EMBODIEDTOOL: ENABLING EXTENSIBLE EMBODIED CAPABILITIES WITH TOOLS

Anonymous authors
Paper under double-blind review

ABSTRACT

Compared with end-to-end embodied models that jointly learn perception, reasoning, planning, and control, augmenting foundation models with external tools offers a flexible alternative, allowing specialized capabilities to be independently optimized, reused, and invoked on demand. Recent Code-as-Policy methods have demonstrated the promise of this paradigm, yet progress remains limited by the lack of a unified interface for heterogeneous tools, a comprehensive reusable tool base, and systematic evaluation of models' tool-use capabilities. To bridge these gaps, we present **EmbodiedTool**, which provides a unified agent-facing interface, integrates 112 validated tools spanning perception, cognition, reasoning, and execution, and introduces **EmbodiedToolBench** to evaluate tool-necessity recognition, tool selection, tool execution, and tool-chain composition. Experiments across simulation and real-world platforms show that tool augmentation can substantially improve embodied performance, achieving average gains of 31% on EB-ALFRED and 36% on EB-Navigation, while revealing a clear capability-dependent boundary: gains are substantial for perception and cognition but remain limited for execution-oriented capabilities. Meanwhile, current models still struggle with when, which, and how to use embodied tools, highlighting embodied tool competence as a key challenge for tool-augmented embodied intelligence.

1 INTRODUCTION

Embodied intelligence requires agents to integrate perception, reasoning, planning, and control to interact with physical environments (Zeng et al., 2023; Liang et al., 2025; Salimpour et al., 2025). Most recent advances pursue this goal through end-to-end embodied models, where multiple capabilities are learned within a unified parameterized policy (OpenVLA Team, 2024; Brohan et al., 2024; Black et al., 2024). In comparison, Code-as-Policy (CaP) agents provide a flexible alternative by augmenting foundation models with external embodied tools: specialized capabilities can be independently developed and optimized, reused across models and tasks, and invoked on demand.

Despite this progress, embodied tool use remains fragmented in three key aspects. **(1) Unified interface.** Current CaP systems typically rely on customized interfaces for task- or system-specific tools (Liang et al., 2023; Singh et al., 2023; Huang et al., 2023c; Gonzalez Arenas et al., 2024; Chen et al., 2024; Fu et al., 2026), hindering the integration and reuse of heterogeneous capabilities across models and environments. **(2) Reusable tool base.** Embodied tools remain scattered across individual systems (Liang et al., 2023; Huang et al., 2023c; Chen et al., 2024; Mower et al., 2024; Royce et al., 2024; Guan et al., 2025; Ma et al., 2025; Fu et al., 2026), lacking a comprehensive and reusable collection that covers diverse embodied capabilities. **(3) Systematic evaluation.** While the tool-use capabilities of foundation models have been extensively studied in digital agents (Huang et al., 2023b; Li et al., 2023; Qin et al., 2024; Yao et al., 2023; Schick et al., 2023; Qu et al., 2025), their ability to use tools for embodied interaction remains largely unexamined. In particular, it remains unclear whether current models can reliably determine when tools are needed, select appropriate tools, execute them correctly, and compose multiple tools in physical environments (Fu et al., 2026). Together, these gaps hinder the development and systematic understanding of tool-augmented embodied agents.

To address these gaps, we first establish a unified protocol for embodied tools, explicitly modeling their applicability, multimodal inputs and outputs, physical constraints, execution, and feedback, thereby supporting heterogeneous capabilities across different foundation models. Building on this abstraction,

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.

we develop and validate 112 tools spanning perception, cognition, reasoning, and execution, forming a comprehensive and reusable embodied capability base. We further introduce **EmbodiedToolBench**, the first systematic evaluation of foundation models' embodied tool-use competence, covering tool-need recognition, tool selection, tool execution, and tool-chain composition. Extensive evaluations across eight foundation models, four embodied benchmarks, and real-world robotic tasks demonstrate substantial performance gains from tool augmentation, with average improvements of $+0.31$ on EB-ALFRED and $+0.36$ on EB-Navigation. However, the gains vary across capabilities, dropping to $+0.07$ on the execution-intensive EB-Manipulation. EmbodiedToolBench further reveals that current foundation models still struggle to effectively use embodied tools, particularly in tool recognition, selection, and composition. Overall, our contributions are as follows:

- We build a unified protocol for tool-augmented embodied agents, integrating 112 validated tools spanning perception, cognition, reasoning, and execution. It enables heterogeneous embodied capabilities to be registered, discovered, invoked, and reused across foundation models and tasks.
- We introduce **EmbodiedToolBench**, the first systematic study of foundation models' ability to use tools in embodied interaction. It evaluates embodied tool-use competence across tool-need recognition, tool selection, tool execution, and tool-chain composition, enabling fine-grained diagnosis beyond end-task success.
- We conduct a large-scale empirical study across eight foundation models, four embodied benchmarks, and real-world robotic tasks. Results demonstrate substantial gains from tool augmentation, reaching average improvements of $+0.31$ on EB-ALFRED and $+0.36$ on EB-Navigation, while revealing capability-dependent gains and persistent limitations in models' embodied tool use.

2 PRELIMINARY

2.1 EMBODIED AGENT

We formulate embodied tasks as sequential decision-making problems. Let $o_t \in \mathcal{O}$ be the observation at timestep t, $l \in \mathcal{L}$ the language instruction, and $\tau_t := (o_0, a_0, \ldots, o_t)$ the interaction history, with history space $\mathcal{H}_t := \mathcal{O}^{\leq t} \times \mathcal{A}^{<t}$. A *monolithic* embodied agent defines a single policy:

$$\pi_\theta : \mathcal{H}_t \times \mathcal{L} \longrightarrow \Delta(\mathcal{A}), \qquad a_t \sim \pi_\theta(\cdot \mid \tau_t, l), \tag{1}$$

where θ jointly encodes *all* capabilities at every decision step. Let $\mathcal{C} = \{c_1, \ldots, c_K\}$ be the full capability set required across tasks (e.g. low-level control, spatial reasoning, memory), and $\mathcal{C}(d) \subseteq \mathcal{C}$ the subset demanded by task d. The agent maximizes the expected discounted return via:

$$\mathcal{J}_{\text{mono}}(\theta) := \mathbb{E}_{d\sim p(d)}\left[\mathbb{E}_{\tau\sim p(\tau|\pi_\theta, d)}\left[\sum_{t=0}^{T}\gamma^t r_t\right]\right], \ \gamma \in (0,1], \quad \theta^* = \arg\max_\theta \mathcal{J}_{\text{mono}}(\theta). \tag{2}$$

Because θ must handle all tasks in $\mathcal{D}$, the heterogeneous capabilities $\mathcal{C}$ are *entangled* within a single parameter set, regardless of which subset $\mathcal{C}(d)$ is actually needed per step.

2.2 EMBODIED AGENT WITH TOOLS

To overcome capability entanglement, we externalize each capability as an independently optimized *tool* invoked on demand.

Tool collection. Define the tool collection $\mathcal{Z} := \{z_m(\cdot\,; \phi_m)\}_{m=1}^{M}$, where each tool z_m, parameterized by ϕ_m, realizes a capability subset $\mathcal{C}(z_m) \subseteq \mathcal{C}$. Tools are optimized independently of one another, and their collective coverage $\mathcal{C}_\mathcal{Z} := \bigcup_{m=1}^{M} \mathcal{C}(z_m)$ replaces the entangled encoding inside θ.

Tool-augmented decision process. At each step t, the agent selects a tool $g_t \in \bar{\mathcal{Z}} := \mathcal{Z} \cup \{\perp\}$ (where $\perp$ denotes invoking no tool), queries it with a generated query $q_\theta(\tau_t, l, g_t)$, and conditions its action on the returned observation y_t. Formally, the three-stage transition is:

$$g_t \sim \mu_\theta(\cdot \mid \tau_t, l), \qquad y_t := \mathcal{T}_{g_t}(q_\theta(\tau_t, l, g_t)), \qquad a_t \sim \pi_\theta(\cdot \mid \tau_t, l, y_t). \tag{3}$$

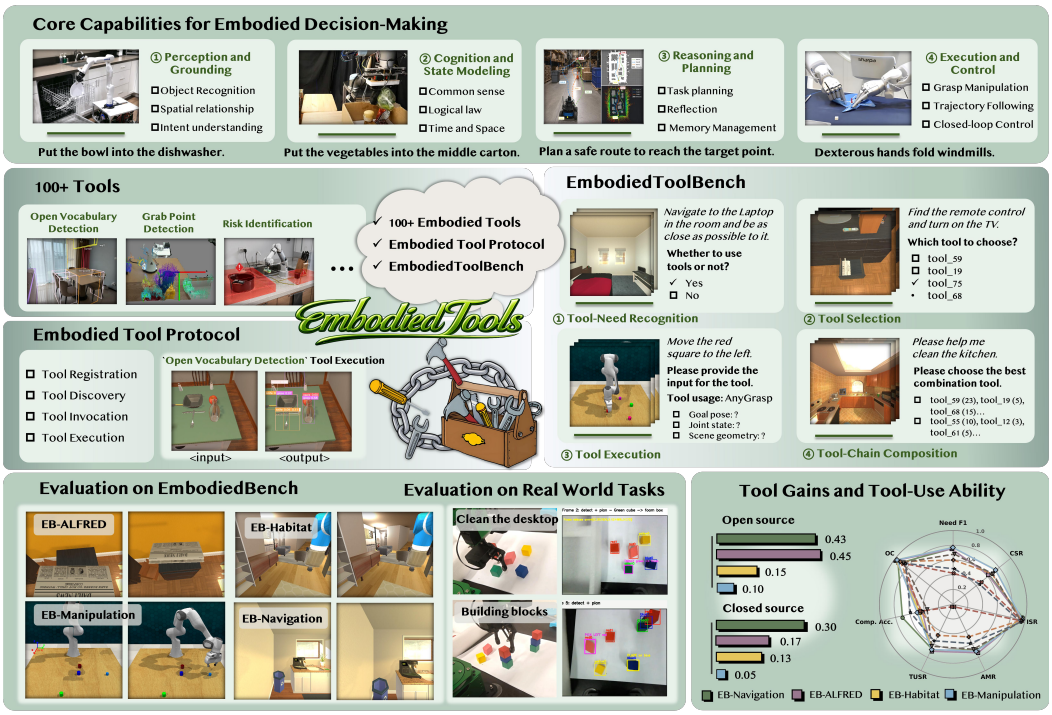

Figure 1: Overview of the **EmbodiedTool**.

Bi-level optimization. Learning decomposes into two decoupled objectives. The *lower-level* objective trains each tool independently on its designated capability dataset $\mathcal{D}_m$:

$$\mathcal{J}_{\text{tool}}(\phi_m) \;:=\; \mathcal{L}_m(\phi_m;\, \mathcal{D}_m), \qquad \phi_m^* \;=\; \arg\min_{\phi_m} \; \mathcal{J}_{\text{tool}}(\phi_m), \quad m \in [M]. \tag{4}$$

The *upper-level* objective trains the orchestration policy θ to maximize expected return over the full task distribution:

$$\mathcal{J}_{\text{orch}}(\theta) \;:=\; \mathbb{E}_{d\sim p(d)}\left[\mathbb{E}_{\tau\sim p_\theta(\tau|d)}\left[\sum_{t=0}^{T}\gamma^t r_t\right]\right], \qquad \theta^* \;=\; \arg\max_{\theta} \; \mathcal{J}_{\text{orch}}(\theta). \tag{5}$$

Each tool ϕ_m^* is optimized independently for its designated capability (Equation (4)), while θ^* learns to orchestrate the tool collection for downstream task completion (Equation (5)). This yields two key advantages: *(i)* capability-specific optimization without cross-task interference, and *(ii)* on-demand invocation that avoids embedding all capabilities into θ at every step.

3 Embodied Tools

3.1 Embodied Tool Protocol and Collection

Embodied Tool Protocol. To provide a common interface for heterogeneous embodied capabilities, we develop the *Embodied Tool Protocol* (ETP) for tool registration, discovery, invocation, and execution. Formally, each tool $z_m \in \mathcal{Z}$ is characterized by its input–output spaces (X_m, Y_m), capability specification ρ_m, interface constraints κ_m, and executable mapping $f_m(\cdot;\phi_m) : X_m \to Y_m$. ETP maintains a registry $\mathcal{R} = \{(z_m, \rho_m, \kappa_m)\}_{m=1}^{M}$, through which an agent identifies applicable tools from ρ_m, constructs valid inputs according to κ_m, and invokes f_m to obtain tool outputs. This protocol allows heterogeneous implementations, including learned models, classical algorithms, and executable programs, to be accessed through a consistent agent-facing interface.

Embodied Tool Collection. We first organize embodied capabilities into four dimensions following prior work (Driess et al., 2023; Ahn et al., 2022): **Perception and Grounding**, **Cognition and State Modeling**, **Reasoning and Planning**, and **Execution and Control**, and further decompose them into

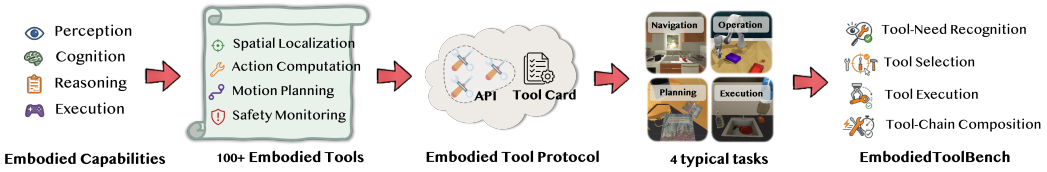

Figure 2: Overview of the EmbodiedToolBench collection process.

fine-grained capability problems. Seven authors survey over 1,000 relevant papers and open-source projects, using established benchmarks to identify strong candidate methods for each capability. For example, for open-vocabulary object detection, we select representative methods (Li et al., 2022; Zhou et al., 2022; Liu et al., 2023b; Cheng et al., 2024) from leading results on COCO (Lin et al., 2014) and LVIS (Gupta et al., 2019), and further validate them across diverse objects, scenes, viewpoints, and backgrounds in embodied environments. Benchmark performance serves only for candidate selection; each candidate undergoes at least five rounds of deployment and validation in simulated or real-world scenarios, and only methods that reliably generalize beyond their original benchmarks are encapsulated as ETP tools. Following this process, we collect more than 112 validated tools spanning the four capability dimensions.

