of personally identifiable information. EmbodiedScope is intended solely for research on embodied AI agents; benchmark performance should not be interpreted as evidence of safety or readiness for real-world deployment. The benchmark covers only a limited set of environments and tasks. All released assets and software will comply with their applicable licenses.

REPRODUCIBILITY STATEMENT

The main text describes the benchmark construction, interaction interface, evaluation protocol, and evaluated model configurations, with additional implementation details provided in Appendices C and B. The anonymous supplementary material includes task configurations, canonical verifier specifications, evaluation scripts, model prompts and action schemas, analysis scripts, and frozen experiment records.

For each evaluated system, we document the model identifier, access date, reasoning configuration, provider route, context policy, and action budget. Because several evaluated models and native agent harnesses are proprietary services that may change over time, exact output-level reproduction cannot always be guaranteed; we therefore preserve the configurations and episode records used to produce the reported results.

REFERENCES

Jixuan Chen, Jianzhi Shen, Haoqiang Kang, Zhi Hong, Qingyi Jiang, Soham Bose, Yiming Zhang, Leon Leng, Amit Vyas, Lingjun Mao, et al. Agentspec: Understanding embodied agent scaffolds through controlled composition. *arXiv preprint arXiv:2606.14674*, 2026a.

Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, et al. Rmbench: Memory-dependent robotic manipulation benchmark with insights into policy design. *arXiv preprint arXiv:2603.01229*, 2026b.

Yanzhe Chen, Zechen Bai, Zhijun Cao, Wenzheng Zeng, Kevin Qinghong Lin, Yiqi Lin, Guoqiang Liang, Kevin Yuchen Ma, Qiming Huang, and Mike Zheng Shou. Show-harness: Just a vlm agent can play robots. *arXiv preprint arXiv:2609.10522*, 2026c.

Yitong Chen, Zezheng Huai, Sixian Li, Yubang Wang, Haozhe Zhang, Yifei Zhang, Hechang Chen, Jingjing Gong, Yu-Gang Jiang, and Xipeng Qiu. Eta: A new agentic paradigm for embodied tasks. *arXiv preprint arXiv:2608.03924*, 2026d.

Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Dantong Niu, Fei-Fei Li, et al. Cap-x: A framework for benchmarking and improving coding agents for robot manipulation. *arXiv preprint arXiv:2603.22435*, 2026.

Hongcheng Gao, Hailong Qu, Jingyi Tang, Jiahao Wang, Zihao Huang, Hengkang Qiao, Shihong Huang, Junming Yang, Yi Li, Hongyixuan Yuan, et al. Spatialworld: Benchmarking interactive spatial reasoning of multimodal agents in real-world tasks. *arXiv preprint arXiv:2606.09669*, 2026.

Atharva Gundawar, Som Sagar, and Ransalu Senanayake. Pac bench: Do foundation models understand prerequisites for executing manipulation policies? *Advances in Neural Information Processing Systems*, 38, 2026.

Songhao Han, Boxiang Qiu, Yue Liao, Siyuan Huang, Chen Gao, Shuicheng Yan, and Si Liu. Robocerebra: A large-scale benchmark for long-horizon robotic manipulation evaluation. *Advances in Neural Information Processing Systems*, 38, 2026.

Yining Hong, Jiageng Liu, Han Yin, Manling Li, Leonidas Guibas, Li Fei-Fei, Jiajun Wu, and Yejin Choi. Esi-bench: Towards embodied spatial intelligence that closes the perception-action loop. *arXiv preprint arXiv:2605.18746*, 2026.

Jinbang Huang, Yuanzhao Hu, Zhiyuan Li, Ran Qi, Yixin Xiao, Zhanguang Zhang, Mark Coates, Tongtong Cao, and Yingxue Zhang. Roboharness: Memory-driven orchestration of heterogeneous robot policies for long-horizon planning. *arXiv preprint arXiv:2607.18060*, 2026.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li E Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking llms for embodied decision making. *Advances in Neural Information Processing Systems*, 37:100428–100534, 2024.

Siyao Li, Jiawei Gu, Shuai Liu, Kairui Hu, Zekun Li, Linjie Li, Chengcheng Tang, Po-Chen Wu, Ivan Shugurov, Lingni Ma, et al. Humanclaw: Can vision-language models act through a body? *arXiv preprint arXiv:2607.27180*, 2026.

Haowen Liu, Xirui Li, Shaoxiong Yao, Peng Shi, Tianyi Zhou, Jia-Bin Huang, Furong Huang, and Jiayuan Mao. Guava: An effective and universal harness for embodied manipulation. *arXiv preprint arXiv:2606.18363*, 2026a.

Lin Liu, Zhicheng Bao, Lu Zhang, Ziying Song, Wu Yang, Shuai Tao, Wulong Liu, and Huchuan Lu. Libero-recover: Beyond task success towards failure recovery in robotic manipulation models. *arXiv preprint arXiv:2609.05178*, 2026b.

Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Song XiXuan, Yifan Xu, Shudan Zhang, Hanyu Lai, Jiadai Sun, Xinyue Yang, et al. Visualagentbench: Towards large multimodal models as visual foundation agents. In *International Conference on Learning Representations*, volume 2025, pp. 95650–95707, 2025.

Yiwei Liu, Luwei Yang, and Shunbo Lei. One word, different action: A real-robot benchmark for language-conditioned embodied reasoning. *arXiv preprint arXiv:2609.05260*, 2026c.

Boyu Mi, Mengchen Ma, Yifei Yao, Xing Gao, Junting Chen, Yangzi Li, Zihou Zhu, Guohao Li, Zhenfei Yin, Tai Wang, et al. Exploratory, communicative, and deployable: Vision-driven embodied agents for open-world mobile manipulation. In *European Conference on Computer Vision*, pp. 627–645. Springer, 2026.

Yu Qi, Haibo Zhao, Ziyu Guo, Siyuan Ma, Ziyan Chen, Yaokun Han, Renrui Zhang, Zitiantao Lin, Shiji Xin, Yijian Huang, et al. Bear: Benchmarking and enhancing multimodal language models for atomic embodied capabilities. *arXiv e-prints*, pp. arXiv–2510, 2025.

Andrea Tupini, Lars Liden, Reuben Tan, Yu Wang, and Jianfeng Gao. Asgardbench–evaluating visually grounded interactive planning under minimal feedback. *arXiv preprint arXiv:2603.15888*, 2026.

Meiqi Wang and Shichao Li. Compiling and benchmarking task-state horizons for embodied agents. *arXiv preprint arXiv:2608.08036*, 2026.

Karmesh Yadav, Yusuf Ali, Gunshi Gupta, Yarin Gal, and Zsolt Kira. Findingdory: A benchmark to evaluate memory in embodied agents. *arXiv preprint arXiv:2506.15635*, 2025.

Bowen Yan, Jiahao Xiao, Kehui Liu, Jianbo Zhang, Zicheng Zhang, Qi Jia, Zhongjie Jia, Haoming Song, Chunyi Li, Bin Zhao, and Guangtao Zhai. Can VLMs diagnose and recover from VLA manipulation faults? In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Q0rAQZg9t5.

Huaigang Yang, Ya Li, Min Ren, Bo Dai, Zhenliang Zhang, and Zhaofeng He. Saferelbench: A spatial-relation-aware benchmark for process-level safety in vlm-driven embodied agents. *arXiv preprint arXiv:2607.14543*, 2026.

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. *arXiv preprint arXiv:2502.09560*, 2025.

Pingyue Zhang, Zihan Huang, Yue Wang, Jieyu Zhang, Letian Xue, Zihan Wang, Qineng Wang, Keshigeyan Chandrasegaran, Ruohan Zhang, Yejin Choi, et al. Theory of space: Can foundation models construct spatial beliefs through active exploration? In *The Fourteenth International Conference on Learning Representations*, 2026a.

Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, et al. Harness vla: Steering frozen vlas into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*, 2026b.

Zilin Zhu, Longteng Guo, Yanghong Mei, Bowen Pang, Zongxun Zhang, Xingjian He, Ruyi Ji, and Jing Liu. When robots do the chores: A benchmark and agent for long-horizon household task execution. *arXiv preprint arXiv:2605.14504*, 2026.

APPENDIX OUTLINE

A Benchmark Construction and Validation Details . . . 13
A.1 Controlled Capability Variants . . . 13
A.2 Capability Integration Across Levels . . . 14
A.3 Task Validation . . . 14
A.4 Benchmark Composition . . . 14
B Action20 Interface and Execution Semantics . . . 14
B.1 Public Observation and Interaction Boundary . . . 14
B.2 Model-Facing Action Protocol . . . 15
B.3 Navigation and Viewpoint Control . . . 15
B.4 Temporary Visual References and Object Selection . . . 17
B.5 Placement Actions and Preconditions . . . 17
B.6 Public Execution Receipts and State-Change Actions . . . 18
B.7 Episode Control and Verification . . . 20
C Evaluation Harnesses and Reproducibility . . . 20
C.1 Evaluation Settings and Shared Contract . . . 22
C.2 Common Harness . . . 22
C.3 Native Agent Harnesses . . . 23
C.4 Frozen Configuration Records . . . 23
C.5 Decision Accounting and Protocol Handling . . . 23
D Additional Quantitative Results . . . 23
D.1 Execution Efficiency . . . 23
E Representative Episode Trajectories . . . 28

A BENCHMARK CONSTRUCTION AND VALIDATION DETAILS

This appendix expands on the construction and validation procedures used for EmbodiedScope. Tasks are built from executable household configurations rather than being assigned capability labels after generation. Each task family starts from a base configuration that specifies the scene, natural-language instruction, shared action interface, and canonical goal and verifier contract. Controlled variants then modify a designated task-relevant condition while preserving the underlying task goal, interaction setting, and non-target conditions where possible.

A.1 CONTROLLED CAPABILITY VARIANTS

Active Information Acquisition. An Active Information Acquisition (A) variant withholds information that is needed for a later decision. The agent must obtain the missing evidence through interaction, for example by changing viewpoint, moving to a new position, or inspecting another part of the scene, and then use that evidence to guide its subsequent actions.

Memory. A Memory (M) variant makes task-relevant information available earlier in the episode but requires it again only after it is no longer directly present in the current observation or the environment-provided recent interaction context. The agent must retain and later use this information while the task state continues to change.

Physical Reasoning. A Physical Reasoning (P) variant changes a physical condition that affects the feasible course of action. Relevant conditions include reachability, containment, support, clearance, articulation, and dependencies between actions. The task goal is preserved, but the agent must account for the changed physical constraints when selecting an action or plan.

Verification and Recovery. A Verification and Recovery (V) variant changes an execution outcome so that the agent must inspect the resulting observation or public execution receipt before continuing. When the actual state differs from the expected state, the agent may need to retry an action, revise its plan, or restore a prerequisite.

When a task-relevant condition can be changed without altering the task goal, we construct matched variants that differ in that condition while keeping the remaining task setting as similar as possible. This supports capability-focused diagnosis within a shared execution setting.

A.2 Capability Integration Across Levels

EmbodiedScope organizes tasks by increasing integration of capability demands.

L0: Basic Interaction. L0 tasks establish the execution substrate of the benchmark. They require basic closed-loop household interactions through the shared interface without deliberately introducing a particular capability demand.

L1: Capability-Focused Tasks. L1 tasks make one demand from A, M, P, or V the primary object of diagnosis. Other abilities may still be required, so L1 is capability-focused rather than perfectly isolated.

L2: Structured Multi-Capability Tasks. L2 tasks combine two or three capability demands within the same episode. The demands are constructed to interact and jointly shape task execution rather than appearing only as independent subtasks placed in sequence.

A.3 Task Validation

Each task undergoes a two-stage validation procedure before inclusion in the benchmark. First, a privileged oracle with access to simulator state executes the task from the specified initial state. A task is retained only if the oracle can complete it and the resulting world state is accepted by the canonical verifier. Tasks whose goals are already satisfied at initialization, or whose reference executions cannot be completed reliably, are rejected. Privileged state is used only for task validation and is never exposed to evaluated agents.

Tasks that pass oracle validation are then manually reviewed for task quality. The review removes tasks whose difficulty is dominated by excessive navigation, whose instructions or scene configurations are unnatural or ambiguous, whose execution requirements are inconsistent with the intended benchmark setting, or whose construction admits an obvious shortcut that bypasses the intended capability demand. Only tasks that pass both oracle validation and manual review are included in the final benchmark.

A.4 Benchmark Composition

The final benchmark contains 240 tasks: 100 L0 basic-interaction tasks, 120 L1 capability-focused tasks with 30 tasks for each of A, M, V, and P, 20 L2 structured multi-capability tasks.

B Action20 Interface and Execution Semantics

This appendix specifies the public interaction boundary, model-facing action protocol, target-selection mechanism, execution receipts, and termination semantics used throughout EmbodiedScope.

B.1 Public Observation and Interaction Boundary

All evaluated systems interact with EmbodiedScope through the same public environment interface. In the default visual setting, each observation is a three-view RGB composite consisting of one head-camera view and two wrist-camera views. The environment supplies the current observation together with the public interaction record from the five most recent completed steps, including their observations, actions, and execution receipts.

Environment-side evaluation state is kept separate from the agent-visible interface. Depth, semantic segmentation, object poses, scene graphs, hidden verifier predicates, private diagnostic state, future outcomes, and other privileged simulator state are not exposed to the agent. Historical recall, when

Table 3: **Model-facing action protocol.** Each call specifies one action; target selection proceeds through sequential calls using the latest observation-bound C-markers.

Component	Specification or example
Model input	Task instruction, onboard RGB observations, previous actions, and public results.
Expected output	One `robot_action` call with JSON arguments. The model waits for its result before the next call; actions are not batched, and plans or reasoning are not included in the arguments.
Translation	{"action":"move_forward", "move_distance_m":0.30, "repeat":2}
Body rotation	{"action":"turn_left", "turn_angle_deg":30}
Candidate request	{"action":"pick"}
Target selection	{"action":"pick","target":3} Select C3 from the latest same-action selection stage.
Staged placement	{"action":"place_inside"} Select a container; if another `selection_required` response is returned, select a compartment or region using its new labels.
No target required	{"action":"place_on_floor"} or {"action":"wait"}.
Termination	{"action":"done"} End the episode; this does not assert verifier success.
Public response	`selection_required`, `completed`, or `blocked`, with observations and applicable metadata such as a blocked category, body-relative motion, and remaining decision budget.

Note. All same-action marker-selection stages count as one action decision. The optional `target_description` is an audit note; only the current C-marker resolves the target.

available through an evaluated harness, is restricted to information that was already made public earlier in the same episode.

Environment-changing actions are issued through Action20. For object-directed interactions, the agent may first request visual candidates and then select a temporary C-marker from the current observation. Marker labels are local to the corresponding observation and selection stage and cannot be used as persistent object identifiers. The appearance of a candidate does not imply that the object is reachable or that the requested physical interaction will succeed; visual selection and physical execution are evaluated separately.

Actions are executed sequentially. Completion of an individual action does not imply completion of the task. Final task success is determined only by the environment-side verifier defined by the frozen task contract.