3.2 EMBODIEDTOOLBENCH

Overview. Building on the above tool suite and representative embodied environments for navigation, planning, and manipulation, we introduce **EmbodiedToolBench** to evaluate models' embodied tool-use capability from four complementary perspectives: *tool-need recognition*, *tool selection*, *tool execution*, and *tool-chain composition* (detailed dataset design principles are provided in Appendix E). Each evaluation instance is defined by a decision state $(l, \tau_t, \mathcal{Z}_{\text{cand}})$, where $l \in \mathcal{L}$ is the task instruction, τ_t is the interaction history, and $\mathcal{Z}_{\text{cand}} \subseteq \mathcal{Z}$ is the candidate tool set.

Task 1: Tool-Need Recognition. This task evaluates whether a model can recognize when tool invocation is necessary. Given $(l, \tau_t, \mathcal{Z}_{\text{cand}})$, the model predicts $\hat{u} \in \{0, 1\}$, where $\hat{u} = 1$ indicates that external tools are needed. We determine tool necessity counterfactually: an instance is labeled positive only when no valid tool-free action can achieve the current subgoal given the available observation and interaction history; instances with a valid tool-free solution are labeled negative, and ambiguous cases are excluded. Performance is reported using accuracy and F1 against the ground-truth label $u^\star \in \{0, 1\}$.

Task 2: Tool Selection. This task evaluates whether a model can identify the most appropriate embodied tool from a fixed candidate set. Given $(l, \tau_t, \mathcal{Z}_{\text{cand}})$, the model selects the single most appropriate tool $\hat{z} \in \mathcal{Z}_{\text{cand}}$ from four candidates, which include semantically similar tools of the same category and irrelevant tools as distractors, with exactly one ground-truth answer $z^\star$. Performance is measured by the Correct Selection Rate $\text{CSR} = (1/N) \sum_{i=1}^{N} \mathbf{1}(\hat{z}_i = z_i^\star)$.

Task 3: Tool Execution. This task evaluates whether a model can correctly invoke a tool and act on its output, and is decomposed into two stages. In **Stage 1**, given the current task context (l, τ_t) and the specification of a designated tool $z^\star \in \mathcal{Z}$, the model constructs a query $\hat{x} \in X_{z^\star}$; success requires a well-formed invocation, i.e., $\text{ISR} = \text{Valid}(\hat{x}, z^\star) = 1$. In **Stage 2**, the model receives the task context together with the tool output $y \in Y_{z^\star}$ and must predict the next action $\hat{a} \in \mathcal{A}$; success requires consistency with the reference action $a^\star$, i.e., $\text{AMR} = \text{Match}(\hat{a}, a^\star) = 1$. We report the success rate of each stage independently, as well as the overall Tool Usage Success Rate $\text{TUSR} = (1/N) \sum_{i=1}^{N} \mathbf{1}(\text{Valid}(\hat{x}_i, z_i^\star) = 1 \wedge \text{Match}(\hat{a}_i, a_i^\star) = 1)$.

Task 4: Tool-Chain Composition. This task evaluates whether a model can select the minimal required tools and arrange them in the correct dependency order. Given the task context $(l, \tau_t, \mathcal{Z}_{\text{cand}})$, the model outputs an ordered tool sequence $\hat{P} = (\hat{z}_1, \ldots, \hat{z}_R)$ where each $\hat{z}_r \in \mathcal{Z}_{\text{cand}}$, and $\hat{S}$ denotes the induced tool set. Let $S^\star \subseteq \mathcal{Z}$ denote the ground-truth minimal tool set and $\Omega^\star = \{(z_a, z_b)\}$ the ground-truth ordering constraints, where (z_a, z_b) indicates that z_a must be invoked before z_b. We report tool selection quality via **Accuracy** $\text{ACC} = (1/N) \sum_{i=1}^{N} \mathbf{1}(\hat{S}_i = S_i^\star)$

Table 1: Performance comparison across four embodied benchmarks. **Bold**: best result per column; underlined: worst result; Avg. Gain : mean improvement of "w/ tool" over "w/o tool".

(a) EB-ALFRED and EB-Habitat

Model	EB-ALFRED							EB-Habitat						
	Avg	Base	Common	Complex	Visual	Spatial	Long	Avg	Base	Common	Complex	Visual	Spatial	Long
Closed-Source MLLMs														
GPT-4o	0.48	0.58	0.52	0.52	0.42	0.50	0.36	0.48	0.92	0.32	0.50	0.46	0.30	0.40
w/ tool	0.76	0.90	0.90	0.88	0.76	0.74	0.36	0.64	0.96	0.72	0.60	0.52	0.62	0.40
GPT-5	0.62	0.70	0.60	0.66	0.56	0.68	0.52	0.57	0.84	0.42	0.64	0.66	0.40	**0.46**
w/ tool	0.81	0.90	0.90	0.88	0.78	0.84	0.54	0.75	0.98	0.82	0.72	0.74	**0.76**	**0.46**
Claude-3.7	0.82	0.86	0.82	0.80	0.80	0.76	0.86	0.67	0.98	0.78	0.76	0.68	0.40	0.43
w/ tool	0.91	**0.96**	**0.96**	0.92	**0.90**	0.86	0.86	0.76	**1.00**	0.94	**0.82**	0.70	0.66	0.43
Claude-4.6	0.81	0.86	0.76	0.80	0.78	0.80	0.88	0.67	0.98	0.72	0.70	0.74	0.40	**0.46**
w/ tool	**0.92**	**0.96**	**0.96**	**0.94**	0.86	**0.90**	**0.92**	**0.77**	**1.00**	**0.96**	0.76	**0.76**	0.68	**0.46**
Open-Source MLLMs														
Qwen3-32B	0.37	0.44	0.32	0.44	0.36	0.30	0.38	0.30	0.78	0.22	0.30	0.10	0.36	0.06
w/ tool	0.77	0.92	0.88	0.88	0.78	0.78	0.38	0.44	0.88	0.46	0.46	0.14	0.66	0.06
Qwen3-8B	0.20	0.14	0.18	0.28	0.20	0.14	0.26	0.23	0.44	0.14	0.24	0.24	0.18	0.12
w/ tool	0.72	0.88	0.86	0.88	0.72	0.70	0.26	0.39	0.74	0.38	0.38	0.24	0.46	0.12
Qwen2.5-32B	0.24	0.34	0.24	0.30	0.20	0.24	0.12	0.34	0.68	0.26	0.48	0.26	0.20	0.18
w/ tool	0.66	0.88	0.74	0.88	0.66	0.70	0.12	0.49	0.90	0.46	0.58	0.28	0.56	0.18
Qwen3.5-35B	0.24	0.30	0.28	0.32	0.22	0.22	0.12	0.11	0.30	0.06	0.12	0.06	0.10	0.02
w/ tool	0.70	0.86	0.86	0.88	0.72	0.74	0.12	0.26	0.52	0.22	0.26	0.10	0.46	0.02
Avg. Gain	↑ 0.31	↑ 0.38	↑ 0.42	↑ 0.38	↑ 0.33	↑ 0.33	↑ 0.01	↑ 0.14	↑ 0.13	↑ 0.25	↑ 0.10	↑ 0.03	↑ 0.32	↑ 0.00

(b) EB-Navigation and EB-Manipulation

Model	EB-Navigation						EB-Manipulation					
	Avg	Base	Common	Complex	Visual	Long	Avg	Base	Common	Complex	Visual	Spatial
Closed-Source MLLMs												
GPT-4o	0.46	0.58	0.50	0.47	0.33	0.43	0.31	0.31	0.29	0.27	0.36	0.33
w/ tool	0.83	0.80	0.85	**0.85**	0.80	0.83	0.37	0.44	0.31	0.40	0.36	0.35
GPT-5	0.57	0.62	0.62	0.68	0.50	0.42	0.27	0.21	0.25	0.31	0.22	0.35
w/ tool	**0.85**	**0.85**	**0.87**	**0.85**	0.83	0.85	0.35	0.42	0.27	0.44	0.31	0.31
Claude-3.7	0.55	0.67	0.62	0.63	0.50	0.35	0.38	0.33	0.38	0.46	0.33	**0.38**
w/ tool	0.83	**0.85**	0.82	0.83	**0.85**	0.82	**0.42**	**0.46**	**0.44**	**0.50**	0.36	0.33
Claude-4.6	0.62	0.68	0.63	0.77	0.55	0.48	0.37	0.40	0.35	0.40	0.36	0.37
w/ tool	**0.85**	**0.85**	0.85	**0.85**	**0.85**	0.85	0.38	0.40	0.35	0.42	**0.40**	0.31
Open-Source MLLMs												
Qwen3-32B	0.40	0.52	0.47	0.55	0.35	0.10	0.20	0.21	0.25	0.17	0.19	0.19
w/ tool	0.84	0.83	0.82	**0.85**	0.82	**0.87**	0.25	0.27	0.25	0.31	0.17	0.22
Qwen3-8B	0.42	0.62	0.48	0.45	0.32	0.22	0.19	0.15	0.15	0.25	0.06	0.31
w/ tool	0.80	**0.85**	0.85	0.77	0.75	0.77	0.27	0.29	0.35	0.31	0.19	0.19
Qwen2.5-32B	0.35	0.48	0.42	0.33	0.30	0.23	0.17	0.13	0.13	0.23	0.11	0.25
w/ tool	0.81	0.75	0.83	0.83	0.82	0.83	0.29	0.27	0.35	0.33	0.21	0.28
Qwen3.5-35B	0.39	0.50	0.42	0.48	0.47	0.07	0.15	0.13	0.17	0.15	0.17	0.17
w/ tool	0.81	0.80	0.80	0.80	0.83	0.82	0.27	0.33	0.29	0.27	0.22	0.23
Avg. Gain	↑ 0.36	↑ 0.24	↑ 0.32	↑ 0.28	↑ 0.40	↑ 0.54	↑ 0.07	↑ 0.13	↑ 0.08	↑ 0.09	↑ 0.05	↓ 0.02

and **F1** $\mathrm{F1} = (1/N)\sum_{i=1}^{N} 2|\hat{S}_i \cap S_i^\star|/(|\hat{S}_i| + |S_i^\star|)$, and the **Order Consistency Rate** $\mathrm{OCR} = (1/N)\sum_{i=1}^{N}(1/|\Omega_i^\star|)\sum_{(z_a,z_b)\in\Omega_i^\star} \mathbf{1}(z_a, z_b \in \hat{P}_i \wedge \mathrm{pos}_{\hat{P}_i}(z_a) < \mathrm{pos}_{\hat{P}_i}(z_b))$.

4 Experiments

In this section, we conduct experiments organized around the following research questions, with implementation details provided in Appendix G.1. We evaluate eight representative open- and closed-source models on Embodiedbench (Yang et al., 2025) (EB-ALFRED, EB-Habitat, EB-Navigation, and EB-Manipulation) and real-world robotic platforms. To ensure result reliability, we report multi-run statistical analysis on a subset of evaluations in Appendix G.3.

- **RQ1: Effectiveness of Tool Augmentation.** Do embodied agents equipped with external tools consistently outperform tool-free baselines across both simulation environments and real-world robotic platforms?
- **RQ2: Proficiency of Tool Use Behavior.** What extent do current models demonstrate competence in tool-use awareness, tool selection, tool execution, and multi-tool composition?

- **RQ3: Task-Specific Amenability to Tool Offloading.** Which types of embodied capabilities derive the greatest benefit from being externalized as tools, and which resist such decomposition?

4.1 Does Tool Augmentation Improve Embodied Performance?

The results in Table 1 support the following observations:

Tool augmentation provides substantial overall gains, with clear variation across tasks. The average improvements reach $+0.31$ on EB-ALFRED and $+0.36$ on EB-Navigation, where most models obtain substantial gains over their tool-free counterparts. Improvements are more moderate on EB-Habitat ($+0.14$) and considerably smaller on EB-Manipulation ($+0.07$). These results indicate that tool augmentation can substantially improve embodied task performance, while its effectiveness depends strongly on the task and capability being externalized.

Tool access benefits models across a wide range of base capabilities. For open-source models, tool augmentation can enable performance comparable to or exceeding stronger closed-source models without tools: Qwen3-8B w/ tool (0.72 on EB-ALFRED) outperforms GPT-4o without tools (0.48), while Qwen2.5-32B w/ tool (0.81 on EB-Navigation) surpasses GPT-5 without tools (0.57). Strong closed-source models also benefit from external capabilities: Claude-4.6 w/ tool achieves the highest overall scores on EB-ALFRED (0.92) and EB-Navigation (0.85). These results suggest that tool augmentation is complementary to model capability rather than restricted to weaker models.

The benefits of tool augmentation diminish for execution-intensive and long-horizon tasks. The average gain on EB-Manipulation is only $+0.07$, substantially lower than those on EB-ALFRED ($+0.31$) and EB-Navigation ($+0.36$), with the Spatial subtask showing a slight decline (-0.02). A similar pattern appears in the Long-horizon subtasks of EB-ALFRED ($+0.01$) and EB-Habitat ($+0.00$). These results reveal a capability-dependent boundary of externalization: perception-, cognition-, and navigation-related capabilities can often be effectively delegated to external tools, whereas fine-grained execution and long-horizon coordination remain more difficult to improve through current tool interfaces. Precise physical interaction requires accurate spatial grounding, contact-sensitive control, and robust adaptation to execution feedback, which are not fully captured by existing tool abstractions. We further analyze these limitations in Section 4.5.

Real-world robot experiments. We further evaluate whether access to external embodied tools improves the performance of general-purpose multimodal models on three real-world robotic tasks: desktop cleaning, balance scale manipulation, and building blocks (Figure 3). We compare the same general-purpose models with and without tool access, without introducing task-specific embodied training or platform-specific adaptation. As shown in Table 2, the tool-free agents obtain 0/10 success across all three tasks, whereas enabling external tools substantially improves their success rates. GPT-4o achieves 8/10 on desktop cleaning, Claude-4.6 achieves 9/10 on balance scale manipulation, and Claude-3.7/Claude-4.6 achieve 8/10 on structure copying. These results show that external embodied capabilities can substantially improve the real-world task performance of otherwise unchanged general-purpose multimodal agents.

Table 2: Performance comparison in real-world tasks.

Model	Desktop Cleaning		Balance Scale		Building Blocks	
	w/	w/o	w/	w/o	w/	w/o
GPT-4o	8/10	0/10	6/10	0/10	6/10	0/10
GPT-5	7/10	0/10	8/10	0/10	6/10	0/10
Claude-3.7	6/10	0/10	8/10	0/10	8/10	0/10
Claude-4.6	6/10	0/10	9/10	0/10	8/10	0/10
Qwen3-8B	6/10	0/10	4/10	0/10	5/10	0/10
Qwen3-32B	6/10	0/10	8/10	0/10	5/10	0/10
Qwen2.5-32B	5/10	0/10	6/10	0/10	7/10	0/10
Qwen3.5-35B	4/10	0/10	7/10	0/10	5/10	0/10

4.2 How Well Do Current Models Use Embodied Tools?

We report the evaluation results on EmbodiedToolBench in Table 3, from which we draw the following conclusions. **(1) Tool-Need Recognition.** Recognizing when to invoke tools remains challenging across all models. Open-source models in particular frequently miss necessary tool invocations, with

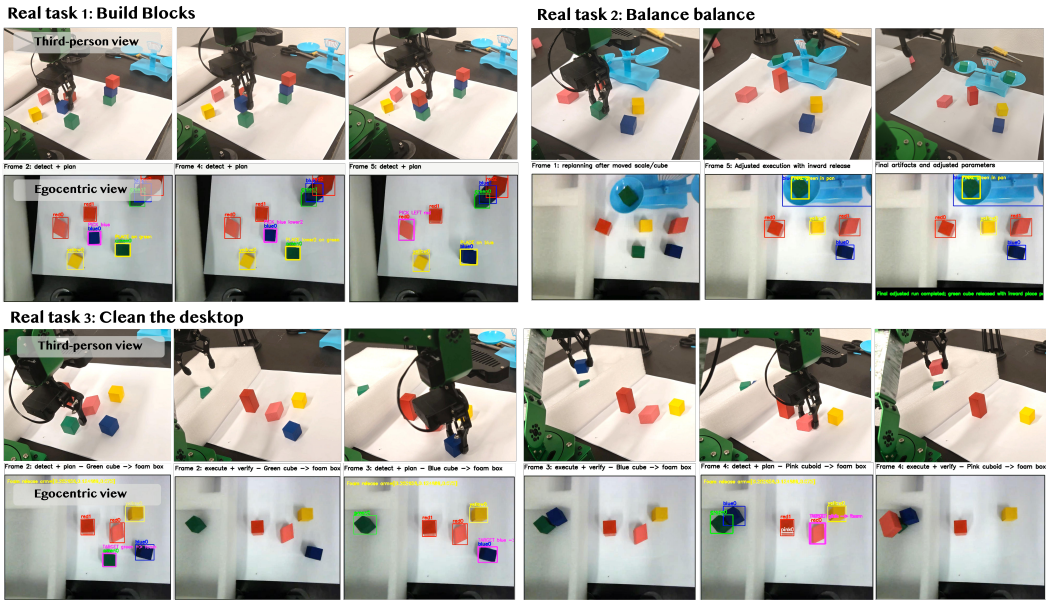

Figure 3: Examples of the real-world robot tasks.

Table 3: Evaluation results on EmbodiedToolBench. Abbreviations are defined in Sec. 3.2.

Model	Tool-Need Recognition				Tool Selection	Tool Execution			Tool-Chain Composition				
	Acc.	Prec.	Rec.	F1	CSR	ISR	AMR	TUSR	Acc.	Prec.	Rec.	F1.	OC
Closed-Source MLLMs													
GPT-4o	0.65	0.67	**0.85**	0.75	0.63	**0.96**	0.70	0.68	**0.71**	**0.93**	0.97	**0.94**	0.93
GPT-5	0.66	0.68	0.84	0.75	0.62	0.94	**0.74**	**0.70**	0.43	0.81	**0.99**	0.88	**0.98**
Claude-3.7	**0.69**	0.72	0.82	**0.77**	**0.74**	0.95	0.71	0.68	0.47	0.83	**0.99**	0.89	**0.98**
Claude-4.6	**0.69**	0.71	0.84	**0.77**	**0.74**	**0.96**	0.71	0.68	0.50	0.83	**0.99**	0.89	0.97
Open-Source MLLMs													
Qwen3-32B	0.63	**0.90**	0.45	0.60	0.67	0.95	0.49	0.47	0.44	0.83	0.96	0.87	0.92
Qwen3-8B	0.48	0.68	0.31	0.42	0.49	**0.96**	0.04	0.04	0.46	0.84	0.92	0.86	0.87
Qwen2.5-32B	0.63	0.74	0.63	0.68	0.68	0.95	0.67	0.64	0.36	0.83	0.94	0.87	0.88
Qwen3.5-35B	0.55	0.84	0.34	0.48	0.45	0.95	0.58	0.56	0.59	0.91	0.91	0.90	0.85

recall dropping as low as 0.31, indicating that current models have not yet developed reliable judgment on tool necessity in embodied settings. **(2) Tool Selection.** Tool selection remains a non-trivial challenge across all models. The best results are achieved by Claude-3.7 and Claude-4.6 (0.74), while weaker open-source models drop to as low as 0.45, reflecting difficulty in grounding the correct tool among semantically similar candidates. **(3) Tool Execution.** Most models achieve consistently high input construction scores (0.94–0.96), yet result utilization emerges as the primary bottleneck: GPT-5 achieves the highest tool-use success rate (0.70), while several open-source models struggle to interpret and act upon returned tool outputs effectively. **(4) Tool-Chain Composition.** Composition accuracy correlates with general model capability, ranging from 0.36 to 0.71. A dominant failure pattern is over-composition: stronger models achieve high recall (0.97–0.99) but precision varies widely (0.81–0.93), indicating that models cover required tools but frequently include superfluous ones, highlighting the difficulty of precise multi-tool coordination.

4.3 Which Capabilities Benefit Most from Externalization?

As shown in Figure 4, tool augmentation improves performance across all four capability dimensions for both closed-source and open-source models. The largest gains occur in Perception and Cognition, reaching +0.26/+0.28 for closed-source models and +0.26/+0.34 for open-source models, respectively, suggesting that external tools are especially effective for visual parsing and knowledge-driven tasks.

Reasoning also improves consistently, e.g., +0.17 for GPT-4o and +0.29 for Qwen2.5-VL-32B, indicating that tools can complement internal inference. In contrast, Execution shows smaller and more variable gains, ranging from +0.03 to +0.10 for closed-source models and +0.14 to +0.19 for open-source models, reflecting its reliance on precise input construction and tool generalization.

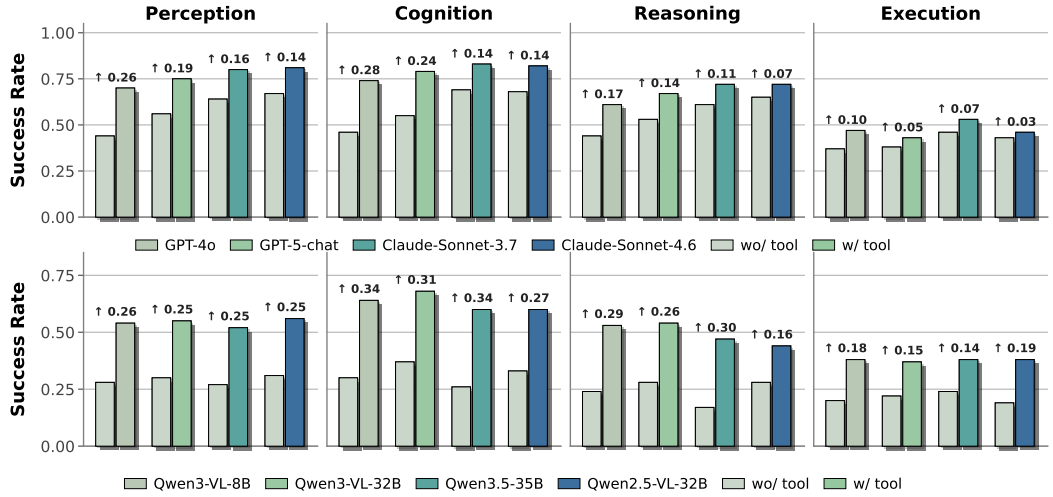

Figure 4: Distribution of EmbodiedToolBench across capability dimensions.

4.4 Inference Time Overhead Analysis

Table 4 reports the average per-task inference time across four embodied benchmarks, comparing settings with and without tool augmentation. Overall, the latency overhead introduced by tool usage remains acceptable in most cases. On EB-ALFRED and EB-Habitat, the additional cost is generally modest, typically within 10 to 30 seconds (e.g., Claude-3.7 on EB-ALFRED: 38.88s $\rightarrow$ 63.36s; Claude-4.6 on EB-Habitat: 39.60s $\rightarrow$ 49.68s), suggesting that tool augmentation can be integrated without substantially disrupting inference efficiency in these settings. By contrast, EB-Manipulation presents a more challenging profile, where several models incur latency increases exceeding 30 seconds. This is mainly because manipulation tasks invoke tools with higher inference costs and are more affected by network latency during execution. These results point to an important direction for future work: given the stringent real-time requirements of embodied tasks, agents must not only select tools accurately but also reason about their temporal cost.

Table 4: Task-level inference time on EmbodiedToolBench with and without tool augmentation.

Task	Setting	Closed-Source MLLMs				Open-Source MLLMs			
		GPT-4o	GPT-5	Claude-3.7	Claude-4.6	Qwen3-8B	Qwen3-32B	Qwen3.5-35B	Qwen2.5-32B
EB-ALFRED	w/o tool	50.40s	67.68s	38.88s	38.88s	51.12s	63.36s	59.76s	56.16s
	w/ tool	69.12s	106.56s	63.36s	54.00s	76.32s	87.84s	87.84s	72.00s
EB-Habitat	w/o tool	46.80s	67.68s	38.88s	39.60s	36.72s	61.92s	46.80s	56.16s
	w/ tool	57.60s	75.60s	44.64s	49.68s	53.28s	74.16s	72.72s	87.12s
EB-Navigation	w/o tool	64.08s	74.16s	53.28s	55.44s	60.48s	59.76s	66.24s	64.80s
	w/ tool	61.20s	66.24s	74.16s	76.32s	57.60s	63.36s	59.04s	54.00s
EB-Manipulation	w/o tool	46.51s	88.85s	98.14s	102.17s	39.89s	50.40s	50.83s	79.63s
	w/ tool	56.95s	55.58s	91.66s	92.52s	72.29s	99.65s	86.47s	87.19s

4.5 Error Analysis

We conduct a fine-grained error analysis across four benchmarks, categorizing failures into five types: Missed Tool Invocation, Invalid Tool Call, Wrong Selection, Ignored Output, and Tool-induced Bias (Figure 5). Missed Tool Invocation is most severe in action-intensive tasks (36% and 45% of error cases in EB-Navigation and EB-Manipulation), while Ignored Output dominates in perception-heavy

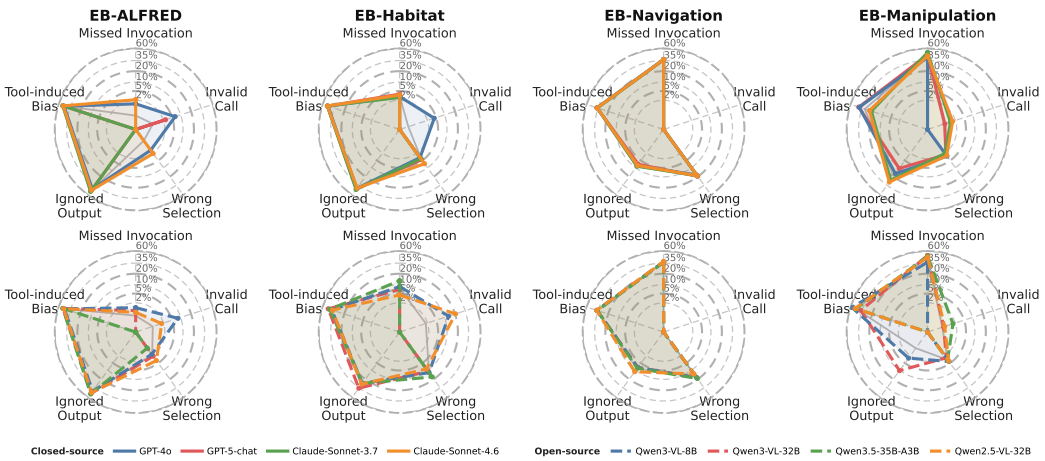

Figure 5: Error analysis on EmbodiedBench.

settings (47% and 43% in EB-ALFRED and EB-Habitat). Tool-induced Bias remains consistently non-trivial, peaking at 56.5% for Qwen3-VL-8B in EB-Manipulation. Wrong Selection and Invalid Tool Call are relatively minor overall, though open-source models show notably higher rates in EB-Habitat (e.g., Qwen3.5-35B-A3B: 17.0%; Qwen2.5-VL-32B: 20.6%), reflecting weaker tool-interface understanding than closed-source counterparts.