B.2 Model-Facing Action Protocol

The model-facing input, expected output, and public responses are summarized in Table 3. Examples show the JSON arguments of a single `robot_action` call.

B.3 Navigation and Viewpoint Control

Translations are expressed in the robot's current body frame, with `move_distance_m` chosen from $\{0.15, 0.30, 0.60\}$ m (default: 0.15 m). Body turns use `turn_angle_deg`$\in \{10, 30\}^\circ$ (default: 10°). The interface decomposes larger commands into 0.15 m translation or 10° rotation substeps. Camera actions move one pitch level among down, center, and up, approximately 15° apart; a request beyond the available range is blocked.

Translation examples are shown for `move_forward` (Figure 3), `move_backward` (Figure 4), `move_left` (Figure 5), and `move_right` (Figure 6). Body-rotation examples are shown for

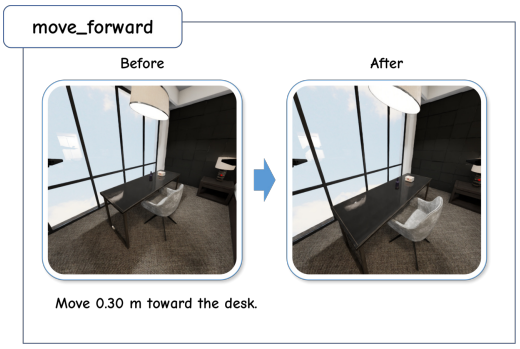

Figure 3: **Forward Translation.** The robot moves 0.30 m toward the desk.

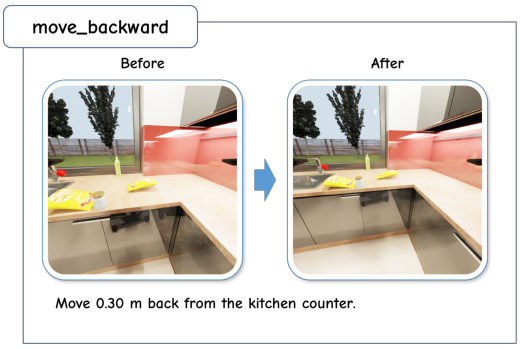

Figure 4: **Backward Translation.** The robot moves 0.30 m away from the kitchen counter.



Figure 5: **Leftward Translation.** The robot moves 0.60 m left past the dining table.



Figure 6: **Rightward Translation.** The robot moves 0.60 m right in the living room.



Figure 7: **Left Body Rotation.** The robot turns 30° left toward the kitchen counter.



Figure 8: **Right Body Rotation.** The robot turns 30° right toward the dining area.

`turn_left` (Figure 7) and `turn_right` (Figure 8); camera-pitch examples are shown for `look_up` (Figure 9) and `look_down` (Figure 10).

Repeated navigation. `repeat`$\in \{1, \ldots, 10\}$ repeats a body translation or turn (default: one), with each constituent substep executed separately. For example, a 0.30 m move with `repeat=3` requests six 0.15 m substeps. Execution stops on a blocked or terminal result, so the requested displacement is not guaranteed. The agent cannot intervene within a repeated command; returned steps preserve their order and report measured body-relative motion. A repeated command counts as one action decision.

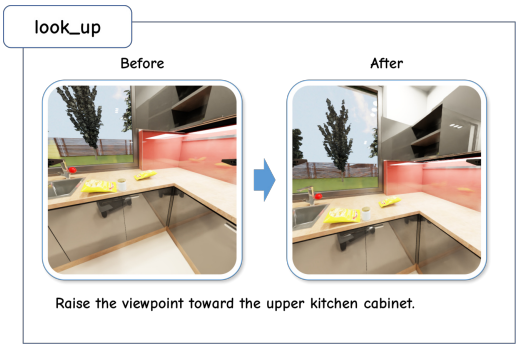

Figure 9: **Upward Viewpoint Adjustment.** The viewpoint is raised toward the upper kitchen cabinet.



Figure 10: **Downward Viewpoint Adjustment.** The viewpoint is lowered toward the outdoor coffee table.



Figure 11: **Object Acquisition with Target Selection.** The agent selects the left paper bag (C4) from the current observation and completes the `pick` action.

B.4 Temporary Visual References and Object Selection

For an object-directed action, the agent first omits `target` to request candidates. A `selection_required` response presents current-observation C-markers; the agent continues with the same action and an integer target, for example `target=3` for C3. Labels are temporary and valid only for the latest selection stage. Object names and persistent simulator identifiers do not resolve targets; an optional `target_description` is an audit note only. Figure 11 illustrates this process for `pick`.

B.5 Placement Actions and Preconditions

`place_on_top` selects a support and then a surface; `place_next_to` selects a reference and then a position. `place_inside` selects a container and, when requested, a compartment or region. `place_under` selects the upper reference, whereas `place_on_floor` requires no target.

Placement moves the object already held in the left hand. `pick`, `open`, `close`, and device toggles require that hand to be empty. A visible marker does not guarantee reachability or feasible placement. All same-action selection stages count as one decision. A new request without a target cancels pending selection; restarting it consumes a new decision.

Examples are shown for `place_on_floor` (Figure 12), `place_inside` (Figure 13), `place_on_top` (Figure 14), `place_next_to` (Figure 15), and `place_under` (Figure 16).

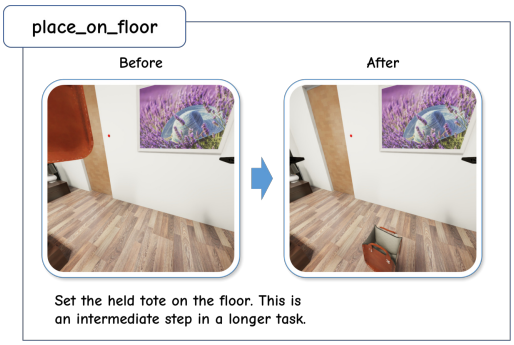

Figure 12: **Floor Placement.** The held tote is placed directly on the floor without a target-selection stage.

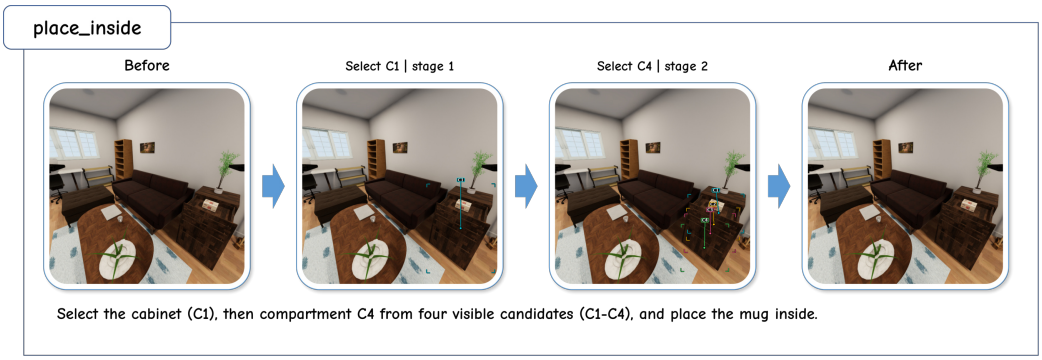

Figure 13: **Two-Stage Inside Placement.** The agent first selects cabinet C1, then compartment C4, and places the held mug inside.



Figure 14: **Two-Stage On-Top Placement.** The agent selects the outdoor coffee table (C8), then a tabletop site (C5), and places the plate on the selected surface.

B.6 Public Execution Receipts and State-Change Actions

A `completed` receipt confirms an action, not task success. A `blocked` receipt may include a coarse `blocked_category` describing reachability, unavailable candidates or selections, hand-state requirements, placement fit, support, collision, or relation failures, articulated obstruction, navigation blockage, or view restrictions. An absent category leaves the cause unspecified. In particular, `execution_unverified` means that the scene may already have changed, requiring inspection before retrying.

Blocked actions need not be state-preserving: symbolic articulated interactions can retain safe partial motion. Public feedback provides neither task scores nor task-progress estimates. Examples of

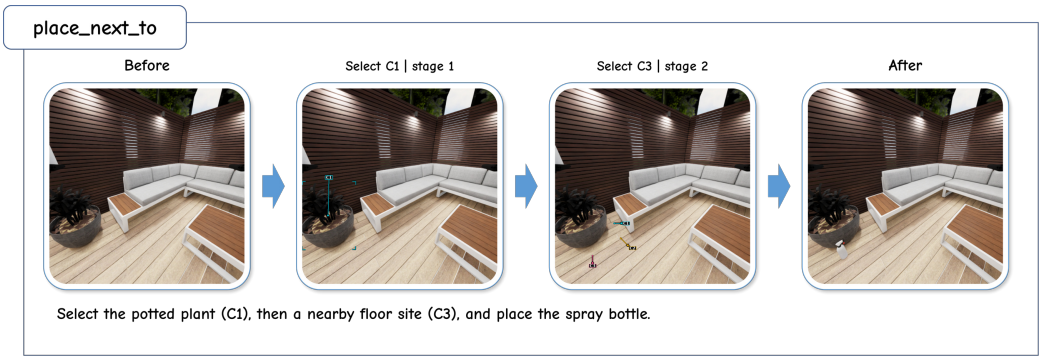

Figure 15: **Two-Stage Next-To Placement.** The agent selects the potted plant (C1), then a nearby floor site (C3), and places the spray bottle next to it.

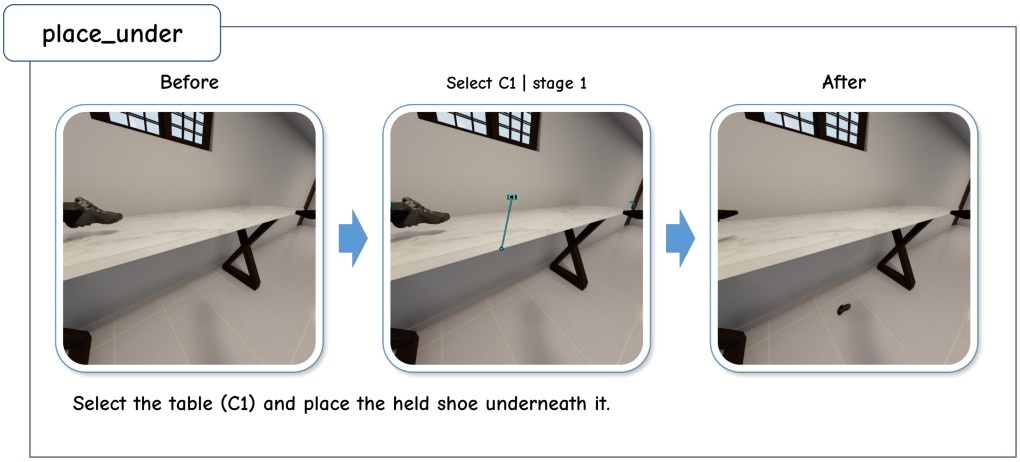

Figure 16: **Under-Object Placement.** The agent selects the table (C1) and places the held shoe underneath it.

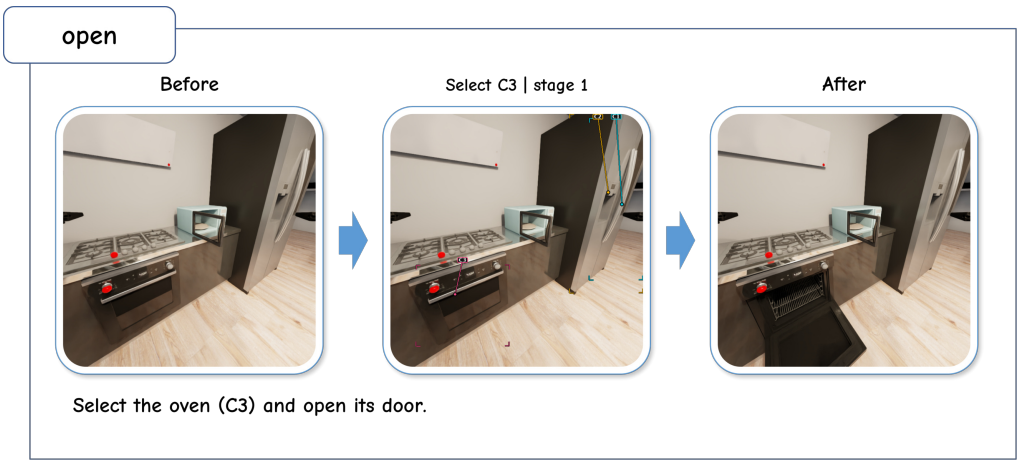

Figure 17: **Opening an Articulated Object.** The agent selects the oven (C3) and opens its door.

`open`, `close`, `toggle_on`, and `toggle_off` are shown in Figures 17, 18, 19, and 20, respectively.

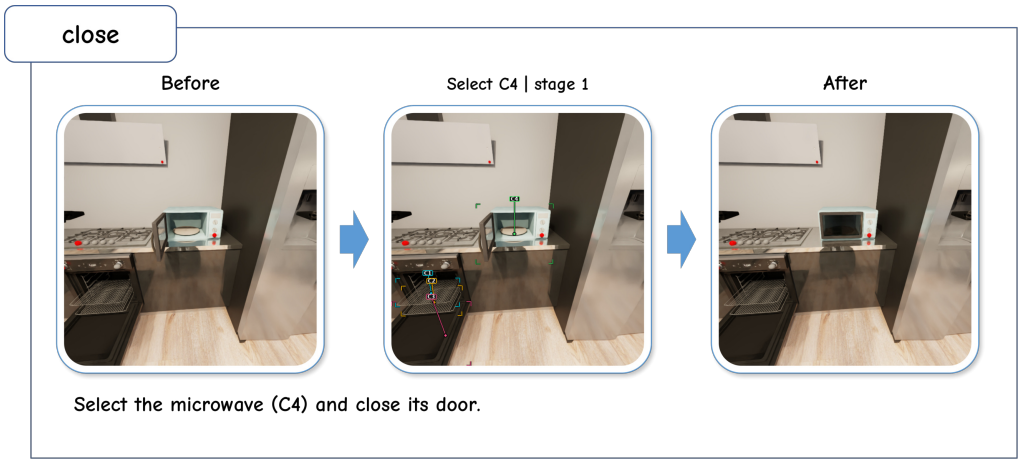

Figure 18: **Closing an Articulated Object.** The agent selects the microwave (C4) and closes its door.