5 Related Work

Embodied agents require heterogeneous capabilities, including perception, reasoning, planning, control, memory, and adaptation, to operate in open-world environments (Zeng et al., 2023; Wang et al., 2024a; Liang et al., 2025; Salimpour et al., 2025). Existing approaches develop these capabilities through hierarchical decision-making (Ahn et al., 2022; Singh et al., 2023; Huang et al., 2022), scene and spatial representations (Chen et al., 2023; Huang et al., 2023a; Rana et al., 2023; Werby et al., 2024; Booker et al., 2024), and integrated policy, planning, and control frameworks (Ma et al., 2024; Mon-Williams et al., 2025; Liu et al., 2023a; Chen et al., 2024; Ao et al., 2024). In parallel, recent Code-as-Policy and tool-augmented embodied systems enable foundation models to access external capabilities through callable functions, robot APIs, or protocol-based services (Mower et al., 2024; Royce et al., 2024; Guan et al., 2025; Ma et al., 2025). These systems demonstrate the feasibility of augmenting embodied agents with external capabilities, but typically integrate a limited set of task- or system-specific tools through customized interfaces. Consequently, embodied tools remain fragmented across individual systems, and the ability of foundation models to recognize, select, execute, and compose such tools has received limited systematic evaluation. Our work builds on this direction by providing a common interface for heterogeneous embodied tools, constructing and validating a reusable suite of 100+ tools, and systematically evaluating embodied tool-use competence with *EmbodiedToolBench* (see Appendix C for an extended discussion).

6 Conclusion

In this paper, we develop a unified protocol and curate 100+ validated embodied tools spanning perception, cognition, reasoning, and execution, providing a reusable capability base for tool-augmented embodied agents. We further introduce **EmbodiedToolBench** to systematically evaluate foundation models across tool-need recognition, tool selection, tool execution, and tool-chain composition. Extensive evaluations across simulation and real-world robotic tasks demonstrate substantial gains from tool augmentation, while revealing that these gains vary across embodied capabilities and that current foundation models still face significant challenges in effectively using embodied tools. We hope this work provides a foundation for developing more capable and extensible tool-augmented embodied agents.

Limitation. Despite consistent gains, embodied tool use remains challenging due to failures in tool selection, invocation timing, and multi-tool coordination in dynamic environments. Future work should focus on adaptive strategies that balance task success, invocation cost, and error propagation.

AI USE STATEMENT

We used generative AI tools to assist with understanding background knowledge and concepts related to the research area, running and debugging research code, and improving the clarity and readability of the manuscript. The authors independently verified all technical content, experimental outputs, and interpretations. All AI-assisted text was reviewed and revised by the authors, who take full responsibility for the final content, experimental results, and conclusions of this work.

ETHICS STATEMENT

This work focuses on general embodied AI methods and does not involve security-related research. The experiments are conducted for research and evaluation purposes in standard simulated and robotic settings. We do not identify any specific ethical risks beyond those generally associated with embodied AI research.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the proposed method, implementation, experimental settings, and evaluation protocols in the main paper and appendix. Key hyperparameters, task configurations, and experimental procedures are documented to facilitate reproduction of the reported results. All experiments follow consistent evaluation settings, and the reported results are obtained using the configurations described in the paper.

REFERENCES

Ahn et al. Do As I Can, Not As I Say: Grounding language in robotic affordances. In *Conference on Robot Learning (CoRL)*, pp. 287–318, 2022.

Ao et al. LLM-as-BT-Planner: Leveraging llms for behavior tree generation in robot task planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1233–1239, 2024.

Bhat et al. ZoeDepth: Zero-shot transfer by combining relative and metric depth. *arXiv*, 2023. doi: 10.48550/arXiv.2302.12288.

Black et al. π_0: A vision-language-action flow model for general robot control. *arXiv*, 2024.

Booker et al. EmbodiedRAG: Dynamic 3d scene graph retrieval for efficient and scalable robot task planning. *arXiv*, 2024.

Brohan et al. Octo: An open-source generalist robot policy. *arXiv*, 2024.

Junting Chen, Yao Mu, Qiaojun Yu, Tianming Wei, Silang Wu, Zhecheng Yuan, Zhixuan Liang, Chao Yang, Kaipeng Zhang, Wenqi Shao, Yu Qiao, Huazhe Xu, Mingyu Ding, and Ping Luo. RoboScript: Code generation for free-form manipulation tasks across real and simulation. *arXiv*, 2024.

Chen et al. Open-vocabulary queryable scene representations for real world planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11509–11522, 2023.

Chen et al. AutoTAMP: Autoregressive task and motion planning with llms as translators and checkers. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6695–6702, 2024.

Cheng et al. Putting the object back into video object segmentation. *arXiv*, 2023. doi: 10.48550/arXiv.2310.12982.

Cheng et al. YOLO-World: Real-time open-vocabulary object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16901–16911, 2024.

Doersch et al. TAPIR: Tracking any point with per-frame initialization and temporal refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10061–10072, 2023.

Driess et al. PaLM-E: An embodied multimodal language model. *arXiv*, 2023.

Fang et al. AnyGrasp: Robust and efficient grasp perception in spatial and temporal domains. *IEEE Transactions on Robotics*, 2023. doi: 10.48550/arXiv.2212.08333.

Max Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Fei-Fei Li, Guanya Shi, Jiajun Wu, Shankar Sastry, Yuke Zhu, Ken Goldberg, and Linxi Fan. CaP-X: A framework for benchmarking and improving coding agents for robot manipulation. In *International Conference on Machine Learning (ICML)*, 2026.

Montserrat Gonzalez Arenas, Ted Xiao, Sumeet Singh, Vidhi Jain, Allen Z. Ren, Quan Vuong, Jake Varley, Alexander Herzog, Isabel Leal, Sean Kirmani, Mario Prats, Dorsa Sadigh, Vikas Sindhwani, Kanishka Rao, Jacky Liang, and Andy Zeng. How to prompt your robot: A promptbook for manipulation skills with code as policies. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4340–4348, 2024. doi: 10.1109/ICRA57147.2024.10610784.

Guan et al. RoboNeuron: A modular framework linking foundation models and ros for embodied ai. *arXiv*, 2025.

Agrim Gupta, Piotr Dollar, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5356–5364, 2019. doi: 10.1109/CVPR.2019.00550.

Hornung et al. OctoMap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34:189–206, 2013. doi: 10.1007/s10514-012-9321-0.

Liyu Hou, Linyuan Gao, Yuan Wu, and Yi Chang. A survey on evaluation of embodied ai. *TechRxiv Preprint*, 2026. doi: 10.22541/au.177023340.02874343/v1.

Huang et al. Inner Monologue: Embodied reasoning through planning with language models. *arXiv*, 2022.

Huang et al. Visual language maps for robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10608–10615, 2023a.

Huang et al. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *arXiv*, 2023b.

Huang et al. VoxPoser: Composable 3d value maps for robotic manipulation with language models. *arXiv*, 2023c.

Hughes et al. Hydra: A real-time spatial perception system for 3d scene graph construction and optimization. *arXiv*, 2022. doi: 10.48550/arXiv.2201.13360.

Ji et al. Action Genome: Actions as compositions of spatio-temporal scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10236–10247, 2020.

Kim et al. LINGO-Space: Language-conditioned incremental grounding for space. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10314–10322, 2024. doi: 10.1609/aaai.v38i9.28898.

Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, Kai-Wei Chang, and Jianfeng Gao. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10965–10975, 2022. doi: 10.1109/CVPR52688.2022.01069.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3102–3116, 2023. doi: 10.18653/v1/2023.emnlp-main.187.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as Policies: Language model programs for embodied control. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500, 2023. doi: 10.1109/ICRA48891.2023.10160591.

Liang et al. Large Model Empowered Embodied AI: A survey on decision-making and embodied learning. *arXiv*, 2025.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pp. 740–755, 2014. doi: 10.1007/978-3-319-10602-1_48.

Liu et al. LLM+P: Empowering large language models with optimal planning proficiency. *arXiv*, 2023a.

Liu et al. Grounding DINO: Marrying dino with grounded pre-training for open-set object detection. *arXiv*, 2023b.

Liu et al. Lang2LTL: Translating natural language commands to temporal specification with large language models. *arXiv*, 2023c.

Ma et al. A survey on vision-language-action models for embodied ai. *arXiv*, 2024.

Ma et al. Orchestrating embodied systems through the embodied context protocol: Motivation, progress, and directions. *Research*, 2025.

Macenski et al. The Marathon 2: A navigation system. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. URL `https://github.com/ros-planning/navigation2`.

Mon-Williams et al. Embodied large language models enable robots to complete complex tasks in unpredictable environments. *Nature Machine Intelligence*, 7:592–601, 2025.

MoveIt Contributors. Moveit motion planning framework. `https://moveit.ai/`, 2024. ROS-based motion planning framework for robotic manipulation.

Mower et al. ROS-LLM: A ros framework for embodied ai with task feedback and structured reasoning. *arXiv*, 2024.

Nair et al. R3M: A universal visual representation for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2022. doi: 10.48550/arXiv.2203.12601.

Nguyen et al. GigaPose: Fast and robust novel object pose estimation via one correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9903–9913, 2024.

Oh et al. Video object segmentation using space-time memory networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9226–9235, 2019.

OpenVLA Team. OpenVLA: An open vision-language-action model. *arXiv*, 2024.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations (ICLR)*, 2024.

Qu et al. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 2025.

Rana et al. SayPlan: Grounding large language models using 3d scene graphs for scalable task planning. In *Conference on Robot Learning (CoRL)*, pp. 23–72, 2023.

Royce et al. Enabling Novel Mission Operations and Interactions with ROSA: The robot operating system agent. In *IEEE Aerospace Conference*, pp. 1–16, 2024.

Salimpour et al. Towards embodied agentic ai: Review and classification of llm- and vlm-driven robot autonomy and interaction. *arXiv*, 2025.

Schick et al. Toolformer: Language models can teach themselves to use tools. *arXiv*, 2023.

Singh et al. ProgPrompt: Generating situated robot task plans using large language models. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11523–11530, 2023.

Sundermeyer et al. Contact-GraspNet: Efficient 6-dof grasp generation in cluttered scenes. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13438–13444, 2021. doi: 10.1109/ICRA48506.2021.9561877.

Wang et al. Large Language Models for Robotics: Opportunities, challenges, and perspectives. *arXiv*, 2024a.

Wang et al. DUSt3R: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20697–20709, 2024b.

Werby et al. Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation. *arXiv*, 2024.

Wu et al. SceneGraphFusion: Incremental 3d scene graph prediction from rgb-d sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7515–7525, 2021.

Wu et al. Grounding Generative Planners in Verifiable Logic: A hybrid architecture for trustworthy embodied ai. *arXiv*, 2026.

Yamazaki et al. Open-Fusion: Real-time open-vocabulary 3d mapping and queryable scene representation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2024. doi: 10.1109/ICRA57147.2024.10610193.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. EmbodiedBench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 70576–70631. PMLR, 2025. URL `https://proceedings.mlr.press/v267/yang25f.html`.

Yao et al. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Yin et al. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11784–11793, 2021.

Zeng et al. Large Language Models for Robotics: A survey. *arXiv*, 2023.

Zhao et al. Fast segment anything. *arXiv*, 2023. doi: 10.48550/arXiv.2306.12156.

Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. In *European Conference on Computer Vision (ECCV)*, 2022.

A LIMITATIONS

Embodied tool use remains challenging for current models. Models may still fail to decide when tools are needed, select appropriate tools, construct valid inputs, or coordinate multiple tools in long-horizon tasks. These failures are more critical in embodied settings, where observations are dynamic, actions affect the physical world, and tool outputs must be grounded in changing environments. Future work should study how embodied tools can be independently optimized, verified, and adapted across diverse tasks and environments. Tool invocation also introduces extra execution overhead. Developing adaptive tool-use strategies that balance task success, invocation cost, and error propagation is an important direction for future research.

B BROADER IMPACTS

Tool-augmented embodied agents may improve the modularity, reproducibility, and reliability of embodied AI systems by allowing agents to reuse specialized external capabilities rather than relying solely on a monolithic policy. This can help diagnose model failures, support more transparent evaluation, and facilitate the development of safer embodied systems. Our released assets are intended for research and evaluation in controlled settings, and all tools used in our experiments are validated before evaluation.