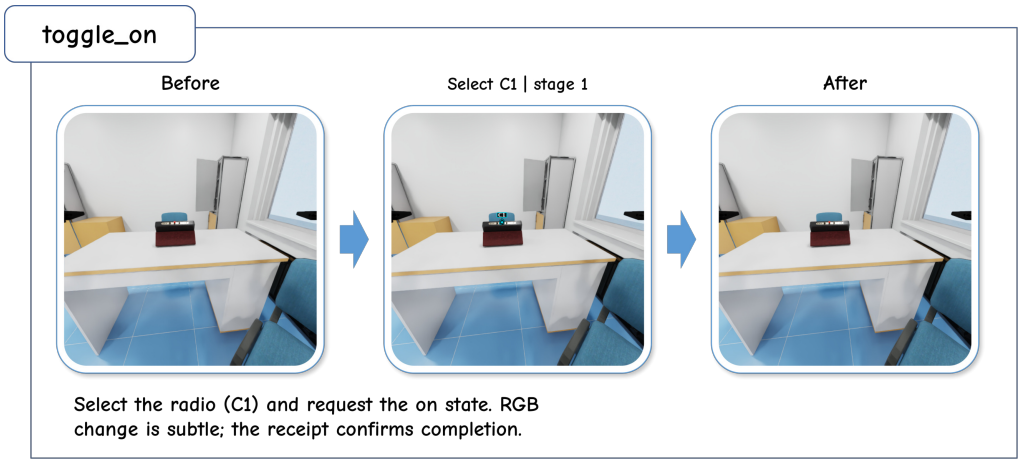

Figure 19: **Device Activation.** The agent selects the radio (C1) and requests the on state. The visual change is subtle, while the action receipt confirms completion.

B.7 Episode Control and Verification

The default Action20 Symbolic profile uses kinematic or state updates together with action-specific geometric and state checks while physics stepping is disabled. It therefore abstracts low-level motor execution. The `wait` action refreshes the observation without advancing physics in this profile.

The implementation also contains an experimental Physical branch for inside-placement. This branch checks an entry path, releases the object, advances eight simulation steps, and reads back the resulting state. It is disabled by default and does not constitute a full low-level motor-control interface.

The `done` action requests episode termination. Success is then determined by the canonical world-state verifier together with any process- or history-dependent requirements declared by the task. Figures 21 and 22 illustrate observation refresh and episode termination.

C Evaluation Harnesses and Reproducibility

This appendix describes the evaluation harnesses used to connect multimodal models and agent systems to EmbodiedScope. Unless otherwise noted, all configurations share the same benchmark

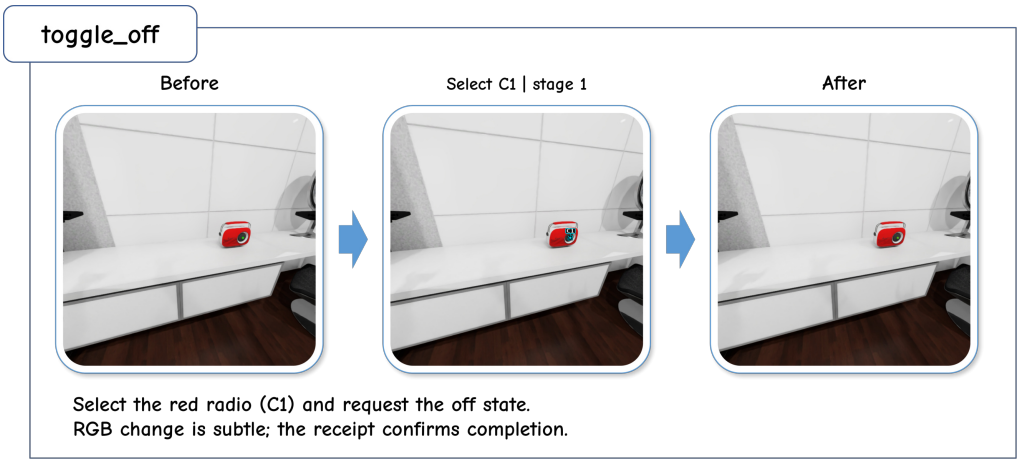

Figure 20: **Device Deactivation.** The agent selects the red radio (C1) and requests the off state. The visual change is subtle, while the action receipt confirms completion.

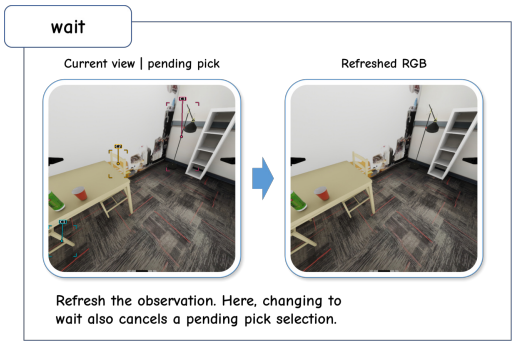

Figure 21: **Observation Refresh.** The `wait` action refreshes the observation; in this example it also cancels a pending `pick` selection.

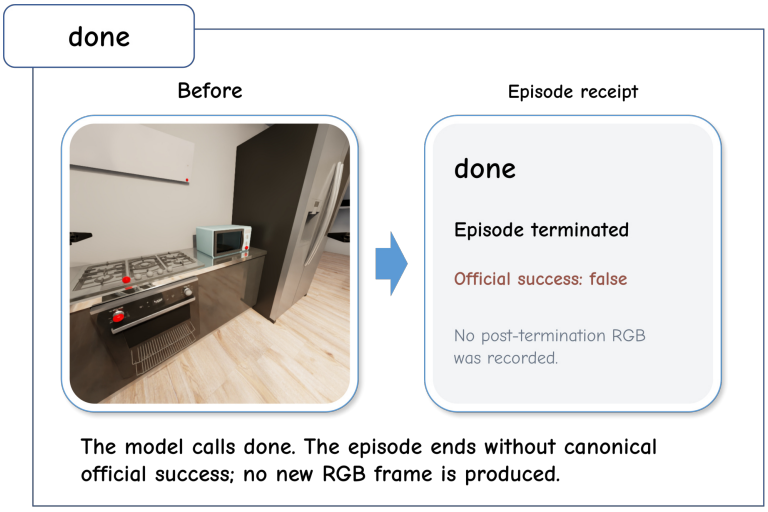

Figure 22: **Episode Termination.** The `done` action terminates the episode and returns an episode-level receipt. No post-termination RGB frame is produced in this example.

environment, frozen task contracts, Action20 interaction semantics, and environment-side verifier.

Table 4: **Comparison of evaluation harnesses.** All configurations share the EmbodiedScope environment and Action20 task semantics, while model interaction, context management, and native planning mechanisms may differ.

Component	Common Harness	Codex	Claude Code
Model interaction	Direct model API	Official Codex CLI	Official Claude Code CLI
Base instructions	Benchmark-constructed instructions	Native instructions + benchmark additions	Native instructions + benchmark additions
Robot interface	Action20 tool interface	Action20 MCP	Action20 MCP
Episode loop	Managed by the evaluation client	Managed by the CLI session	Managed by the CLI session
History management	Benchmark-constructed history	Native context + public-history recall	Native context + public-history recall
Native planning	No additional CLI planning tools	Codex native planning facilities	Claude Code task-management facilities
Privileged simulator state	No	No	No
Environment / verifier	EmbodiedScope	EmbodiedScope	EmbodiedScope

Different harnesses may nevertheless differ in system instructions, context management, transport format, and native planning support; each model–harness combination is therefore reported as a distinct agent system.

C.1 Evaluation Settings and Shared Contract

We evaluate two types of agent-system configuration. In the **Common Harness**, the evaluation client directly invokes a model API and manages the observation–decision–action loop. In the **native-harness setting**, an official agent CLI manages model interaction and its native planning mechanisms, while an integration layer connects the CLI to EmbodiedScope through Action20. The native-harness setting is instantiated with Codex and Claude Code.

These settings answer related but different questions. The Common Harness provides a largely shared execution framework for comparing model backbones, subject to provider-specific API adaptations. Native-harness configurations measure the complete model–harness system. Differences between the two settings are therefore not attributed solely to the underlying model.

The term *Base Agent Configuration* does not imply the absence of all scaffolding. The Common Harness still constructs prompts and interaction history, provides the action schema, validates proposed tool calls, and manages bounded protocol recovery when enabled.