C RELATED WORK SUPPLEMENT

Embodied agents. Embodied agents must coordinate a broad spectrum of capabilities to operate in open-world environments, including perception and scene understanding, spatial reasoning, task planning, action sequencing, motion control, memory, and feedback-driven adaptation (Zeng et al., 2023; Wang et al., 2024a; Liang et al., 2025; Salimpour et al., 2025). Recent embodied-agent research has therefore moved beyond isolated perception or control modules toward hierarchical decision-making systems that couple high-level cognition with low-level execution (Zeng et al., 2023; Liang et al., 2025). Representative works such as SayCan, ProgPrompt, and Inner Monologue illustrate this trend by combining language understanding with affordance grounding, task decomposition, and interactive replanning for long-horizon embodied tasks (Ahn et al., 2022; Singh et al., 2023; Huang et al., 2022). These studies collectively suggest that embodied decision-making is fundamentally a heterogeneous capability integration problem rather than a single-task prediction problem.

Expanding embodied capabilities within the model. A dominant line of work seeks to expand embodied capabilities by improving the model itself through pretraining, post-training, architectural specialization, or tighter perception-action integration. For spatial understanding, prior work has developed open-vocabulary scene representations, visual-language maps, 3D scene graphs, and retrieval-based world grounding to improve object semantics, geometric reasoning, and environment querying (Chen et al., 2023; Huang et al., 2023a; Rana et al., 2023; Werby et al., 2024; Booker et al., 2024). For action and control, another major direction is to train more capable vision-language-action models and embodied policies that more tightly couple perception, reasoning, and action generation (Ma et al., 2024; Mon-Williams et al., 2025). Planning-oriented approaches further enhance the policy core by integrating language models with symbolic planners, task-and-motion planning, and behavior-tree generation (Liu et al., 2023a; Chen et al., 2024; Ao et al., 2024). While these efforts substantially improve embodied competence, they still face persistent limitations in coverage, modular reuse, robustness, and the flexible composition of heterogeneous capabilities, especially in long-horizon, compositional, and safety-critical settings (Wang et al., 2024a; Liang et al., 2025; Salimpour et al., 2025).

Tool usage. In parallel, the LLM agent literature has shown that outsourcing capabilities to external tools can significantly improve scalability, flexibility, and task performance, leading to a mature paradigm of tool selection, tool calling, and multi-step orchestration (Qu et al., 2025). Embodied AI has already exhibited early signs of this direction: ROS-LLM exposes robot actions and services to language-model-based reasoning, ROSA wraps ROS functionality into callable tools, RoboNeuron uses MCP to expose robot capabilities as agent-callable tools, and ECP-style work argues for protocol-level coordination across embodied subsystems (Mower et al., 2024; Royce et al., 2024; Guan et

al., 2025; Ma et al., 2025). Recent surveys also recognize modular tool use and protocol-oriented integration as emerging themes in embodied AI (Liang et al., 2025; Salimpour et al., 2025). However, despite these promising early efforts, there is still no standardized embodied tool protocol for unified capability access, no systematically collected and validated embodied tool suite spanning the full embodied decision stack, and no clear understanding of how effectively current embodied models can select, coordinate, and benefit from outsourced tools in embodied settings. These gaps motivate our *EmbodiedTool* framework.

D EMBODIEDTOOL

D.1 TAXONOMY OF EMBODIED CAPABILITIES

To justify the capability decomposition adopted in this work, we organize embodied intelligence around four core capability groups: **perception and grounding**, **cognition and state modeling**, **reasoning and planning**, and **execution and control**. This taxonomy is consistent with recent surveys that analyze embodied systems through the full perception-cognition-planning-action loop, arguing that embodied intelligence should be examined as an integrated process from environmental sensing to physical execution (Hou et al., 2026).

Our decomposition follows the same high-level logic while adapting it to the goal of tool construction. Specifically, we focus on capability groups that can be externalized as independently parameterized embodied tools and invoked on demand during decision-making. The resulting taxonomy provides a principled basis for deciding what kinds of tools should be built, how they should be organized, and what role they play in embodied agents.

Perception and grounding. This capability group concerns how an agent acquires environmental information and grounds task instructions in multimodal observations. It includes visual perception, object recognition, open-vocabulary detection, spatial localization, scene understanding, and language-vision grounding. The goal is not merely to sense raw inputs, but to convert observations and instructions into structured, task-relevant representations that can support subsequent reasoning, planning, and control.

Cognition and state modeling. This capability group concerns how an agent interprets perceived information and maintains an internal representation of the current situation. It includes state estimation, relational modeling, affordance understanding, task-constraint interpretation, risk identification, and commonsense-based situation assessment. In embodied settings, this capability enables the agent to determine what the current environment means, which entities and relations are relevant, and what latent state or dependency should be modeled for downstream decision-making.

Reasoning and planning. This capability group concerns how an agent derives decisions and organizes future behavior based on the modeled state and task goal. It includes goal decomposition, subgoal generation, causal reasoning, long-horizon planning, action sequencing, contingency handling, and replanning under environmental changes. While cognition and state modeling focus on understanding the current situation, reasoning and planning focus on deciding what should be done next and how individual decisions should be composed into a coherent executable plan.

Execution and control. This capability group concerns how an agent realizes planned decisions through physically grounded actions. It includes trajectory generation, grasp planning, manipulation control, navigation control, low-level actuation, and execution monitoring. This capability ensures that abstract decisions and planned action sequences can be translated into stable, feasible, and effective behavior in the physical world.

D.2 EMBODIED TOOL COLLECTION

Collection objective. Our embodied tool collection is designed to externalize the major capabilities required by embodied decision-making into callable and independently deployable tools. Starting from the capability taxonomy in Section D.1, we organize the collection around four major dimensions: **perception and grounding**, **cognition and state modeling**, **reasoning and planning**, and **execution**

and control. For tool construction, each capability dimension is further decomposed into finer-grained embodied problems, such as object detection, segmentation, spatial grounding, memory retrieval, affordance reasoning, task decomposition, action grounding, and closed-loop execution monitoring. This decomposition allows us to move from abstract capabilities to concrete toolable functions.

Problem decomposition and source collection. For each fine-grained problem, we use AI-assisted literature exploration tools, including deep researcher, to collect representative state-of-the-art methods from the corresponding domains, such as computer vision, robot learning, multimodal reasoning, and motion planning. The collected candidates include both widely adopted research models and practically useful system modules. At this stage, our goal is not simply to maximize the number of candidate methods, but to identify methods whose input-output behavior, execution assumptions, and deployment cost are compatible with embodied use.

Human review and tool selection. All collected candidates are then manually reviewed before being admitted into the tool suite. In particular, we prioritize methods that are 1) *lightweight*, so that they are practical for embodied use, 2) *easy to deploy*, so that they can be maintained within a unified serving framework, and 3) *environmentally robust*, so that they can adapt across different environments and operating conditions. We also examine whether a method matches the target embodied subproblem, whether its interface can be exposed as a reusable tool, and whether it provides sufficiently stable outputs for downstream decision-making. This review step is important because not every strong method on its original benchmark is suitable as an embodied tool: some methods are too heavy for efficient deployment, some are difficult to integrate into real systems, and some do not generalize reliably beyond a narrow evaluation setting.

Deployment, validation, and API wrapping. For each selected method, we first implement or adapt a deployable version and validate it in our embodied setting. Importantly, we do not rely only on the method's original benchmark. Instead, beyond its native evaluation setup, we additionally verify each tool on two to three extra benchmarks or testing scenarios whenever possible, so as to ensure that the tool remains effective across different environments, task distributions, and input conditions. After validation, each tool is deployed on our servers and encapsulated as an accessible API. We further attach a structured tool card to each deployed tool, specifying its functionality, input-output schema, applicability conditions, and usage constraints, so that the tool can be registered under ETP and invoked consistently by embodied agents.

Representative tools. Figure 6 visualizes the word embeddings of over 100 tools, showing how they span diverse embodied capabilities in the tool space. Table 5 lists a representative subset of tools from the broader collection. We select examples that are already deployed or operationally validated, and that together cover memory retrieval, intent grounding, closed-loop monitoring, grasp generation, motion planning, inverse kinematics, and contact-aware control.

Table 5: Representative tool catalogue of the embodied benchmark suite. Tools are grouped by capability unit. **Mode**: *On-demand* = invoked once per query; *Continuous* = runs in background loop; *Event* = triggered by runtime condition. **Source** lists the originating paper or framework.

ID	Tool Name	Capability Unit	Description	Input	Output	Trigger Condition	Mode	Source
Queryable Memory								
T01	`query_3d_scene_graph`	Queryable Memory	Queries historical position and spatial relations of objects observed in a 3-D scene graph.	Scene graph; object/relation query	Object location; neighbor relations; confidence	Target lost from current view	On-demand	(Wu et al., 2021)
T02	`STM`	Queryable Memory	Maintains a fixed-size space-time memory bank; reads and writes target features frame by frame to combat long-term drift.	Current frame features; memory bank query	Memory readout; attention map; confidence score	Target blurred; VLM needs early-frame feature anchor	Continuous	(Oh et al., 2019)

Continued on next page

Table 5 – continued from previous page

ID	Tool Name	Capability Unit	Description	Input	Output	Trigger Condition	Mode	Source
T03	ActionGenome	Queryable Memory	Infers the current physical state of objects from their motion trajectory via a spatio-temporal scene graph network.	Short video clip; target object list	State graph; temporal relations	VLM needs exact object state for next-step decision	On-demand	(Ji et al., 2020)
Intent Grounding								
T04	Language2LTL	Intent Grounding	Translates natural-language SOP into a formal LTL formula; rejects step-skipping or unsafe hallucinated plans before execution.	VLM action plan; safety LTL formulas	Validation result; violation feedback	VLM completes abstract planning; output about to be queued	On-demand	(Liu et al., 2023c)
T05	LINGO-Space	Intent Grounding	Incrementally estimates a probability distribution over goal positions given composite spatial-referring instructions and scene geometry.	Composite instruction; scene objects	Target location distribution; confidence	Instruction contains spatial referring expressions	On-demand	(Kim et al., 2024)
Detection & Segmentation								
T06	YOLO-World	Det. & Seg.	Real-time open-vocabulary object detector; finds arbitrary text-described objects in RGB frames.	RGB image; text query	Bounding boxes; object categories	Open-vocabulary detection required	On-demand	(Cheng et al., 2024)
T07	FastSAM	Det. & Seg.	CNN-based real-time solution for the Segment Anything task; supports point, box, and text prompts.	RGB image; point / box / text prompts	Segmentation masks	Fast instance segmentation required	On-demand	(Zhao et al., 2023)
T08	GigaPose	Det. & Seg.	Estimates the 6-DoF pose of known CAD-model objects via one-correspondence matching.	RGB image; CAD model templates	6-D pose (R, t)	6-DoF pose of known object needed	On-demand	(Nguyen et al., 2024)
T09	CenterPoint	Det. & Seg.	Center-based 3-D object detection and multi-object tracking on LiDAR point clouds.	3-D point cloud	3-D bounding boxes; tracking IDs; velocities	3-D detection and tracking needed	On-demand	(Yin et al., 2021)
T10	Cutie	Det. & Seg.	Video object segmentation with long-term memory; tracks and segments target from first-frame mask.	Video frames; initial mask	Per-frame segmentation masks	Continuous video tracking needed	Continuous	(Cheng et al., 2023)
Localization & World Modeling								
T11	Open-Fusion	Loc. & World	Builds a real-time open-vocabulary queryable 3-D semantic field from RGB-D sequences.	RGB-D sequence; camera poses; query text	Open-vocabulary 3-D semantic map	World model refresh or semantic query needed	Map update + query	(Yamazaki et al., 2024)
T12	Hydra	Loc. & World	Organises a 3-D metric map into a room-region-object hierarchy for high-level spatial reasoning.	3-D map; semantic observations; robot poses	Hierarchical 3-D scene graph	High-level planning or semantic query	Background + query	(Hughes et al., 2022)
T13	DUSt3R	Loc. & World	Reconstructs dense 3-D point maps and camera poses from an unconstrained image collection without prior calibration.	Unconstrained RGB images	3-D point maps; camera poses	3-D scene reconstruction from unposed images needed	On-demand	(Wang et al., 2024b)
T14	ZoeDepth	Loc. & World	Predicts metric-scale depth from a single RGB image using relative-to-metric transfer; provides geometric prior for grasping and navigation.	RGB image / batched tensor	Metric depth map; 16-bit depth image; colorized visualization	Depth sensor absent; downstream module needs geometry	On-demand	(Bhat et al., 2023)

Continued on next page

Table 5 – continued from previous page

ID	Tool Name	Capability Unit	Description	Input	Output	Trigger Condition	Mode	Source
T15	`build_query_3d_occupancy_map`	Loc. & World	Fuses depth / point-cloud streams into an OctoMap octree encoding occupied, free, and unknown voxels for navigation and arm workspace planning.	Depth stream; sensor pose; map resolution	Octree map; occupied / free / unknown voxel queries	3-D obstacle or free-space query needed	Continuous update	(Hornung et al., 2013)
Closed-loop Execution, Recovery & Safety								
T16	`TAPIR`	Closed-loop Exec.	Lightweight point-tracker that outputs up-to-date 2-D coordinates of a displaced target, driving visual servo without heavy pose estimation.	Query point; live video stream	Tracked point; occlusion flag	Arm approaching target; target may have moved	Continuous stream	(Doersch et al., 2023)
T17	`R3M`	Closed-loop Exec.	Computes cross-modal distance between post-action frame features and language instruction; outputs a Boolean completion assertion.	Post-action frame; goal instruction text	Completion score; completion flag	Discrete action (grasp, place) just executed	On-demand	(Nair et al., 2022)
T18	`VIRF_Main_Loop`	Closed-loop Exec.	Orchestrates a plan-verify-diagnose-correct closed loop; invokes LLM planner and safety verifier iteratively until a safe plan is found or task is rejected.	User instruction; world knowledge core	Safe plan; task rejection signal	Embodied task instruction received	Continuous loop	(Wu et al., 2026)
Task-to-Action Bridging								
T19	`AnyGrasp`	Task-to-Action	Translates coarse 2-D semantic intent into a precise 6-DoF gripper pose from point cloud input; bridges semantic and kinematic gap.	Goal pose; joint state; scene point cloud	6-DoF pose; gripper width; confidence	VLM locks target; physical grasp action to be generated	On-demand	(Fang et al., 2023)
T20	`Contact-GraspNet`	Task-to-Action	Regresses 6-DoF gripper pose and width directly from a cluttered point cloud; refines semantic grasp to physical contact mechanics.	Cluttered scene depth / point cloud; optional segmap	6-DoF grasp candidate set; contact point distribution	Dense-packing grasp planning triggered	On-demand	(Sundermeyer et al., 2021)
T21	`navigate_to_goal_pose`	Task-to-Action	Translates a high-level goal pose into a navigation action stream via Nav2 stack; handles path planning and obstacle avoidance.	Goal pose; map; obstacle state	Path; navigation action stream	High-level navigation goal pose provided	On-demand + exec.	(Macenski et al., 2020)
T22	`plan_collision_free_manipulation`	Task-to-Action	Generates a collision-free joint trajectory for manipulation via MoveIt 2; resolves goal pose into executable waypoints.	Goal pose; robot state; collision scene	Collision-free joint trajectory	Manipulation goal pose and grasp intent available	On-demand	(MoveIt Contributors, 2024)

E EMBODIEDTOOLBENCH DESIGN

E.1 TASK 1: TOOL-NEED RECOGNITION

Data Collection and Annotation. The dataset for Tool-Need Recognition consists of two categories of samples: positive instances (requiring tool invocation) and negative instances (solvable without tools). Positive instances are directly sourced from complete embodied interaction trajectories: decision states (l, τ_t, o_t, L) in which a tool invocation occurs are labeled $u^\star = 1$. Negative instances are collected from two types of scenarios:

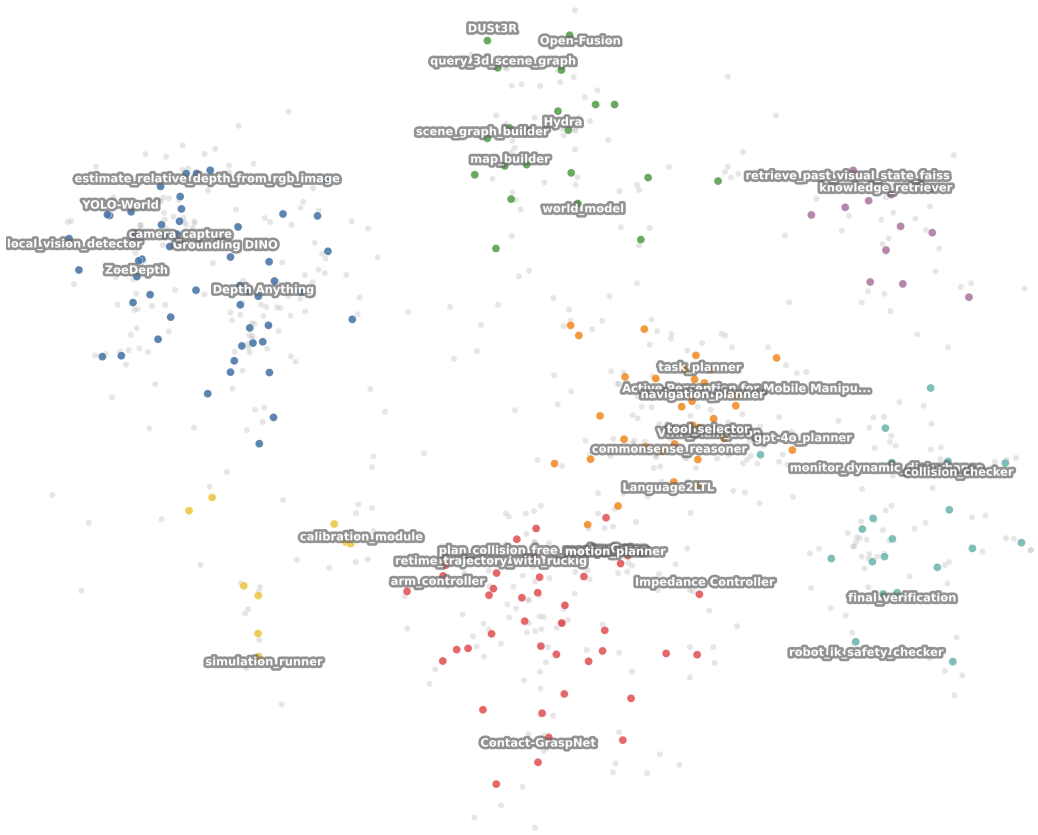

Figure 6: Embodied tool embedding visualization.

1. **Directly solvable states**, where the model can produce a valid action based solely on the current observation o_t and interaction history τ_t, without resorting to any external tool;
2. **Tool-redundant states**, where the candidate tool set L is non-empty, yet no tool invocation is required at the current stage of the task.

Class Balance. To prevent systematic prediction bias, we maintain a positive-to-negative sample ratio of 1:1 and apply stratified sampling across task types (navigation, planning, and manipulation), ensuring that each scenario is uniformly represented in the dataset.

Distractor Difficulty Control. To increase the difficulty of negative instances, we deliberately introduce *tool-inducing* negative samples: these samples contain task descriptions with keywords strongly associated with tool functionality (*e.g.*, "detect", "grasp", "navigate to"), yet based on the current observation and interaction history, no tool invocation is actually required. This design prevents models from relying solely on superficial textual cues to make decisions.

E.2 Task 2: Tool Selection

Candidate Tool Set Construction. For each query q, the corresponding candidate tool set L_t consists of four tools: one ground-truth tool $z^\star$ and three distractors. Distractor tools are selected according to the following hierarchical strategy:

- **Semantically Similar Distractors (Hard Negatives).** We employ the `text-embedding-ada-002` model to embed the descriptions of all tools in the tool library $\mathcal{Z}$, compute the cosine similarity between each tool and $z^\star$, and select the top-$(n-1)$ most similar tools as the candidate distractor pool. These tools are functionally

close to $z^\star$ in their descriptions, effectively preventing models from completing the selection task via simple keyword matching.

- **Uniqueness of the Ground-Truth Answer.** Filtering distractors by semantic similarity alone is insufficient, as some similar tools may functionally overlap with $z^\star$, potentially allowing multiple tools to accomplish the current task and thereby undermining annotation uniqueness. To address this, we perform a **precondition check** for each candidate distractor: if a tool z' has its preconditions satisfiable under the current state (l, τ_t, o_t) and its effects can fulfill the task objective, it is excluded from the distractor pool. Only tools that **cannot be successfully executed in the current state** due to precondition constraints are retained as valid distractors.
- **Intra-category and Inter-category Distractor Mix.** Among the three final distractors, at least two are intra-category distractors drawn from the same tool category as $z^\star$ (*e.g.*, all from the perception, manipulation, or navigation category), and at most one is an inter-category distractor from a different category. This design ensures that the evaluation tests both the model's ability to distinguish fine-grained functional differences and its capacity to filter out irrelevant tools.

Example. Consider the task "detect the target object on the table", where the ground-truth tool is `ObjectDetector` (supporting object detection from the current viewpoint). Distractors may include: `SceneSegmentor` (semantic segmentation; functionally similar but with output incompatible with subsequent actions), `DepthEstimator` (depth estimation; in the same perception category but does not satisfy the classification requirement of the task), and `GraspPlanner` (from the manipulation category; its precondition requires a known target location, which is not satisfied in the current state).

E.3 Task 3: Tool Execution

Stage 1 (Query Construction) Data Design. Each Stage 1 sample contains the current task context (l, τ_t, o_t) and the complete specification of the designated tool $z^\star$ (API name, parameter list, parameter types, and value ranges). The reference query $x^\star$ is constructed by human annotators based on the actual environment state and is automatically verified by a format validator ($\mathrm{Valid}(x^\star, z^\star) = 1$). To increase difficulty, we specifically select tool invocation scenarios where **parameter values depend on the current observation**, such as coordinates or object IDs that must be read from o_t, rather than simple invocations that can rely on default values.

Stage 2 (Output Comprehension) Data Design. Stage 2 samples extend Stage 1 by appending the real tool output y (obtained from actual tool execution), requiring the model to predict the next action $\hat{a}$. The reference action $a^\star$ is likewise sourced from expert trajectories and validated through semantic equivalence matching (Match) to accommodate variations in action descriptions. We ensure that the output format of y is diverse, including structured JSON, natural language descriptions, and numerical results, so as to comprehensively evaluate the model's ability to interpret and utilize heterogeneous tool outputs.

Joint Evaluation Logic for TUSR. Since TUSR requires both Stage 1 and Stage 2 to succeed, the dataset construction deliberately ensures that the error patterns of the two stages are independent of each other: Stage 1 errors arise primarily from failures in parameter construction, while Stage 2 errors stem mainly from misinterpretation of tool outputs. There is no systematic co-occurrence bias between the two types of failures.

E.4 Task 4: Tool-Chain Composition

Source of Tool Chains. The reference tool sequence $P^\star = (z_1^\star, \ldots, z_K^\star)$ in the Tool-Chain Composition task is derived from consecutive tool invocation segments in complete expert trajectories, manually annotated as the **minimal tool set** required to accomplish a specific sub-goal. Trajectory segments containing redundant tool invocations are excluded to ensure the minimality of $S^\star$.

Dependency Constraint Annotation. The execution order constraint set $\mathcal{C}^\star = \{(z_a, z_b)\}$ is constructed from two types of dependency relations:

1. **Data dependencies**: the input parameters of tool z_b are derived from the output of tool z_a, forming a direct dataflow dependency;
2. **State dependencies**: the preconditions of z_b require the environmental state produced by the execution of z_a (*e.g.*, `GraspPlanner` can only plan a grasping path after `ObjectDetector` has successfully localized the target).

All constraints are formally verified to ensure their logical necessity and completeness.

Candidate Tool Set Construction (Distractor Strategy). The candidate tool set L is constructed by augmenting $S^\star$ with two additional categories of distractor tools:

- **Functionally Overlapping Distractors**: tools that are functionally similar to some tool in $S^\star$ but would cause dataflow disruption or state inconsistency if inserted into the current tool chain. The model must identify the inapplicability of such tools in the specific chain context.
- **Redundancy-Inducing Distractors**: tools that are functionally correct but redundant given the minimal tool set constraint (*e.g.*, introducing an additional perception tool when the target location is already known). These distractors are designed to test the model's understanding of the *minimality* principle.

Robustness Design for OCR. To prevent models from achieving inflated OCR scores by memorizing common tool ordering patterns, we deliberately include samples with **non-canonical dependency orders**, *i.e.*, tool invocation sequences that are counter-intuitive yet logically valid (*e.g.*, performing local perception before global planning). Such samples account for approximately 20% of the total Task 4 instances, effectively preventing models from completing the ordering task through prior biases rather than genuine reasoning.

F Statistical Distribution

F.1 EmbodiedTools

Figure 7 summarizes the composition of our collected tool pool. In total, our collection comprises 112 tools distributed across four macro capability groups, spanning the full pipeline of an embodied agent: perception and grounding (36 tools), cognition and state modeling (25), reasoning and planning (27), and execution and control (24). Crucially, no single stage dominates the collection. The four groups are broadly balanced, ensuring that our benchmark does not inadvertently favor agents with narrow, stage-specific strengths.

Beyond this macro-level balance, coverage within each group is equally diverse. Rather than concentrating tools around one or two dominant subcategories, each macro group distributes its tools across six coherent subcategories. For instance, perception and grounding spans low-level geometric understanding, such as depth and 3D geometry, pose and localization, as well as high-level semantic grounding, such as open-vocabulary detection, segmentation and tracking, and spatial grounding. Cognition and state modeling captures the internal state-building capabilities required by embodied agents, including queryable memory, spatio-temporal state modeling, scene graph and world modeling, 3D map and occupancy modeling, physical/contact state estimation, and representation embedding. Reasoning and planning bridges formal symbolic methods, such as formal verification and TAMP knowledge, with planning-oriented capabilities, including active perception, navigation and motion planning, language and intent parsing, and optimization under constraints. Execution and control covers both the kinematic layer, such as inverse kinematics and trajectory generation, and the adaptive control layer, such as force/impedance control, controller dispatch, and closed-loop recovery.