The harnesses share task instructions, Action20 semantics, and the environment contract, but they do not necessarily receive byte-identical model inputs. Native system instructions, transport formats, image representations, context retention, and adapter-specific instructions may differ. We therefore distinguish a shared environment contract from identical prompts or identical effective context.

C.2 Common Harness

The Common Harness is implemented as a model-independent episode client. At each decision step, it constructs a model request from the task instruction, current public observation, retained interaction history, and Action20 tool schema. The model response is parsed and validated before any action is dispatched. A valid action is then executed in the environment, after which the public result and the next observation are recorded and used to construct the following decision step.

The Common Harness does not add an external memory, retrieval, or planning module beyond its prompt construction, history formatting, tool validation, and episode-loop responsibilities. In the Base Agent Configuration, the environment-provided recent context constitutes the model's available episode history unless the model itself retains information within its response context.

C.3 Native Agent Harnesses

Native-harness configurations retain the official CLI's context-management and planning mechanisms while using the same public EmbodiedScope interface. The integration layer exposes Action20 through MCP and does not provide access to privileged simulator state, hidden verifier predicates, or future outcomes.

Codex. GPT-6 Astra and GPT-5.6 Sol are additionally evaluated through the official Codex CLI. The CLI manages the model session, native instructions, context, and planning facilities, while the integration layer translates valid Action20 MCP calls into environment actions and returns the corresponding public observations and receipts.

Claude Code. Claude Opus 5.5 is evaluated through the official Claude Code CLI. Claude Code manages its native session context and task-management facilities, while the integration layer connects the CLI to the same Action20 MCP interface and environment-side verifier.

Because native harnesses can organize or retrieve information that was already made public earlier in the episode, they may provide different effective context from the Common Harness. They nevertheless receive no additional environment state.

C.4 Frozen Configuration Records

Each reported run is associated with frozen experiment artifacts recording the agent-system name, model identifier, provider route, harness, reasoning and context settings, and resource limits used for that run. These records are the source of the model-system labels and quantitative results reported in the paper.

C.5 Decision Accounting and Protocol Handling

A model response is parsed and validated before execution. Same-action marker-selection stages count as one action decision, and a repeated navigation command also counts as one decision even though it may contain multiple environment substeps. Repeated execution stops when a substep is blocked or terminal. A `completed` receipt confirms only the requested action; it does not indicate task success. The `done` action terminates the episode, after which the canonical verifier determines success from the final world state and any process- or history-dependent task requirements.

D Additional Quantitative Results

D.1 Execution Efficiency

Table 5 reports Average Action Steps (AAS), computed from the recorded action-decision counts for each evaluated setting. Lower values indicate fewer action decisions on average. Because step counts alone do not indicate whether a task was completed successfully, AAS should be interpreted together with Task Success Rate rather than as a standalone measure of system performance.

D.2 Performance on Representative L1 Instances

Table 6 complements the aggregate capability-level results by reporting Task Success Rate on four representative L1 instances, one from each capability category.

The following figures show three independent successful executions of the same container-completion task by GPT-6 Astra. The agent must identify the container that already holds two of the three target items—shrimp, pea pod, and broccoli—and place the missing item inside. The runs reach the same goal state through different inspection and viewpoint-adjustment strategies.

Table 5: **Average action steps across EmbodiedScope.** AAS reports the mean number of recorded action decisions for each evaluated setting; lower values indicate fewer action decisions. Bold and underlined values denote the lowest and second-lowest agent-system results within each column, respectively. A dash indicates that a result is not available.

Model	**L0** (100)	**L1**				**L2** (20)
		L1-A (30)	**L1-M** (30)	**L1-V** (30)	**L1-P** (30)	
GPT-6 Astra	9.82	**15.13**	20.80	<u>13.90</u>	**9.10**	47.65
Codex + Codex	9.54	<u>16.00</u>	<u>17.97</u>	**13.33**	9.93	<u>35.75</u>
GPT-5.6 Sol	12.33	26.83	27.13	18.87	10.31	73.20
Codex + Codex	13.30	25.97	26.30	16.77	<u>9.70</u>	64.65
Opus 5.5 (Claude Code)	**8.98**	18.60	22.97	14.23	10.20	**28.79**
Gemini-3.8-Flash	<u>9.15</u>	23.47	27.23	20.13	12.53	54.10
DeepSeek-V4.1-Flash	17.01	48.43	27.97	33.63	23.73	48.90
GLM-5.3-Flash	15.17	42.70	25.93	28.73	14.93	57.05
Qwen3.8-Max	16.60	29.53	**15.80**	22.23	12.90	44.90
Qwen3.8-27B	16.14	32.93	24.53	35.73	14.77	68.25
Human	7.32	6.24	13.24	8.22	10.02	55.80

Table 6: **Success rates on representative L1 capability instances.** Task Success Rate (SR, %) is reported for four selected capability-specific instances: Active Information Acquisition (A1), Physical Reasoning (P1), Memory (M1), and Verification and Recovery (V1). Bold and underlined values denote the best and second-best agent-system results within each column, respectively. The shaded row denotes a native-harness configuration. Each value aggregates three independent trials of the selected instance.

Model	**Representative L1 Capability Instances**			
	A1	**P1**	**M1**	**V1**
Codex GPT-6 Astra + Codex	**100.0**	**100.0**	**100.0**	**100.0**
Gemini-3.8-Flash	**100.0**	<u>66.7</u>	<u>66.7</u>	33.3
GPT-5.6 Sol	**100.0**	<u>66.7</u>	**100.0**	**100.0**
GPT-6 Astra	<u>66.7</u>	<u>66.7</u>	<u>66.7</u>	<u>66.7</u>
Qwen3.8-Max	33.3	<u>66.7</u>	33.3	33.3

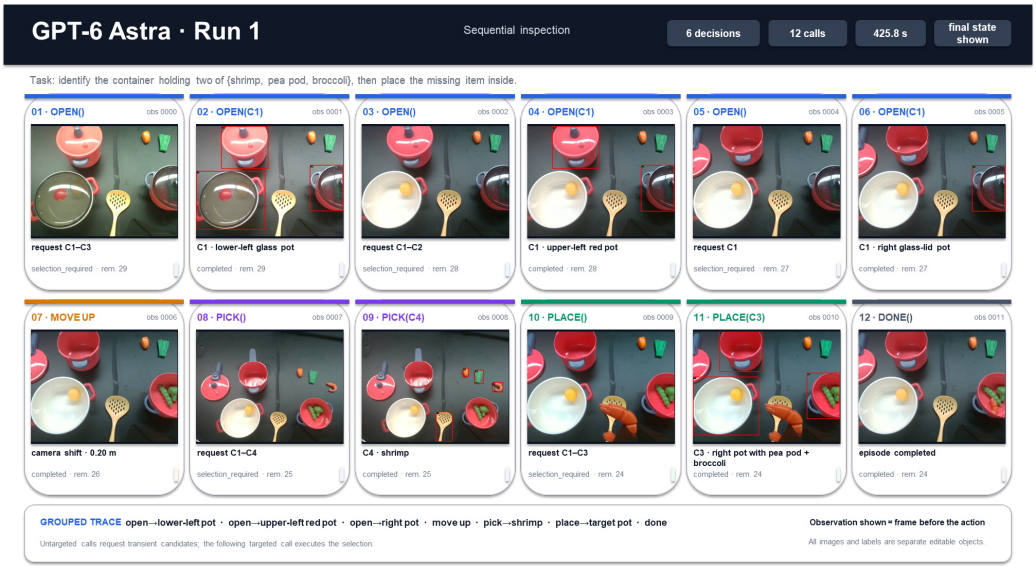

Figure 23: **GPT-6 Astra Run 1: Sequential Inspection of All Candidate Containers.** The agent sequentially opens and inspects the lower-left glass pot, the upper-left red pot, and the right glass-lid pot. After determining that the right-hand pot contains the pea pod and broccoli, it identifies the shrimp as the missing item, adjusts the viewpoint, picks up the shrimp, and places it into the target pot. The run successfully completes the task using six action decisions and twelve model–environment calls.

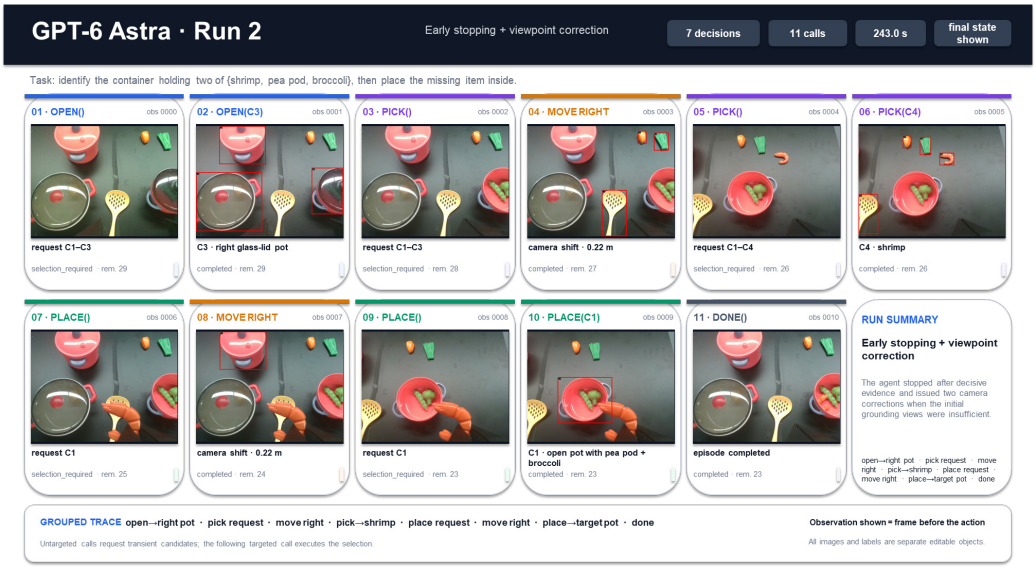

Figure 24: **GPT-6 Astra Run 2: Early Stopping with Viewpoint Correction.** The agent first opens the right-hand pot and observes that it contains the pea pod and broccoli. This provides sufficient evidence to infer that the shrimp is the missing item, allowing the agent to stop inspecting the remaining containers. It then performs lateral viewpoint corrections, picks up the shrimp, and places it into the target pot. The run successfully completes the task using seven action decisions and eleven model–environment calls.

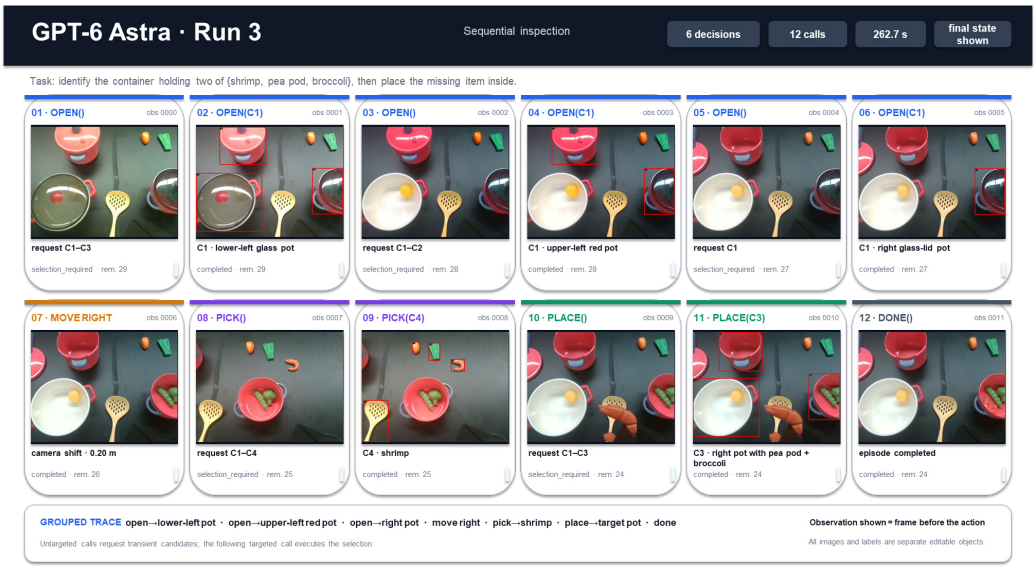

Figure 25: **GPT-6 Astra Run 3: Sequential Inspection with Lateral Viewpoint Adjustment.** The agent again inspects all three candidate containers before acting on the missing item. After identifying the right-hand pot as the container holding the pea pod and broccoli, it moves laterally to obtain a clearer grounding view, picks up the shrimp, and places it into the target pot. The run successfully completes the task using six action decisions and twelve model–environment calls.

E REPRESENTATIVE EPISODE TRAJECTORIES

This appendix presents representative trajectories from the basic-interaction and capability-focused task settings. Each trajectory shows the public visual observations, selected actions, temporary C-markers when applicable, and public execution outcomes available to the evaluated agent.



Figure 26: **L0 Basic Interaction: Closing the Microwave.** The agent requests candidates for `close`, selects the microwave (C4), and closes it while leaving the oven open.

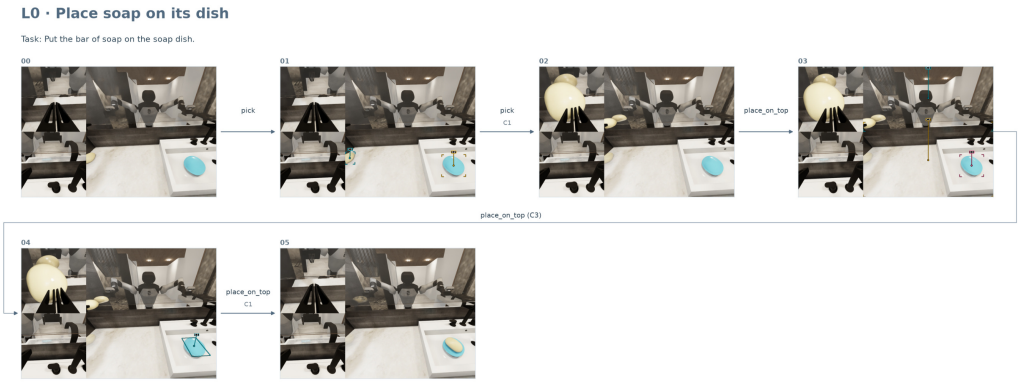

Figure 27: **L0 Basic Interaction: Placing Soap on Its Dish.** The agent selects and picks up the bar of soap, identifies the soap dish as the placement target, and places the soap on the dish.