F.2 EmbodiedToolBench

Table 6 summarizes the statistical coverage of EmbodiedToolBench. The benchmark comprises four complementary tool-use tasks, each consisting of 100 samples spanning four distinct embodied environments, yielding a total of 400 samples with consistent cross-environment coverage. The four tasks are carefully structured to reflect a progressive hierarchy of cognitive demand, from binary

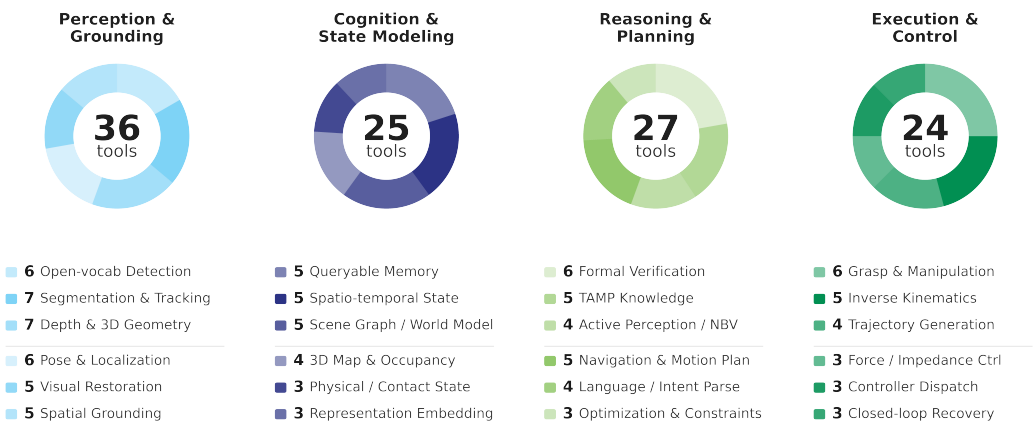

Figure 7: Distribution of tools across macro capability groups and fine-grained subcategories.

tool-presence judgments in *Tool-Need Recognition* to ordered tool-chain composition in *Tool-Chain Composition*, with each task containing samples of varying difficulty levels to ensure fine-grained evaluation within each capability tier. Each subset is also designed with broad and diverse coverage across multiple dimensions. *Tool-Need Recognition* and *Tool Selection* each cover over 60 candidate tools, while *Tool-Chain Composition* encompasses as many as 112 candidates alongside 51 gold tools, reflecting the rich combinatorial space required for multi-step reasoning. The average history context ranges from 1.06 to 2.26 turns with 1.44 to 2.02 accompanying images per sample, providing graded multimodal contextual signals across tasks. These statistics suggest that EMBODIEDTOOLBENCH provides a comprehensive and well-structured testbed that spans a wide spectrum of embodied tool-use scenarios, varying systematically in task complexity, tool space coverage, and contextual richness.

Table 6: Overview of EmbodiedToolBench. Each subset contains 100 samples across four embodied environments. Tasks progress from binary tool-awareness decisions to ordered tool-chain composition. **Diff.** denotes task difficulty, and "–" indicates no fixed candidate set for Tool Usage.

Task	Samples	Envs.	Diff.	Task Types	Gold Tools	Candidate Tools	Avg. History Turns	Avg. History Images
Tool-Need Recognition	100	4	2	1	36	62	2.02	2.02
Tool Selection	100	4	3	1	35	66	2.26	1.62
Tool Execution	100	4	3	2	25	–	1.65	1.60
Tool-Chain Composition	100	4	3	2	51	112	1.06	1.44

Figure 8 further illustrates the dataset distribution across difficulty levels and embodied environments. The difficulty profiles vary meaningfully across tasks, reflecting the distinct cognitive demands of each. *Tool-Need Recognition* is heavily skewed toward easy instances (86%), consistent with its binary decision nature. In contrast, *Tool Selection* achieves a well-balanced distribution across easy, medium, and hard cases (35%/33%/32%), ensuring comprehensive coverage of selection complexity. *Tool Execution* and *Tool-Chain Composition* are predominantly composed of medium and hard instances (87% and 71%, respectively), reflecting the greater contextual and compositional demands these tasks impose. Across all four tasks, the difficulty design ensures that no single difficulty level dominates the benchmark as a whole, providing a balanced and challenging evaluation suite. The environment distribution is similarly well-balanced. *Tool-Need Recognition* draws from all four environments, with EB-ALFRED contributing the largest share (38%). The remaining three tasks exhibit a more even spread across EB-ALFRED, EB-Habitat, EB-Navigation, and EB-Manipulation, with no single environment accounting for more than 35% in any task. Together, these distributions confirm that EMBODIEDTOOLBENCH is designed with both deliberate difficulty stratification and broad environmental diversity, enabling a rigorous and representative evaluation of embodied tool-use capabilities.

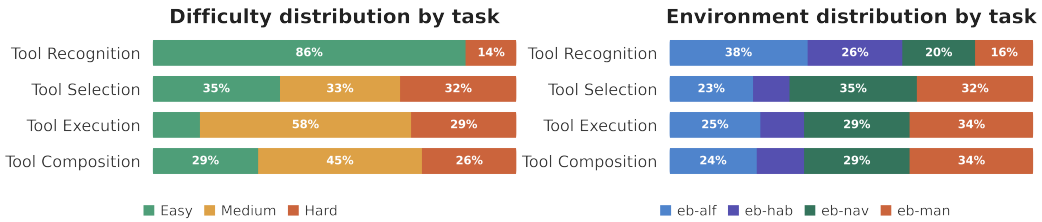

Figure 8: Distribution of EmbodiedToolBench samples across tasks, environments, and tool-use dimensions.

Table 7: Details of the models used in our experiments. **Abbr.** denotes the abbreviated model name used throughout the paper. †Exact parameter counts are not publicly disclosed by the provider. ‡Mixture-of-Experts (MoE) architecture; only 3 B parameters are activated per forward pass.

Model Identifier	**Abbr.**	**Provider**	**Parameter Scale**
Closed-Source Models			
`gpt-4o`	GPT-4o	OpenAI	Not disclosed†
`gpt-5-chat`	GPT-5	OpenAI	Not disclosed†
`claude-3-7-sonnet-20250219`	Claude-3.7	Anthropic	Not disclosed†
`claude-sonnet-4-6`	Claude-4.6	Anthropic	Not disclosed†
Open-Source Models			
`Qwen3-VL-32B-Instruct`	Qwen3-32B	Alibaba	∼32 B (dense)
`Qwen3-VL-8B-Instruct`	Qwen3-8B	Alibaba	∼8 B (dense)
`Qwen2.5-VL-32B-Instruct`	Qwen2.5-32B	Alibaba	32 B (dense)
`Qwen3.5-35B-A3B-Instruct`	Qwen3.5-35B	Alibaba	35 B total, 3 B active‡

G Additional Experiments

G.1 Implementation and Evaluation Details

Compute Resources. The compute resources used in this work consist of API-based evaluation resources and local GPU resources. Closed-source models are evaluated through their official APIs, with an approximate total cost of $3000 for model invocation and benchmark evaluation. Specifically, we use a server with 8 NVIDIA A800 GPUs for tool inference and open-source component deployment.

Model Details. Due to space limitations, we report abbreviated model names in the main paper. Table 7 provides the full model identifiers, the corresponding abbreviations used in our experiments, model providers, and parameter scales. For closed-source models, the exact number of parameters is not publicly disclosed by the corresponding providers.

G.2 Real-world Robotic Setup

We conduct real-world experiments on a tabletop robotic manipulation platform equipped with a 6-DoF bus-servo robotic arm, a parallel robotic gripper, and a Gemini Plus RGB-D camera for visual and depth perception. The camera provides synchronized RGB and depth observations within a working range of 0.25–2.5 m, enabling object localization and scene understanding in tabletop environments. The arm is driven by high-torque smart bus servos and controlled through an STM32-based robotic controller, which handles low-level actuation and servo communication. High-level computation and policy execution are supported by an onboard Jetson control system, while a ROS-compatible control board and USB hub provide communication interfaces for perception, control, and peripheral devices. The whole system is mounted on a rigid metal base with vacuum suction cups to improve stability during manipulation.

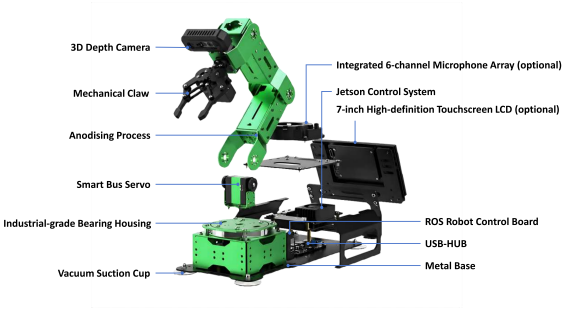

Figure 9: Overview of the high-performance robotic hardware platform.

Table 8: Performance comparison across four embodied benchmarks over three independent runs. Results are reported as mean ± standard deviation. **Bold**: best result per column; underlined: worst result; Avg. Gain : mean improvement of "w/ tool" over "w/o tool".

Model	Setting	EB-ALFRED	EB-Habitat	EB-Navigation	EB-Manipulation	Avg.
Closed-Source MLLMs						
GPT-4o	w/o tool	58.67 ± 5.03	85.33 ± 7.02	55.56 ± 3.47	27.09 ± 3.61	56.66 ± 2.50
	w/ tool	84.67 ± 5.03	93.33 ± 2.31	79.44 ± 0.96	**39.58 ± 7.51**	74.26 ± 0.87
GPT-5	w/o tool	69.33 ± 1.15	82.67 ± 2.31	63.89 ± 3.85	20.14 ± 5.24	59.01 ± 1.02
	w/ tool	86.00 ± 4.00	**98.00 ± 0.00**	**84.44 ± 0.96**	34.03 ± 6.37	**75.62 ± 0.96**
Avg. Gain		↑ 21.33	↑ 11.67	↑ 22.22	↑ 13.20	↑ 17.10
Open-Source MLLMs						
Qwen3-32B	w/o tool	42.00 ± 2.00	74.67 ± 5.77	53.89 ± 3.85	19.44 ± 5.24	47.50 ± 2.22
	w/ tool	**89.33 ± 3.06**	85.33 ± 4.62	80.55 ± 6.31	27.08 ± 2.09	70.58 ± 1.67
Qwen3-8B	w/o tool	14.00 ± 0.00	45.33 ± 2.31	59.44 ± 10.18	17.36 ± 3.18	34.03 ± 3.40
	w/ tool	88.00 ± 0.00	74.67 ± 1.15	79.44 ± 5.09	36.11 ± 1.20	69.56 ± 1.50
Avg. Gain		↑ 60.67	↑ 20.00	↑ 23.33	↑ 13.20	↑ 29.30

G.3 Statistical Significance Analysis

Due to the substantial computational cost of running all experiments in Table 1, estimated at approximately $3,000 per full pass, it is infeasible to repeat every configuration multiple times for rigorous statistical analysis. Instead, we selected the base environment from each task and chose two representative models from both the closed-source group (GPT-4o and GPT-5) and the open-source group (Qwen3-VL-32B and Qwen3-VL-8B) for repeated evaluation. Each selected configuration was executed across three independent runs, with the resulting means and standard deviations reported in Table 8.

The results demonstrate that the performance gains brought by our tool are both substantial and consistent. Across all four benchmarks and all four models, the `w/ tool` setting uniformly outperforms the `w/o tool` baseline without exception. The standard deviations remain small relative to the observed gains. For instance, GPT-4o improves from 58.67 ± 5.03 to 84.67 ± 5.03 on EB-ALFRED, while Qwen3-8B improves from 14.00 ± 0.00 to 88.00 ± 0.00 on the same benchmark. This confirms that the improvements are not artifacts of run-to-run variance, but reflect a stable and reliable benefit. The average gain reaches 60.67 points on EB-ALFRED for open-source models, and closed-source models achieve consistent improvements of over 20 points on both EB-ALFRED and EB-Navigation. Taken together, these findings provide strong statistical evidence that the proposed tool integration yields robust and reproducible performance improvements across diverse models and embodied task settings.

H EXAMPLES OF CASE STUDY

H.1 EMBODIEDTOOL

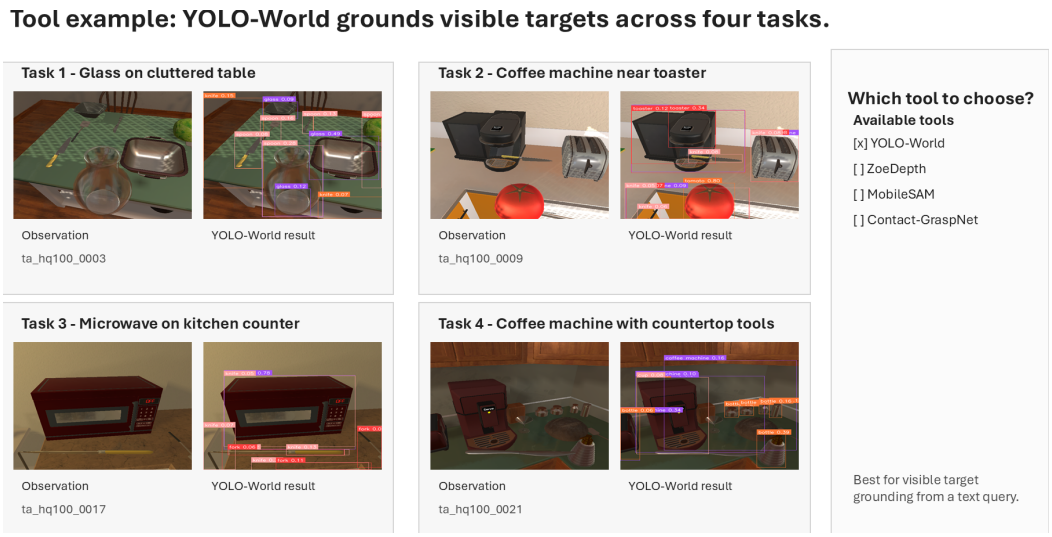

Figure 10: Illustrative example from the tools.