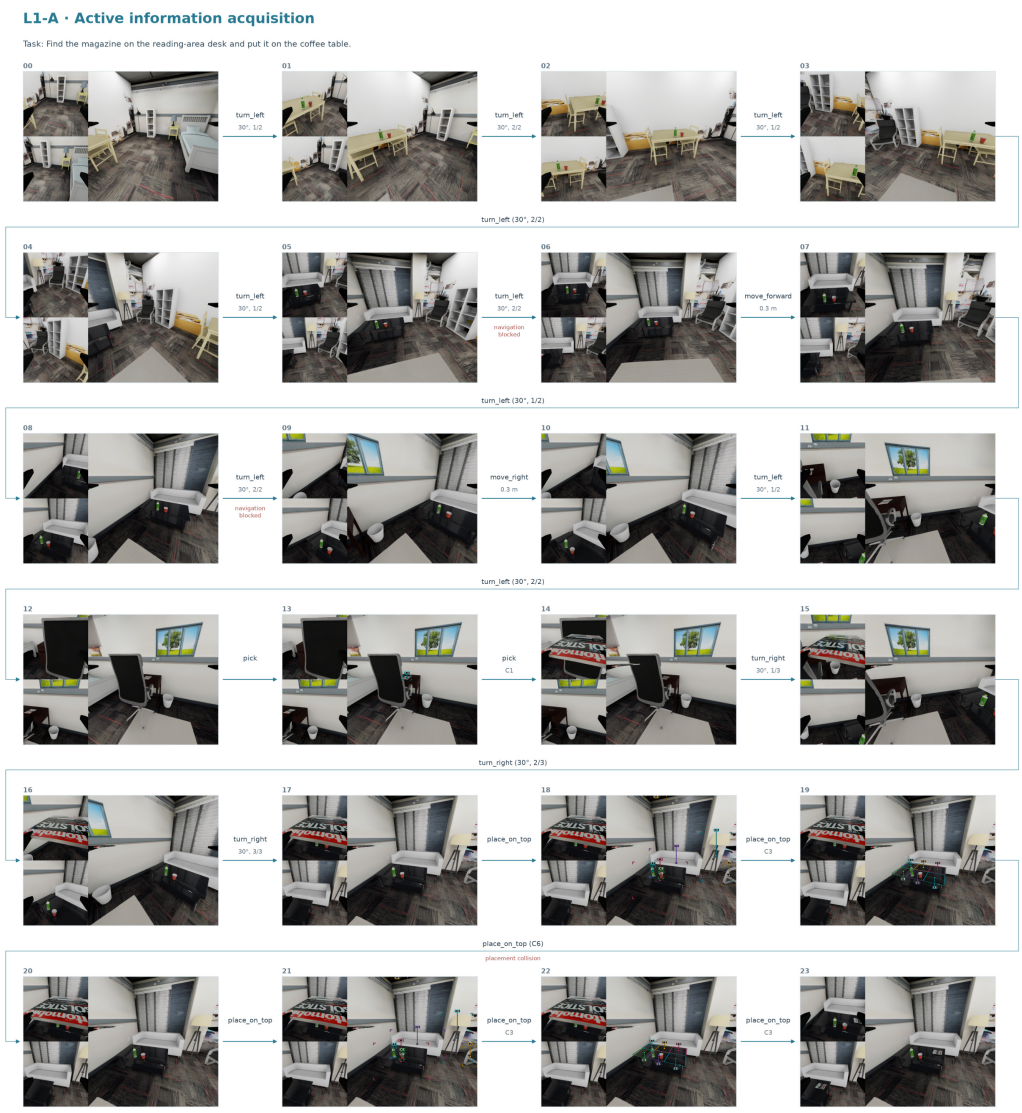

Figure 28: **L1-A: Active Information Acquisition.** The agent changes viewpoint and position to locate the magazine on the reading-area desk, picks it up, and places it on the coffee table. The trajectory also records blocked navigation and a placement collision that must be handled during execution.

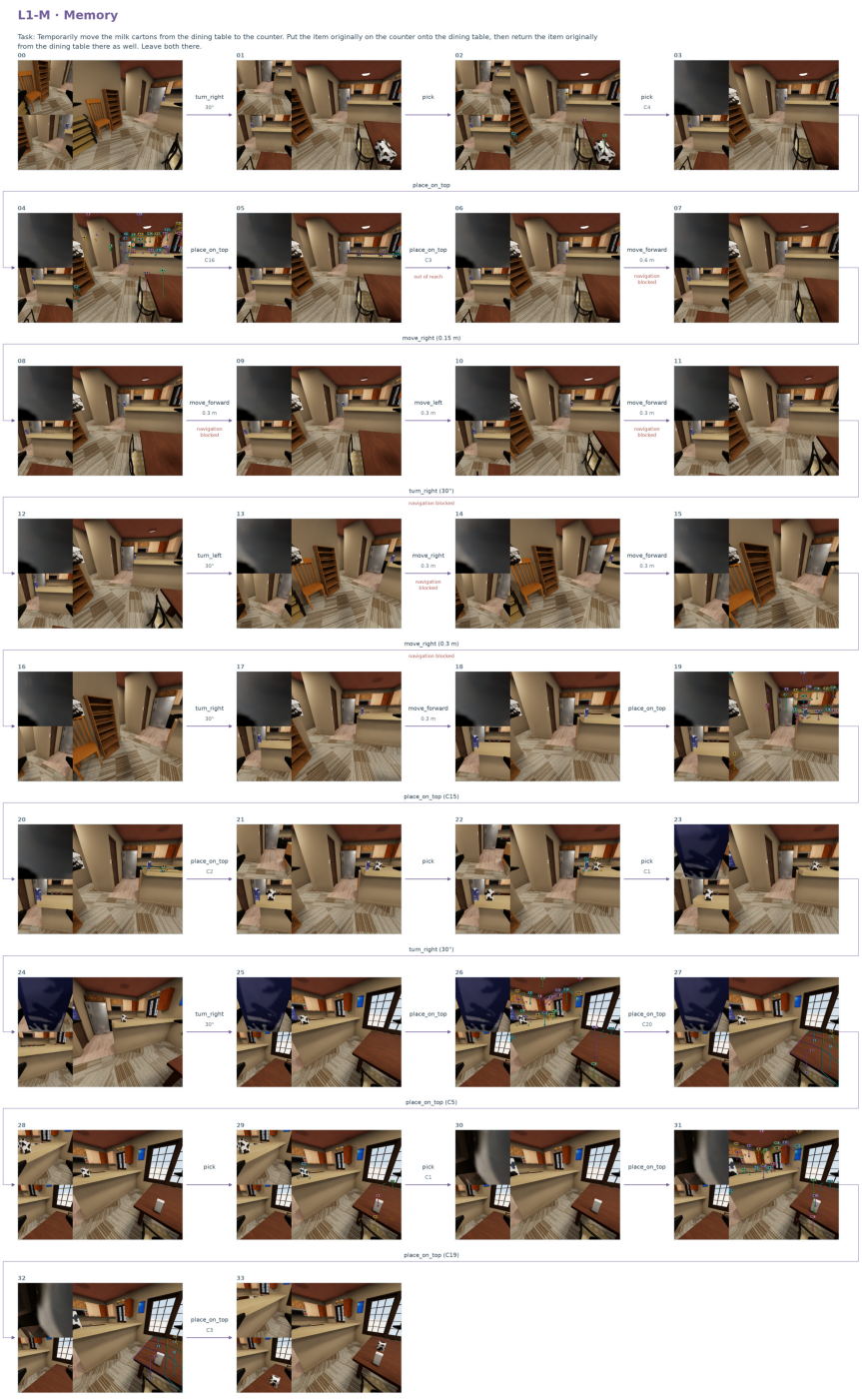

Figure 29: **L1-M: Memory.** The task requires the agent to move the milk cartons, preserve which item originally occupied the counter, and later restore the designated item after the relevant evidence is no longer directly available in the recent interaction context.



Figure 30: **L1-P: Physical Reasoning.** The agent inspects the available supports, picks up the soap dispenser, and selects the support that is physically appropriate for the requested placement.



Figure 31: **L1-V: Verification and Recovery.** After the first closing action, an intervention reopens the microwave door. The agent must inspect the resulting state and close the door again rather than assuming that the first action completed the task.