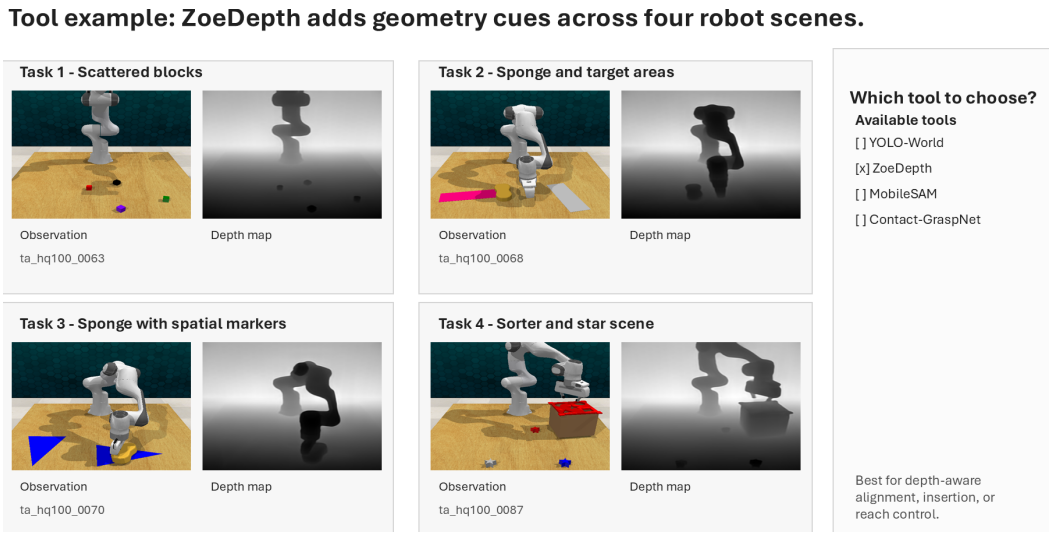

Figure 11: Illustrative example from the tools.

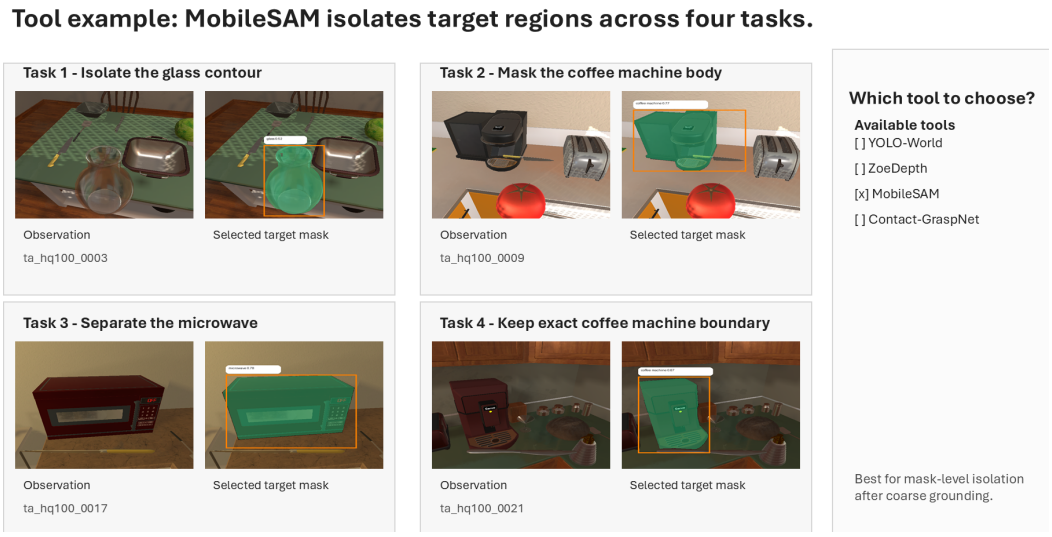

Figure 12: Illustrative example from the tools.

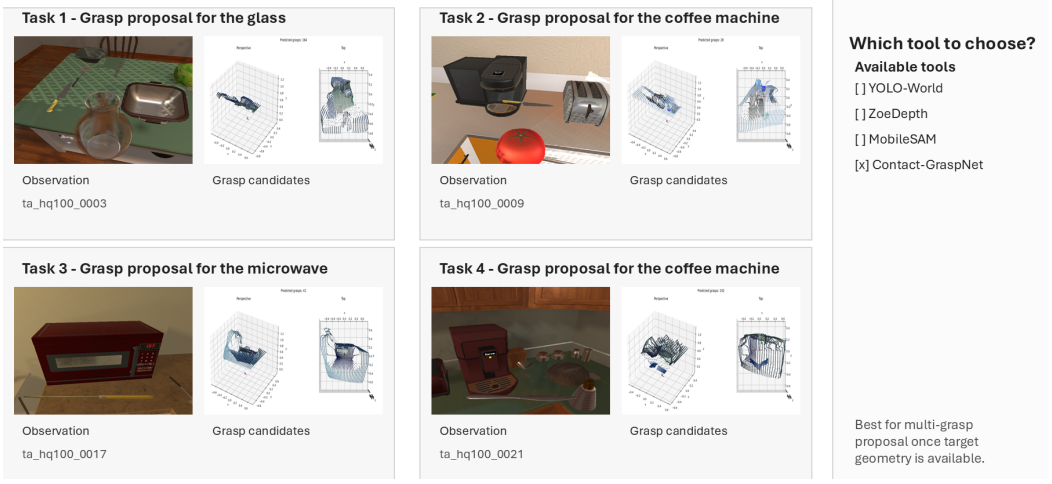

Figure 13: Illustrative example from the tools.

H.2 EMBODIEDTOOLBENCH



Figure 14: Illustrative examples from the tool-awareness evaluation.



Figure 15: Illustrative examples from the tool-selection evaluation.



Figure 16: Illustrative examples from the tool-usage evaluation.



Figure 17: Illustrative examples from the tool-chain composition evaluation.

H.3 EMBODIEDBENCH

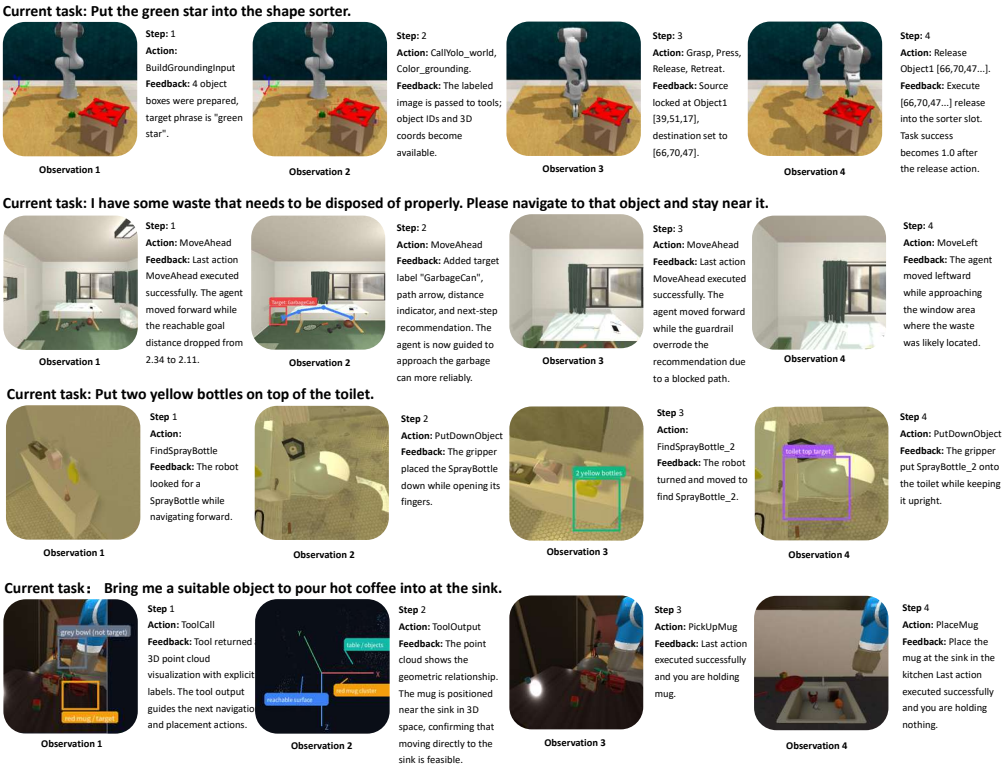

Figure 18: Illustrative examples from the tool-call evaluation.

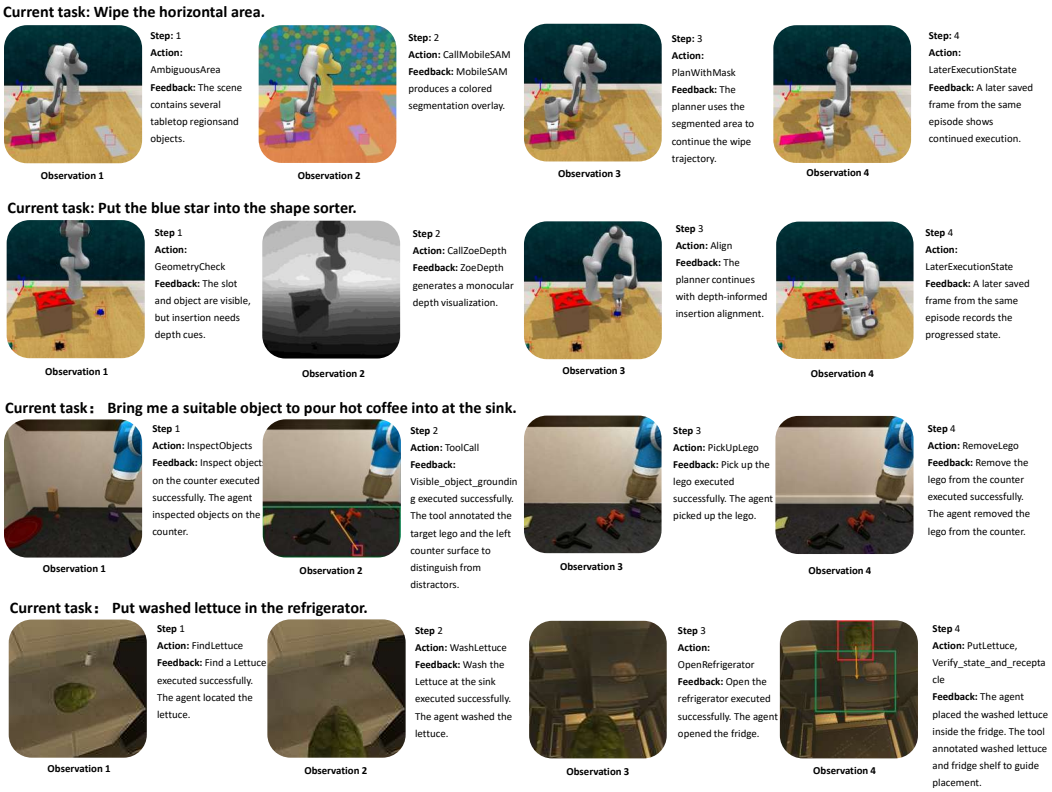

Figure 19: Illustrative examples from the tool-call evaluation.

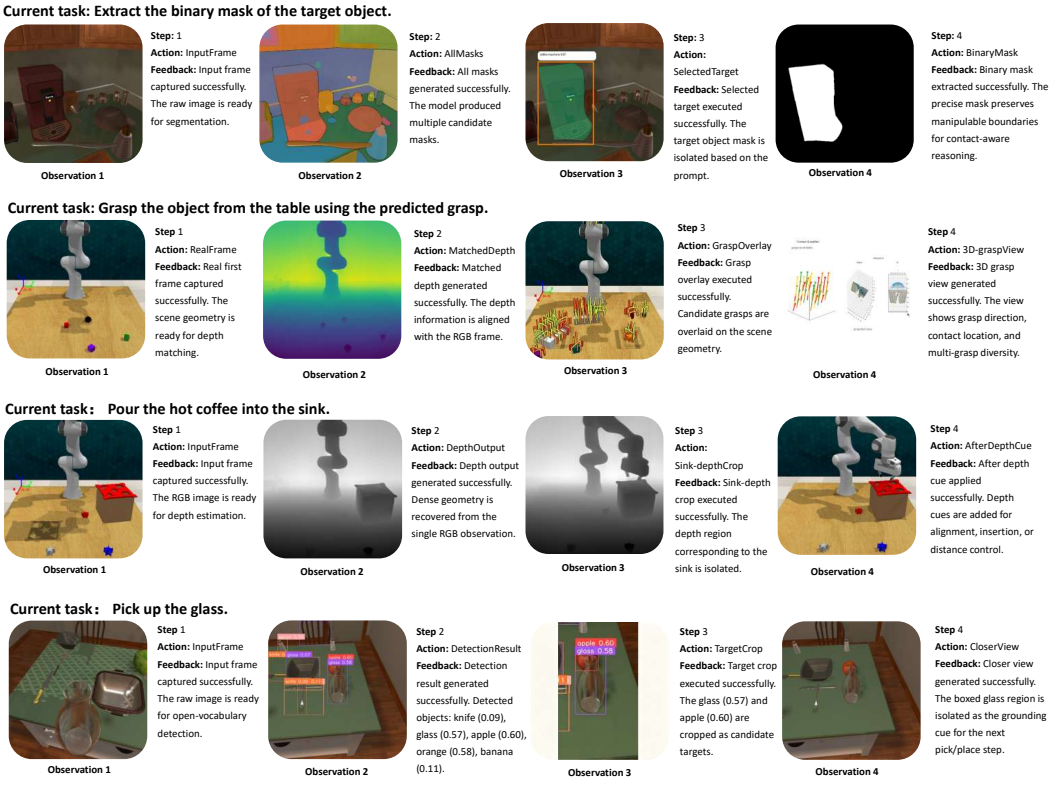

Figure 20: Illustrative examples from the tool-call evaluation.