META-ROBOT: SELF COGNITION AND CAPABILITY SCALING IN AGENTIC ROBOTS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-based robotic agent systems can generate programs for robot control, yet their generalization to new tasks often depends on elaborate scaffolding. Expanding their skill repertoires alone offers limited guidance on how to assess new capabilities or identify what remains beyond their reach. We introduce META-Robot, a framework that couples self cognition with capability scaling by expanding the criteria through which an agent evaluates and develops its skills. At its core, action-level predicates express explicit conditions for assessing execution and provide a structured basis for LLM reasoning. Using the resulting assessments, the agent identifies capability gaps, extends relevant evaluation criteria, and guides the structured expansion of its skills. Proprioceptive feedback further grounds this self-scaling process in the robot's physical state. Experiments in simulation and on real robots demonstrate improved overall task performance, generalization to new tasks, continued expansion of skills and evaluation criteria, and effective use of non-visual proprioceptive signals.

1 INTRODUCTION

Robotic agents have emerged as a promising paradigm for general-purpose robotic manipulation recently with generated code as control policy Liang et al. (2023); Xiao et al. (2026). While these systems Kumar (2026) can adapt to new tasks without training to some extent, their scalability remains limited. When tasks require evaluation criteria, feedback modalities, or skills beyond predefined scaffolds, agents cannot revise these system-level limitations, leading to repeated strategy switching and compounding ineffective modifications that make meta-level task generalization or reliable recovery nearly impossible.

Predicate scaling offers a natural solution by allowing agents to retain and reuse predicates acquired from previous tasks. This accumulation can improve transfer and reduce the need for additional predefined scaffolding. However, scaling predicates alone cannot overcome fixed system scaffolding, as systems cannot have explicit cognition of task and its own capability limits. Recent efforts to improve scalability follow a primary strategy: replacing accumulation of fixed predicates with validated, agent-discovered skills distilled from execution processes Zhang et al. (2026); Lu et al. (2026). This reduces manual engineering and allows past repairs to guide future tasks. However, these skills are accumulated without structured links to their expected effects and required evidence. Consequently, the skill library may continue to grow while the system's understanding of the task and its own capabilities remains unchanged, limiting substantive capability scaling.

The emergence of Neuro-Symbolic Predicates (NSPs) Liang et al. (2025); Athalye et al. (2026) presents a compelling foundation for scaling robot agent system cognition beyond fixed mechanisms. By combining neural perceptual queries with programmatic logic, NSPs can construct abstractions for evaluation, offering a path toward adapting to new evaluation criteria. Nevertheless, extending predicate invention to self cognition and capability scaling in robotic agent systems remains challenging. Predicates must assess not only whether task goals are satisfied, but also whether the current system has sufficient observations, evaluation criteria, and executable skills. We argue that grounding predicates with physical evidence at the action level is crucial, as it allows the agent to exploit available proprioceptive signals when vision is insufficient, construct corresponding skills and evaluation criteria, and integrate them into control and verification, effectively coupling cognitive and capability scaling with evidence-grounded assessment.

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.

To this end, we propose META-Robot, as illustrated in Fig. 1, a system for self cognition and capability scaling that grounds robotic meta-generalization in evidence-based, action-level predicates for both system self-assessment and task evaluation. By recording predicate transitions around skill executions, coupling validated predicates with reusable skills, and orchestrating proprioceptive feedback, META-Robot creates a scalable system for jointly scaling robotic cognition and capabilities.

META-Robot incorporates several key innovations. First, to mitigate the ambiguity between coarse execution logs and actual task-state changes, we introduce a closed-loop execution agent that evaluates each action execution through action-level predicate transitions. The agent records observations, inputs, outputs, and pre- and post-execution predicate states, associating program behavior with changes in robot action. When an error is detected, it performs targeted local repair and re-execution, followed by full-task validation of the revised program. Second, fixed evaluation criteria make failures beyond the coverage of the existing system difficult to diagnose and can lead to unstructured skill accumulation. To address this, META-Robot maintains coupled predicate and skill libraries that expand through validated execution experience. Although this design supports continual knowledge scaling, it also requires precise alignment between the two libraries. META-Robot establishes explicit predicate–skill associations to support grounded skill retrieval, composition, and expansion.

We conduct extensive experiments in both simulation environments [23] and real-world environments to validate the effectiveness of META-Robot. (i) META-Robot achieves stronger overall performance than prior methods, reaching a success rate of 95.8% on the LIBERO-Pro Spatial-Task, compared with 60.0% for other baselines. Crucially, we observe (ii) emergent cognition and capabilities including the scaling of reusable skills and their corresponding evaluation criteria from validated execution experience. We further demonstrate (iii) robust generalization through zero-shot transfer to held-out long-horizon tasks without additional debugging and successful deployment across diverse real-robot tasks. Together with the ablation results, these findings establish META-Robot as a scalable framework for jointly expanding robotic cognition and capability. This paper makes the following contributions:

- We introduce META-Robot, which scales its cognition and capability by jointly improving task policies and the supporting system itself through predicate-guided self-assessment.
- We develop coupled action-level predicate and skill libraries that support continual refinement of evaluation criteria and policies, as well as the generation of new structured criteria and skills beyond the existing repertoire.
- We propose a proprioceptive signal orchestration mechanism that expands the system's capacity for perception, evaluation, and control.

2 Related Work

Agentic Robotic Systems. Recent work increasingly formulates robot control as a closed-loop agentic process rather than a one-shot mapping from language to actions. Systems such as Hi Robot and Being-0 combine high-level vision-language reasoning with lower-level VLA policies or modular skills Shi et al. (2025); Yuan et al. (2025). Robot-use agents further expose capabilities through executable interfaces and stateful harnesses: CaP-X systematically studies Code-as-Policy agents across different abstraction levels and introduces structured execution feedback and visual differencing; Guava introduces semantic tools and multimodal observations; Thea adds persistent scene context and execution evaluation; and Show-Harness enables VLM-driven manipulation through semantic motion primitives, like Cap-X, Guava Fu et al. (2026); Liu et al. (2026). These works establish perception, planning, execution, verification, and recovery as core components of agentic robotics. However, most still rely on predefined tools, observation channels, and evaluators provided by the surrounding harness.

Skill-Scaling Robotic Agents. Beyond closed-loop execution, skill-scaling agents retain and adapt experience across tasks. One line of work reuses prior programs or reflections as memory, including Memory Transfer Planning, Growing with Your Embodied Agent, DAHLIA, and EEAgent Kagaya et al. (2025); Meng et al. (2025a;b); Wang et al. (2026a). Another expands the executable repertoire itself: Playful discovers reusable skills, ASPIRE distills validated repairs, RHO searches repository-level policies, and Embodied Skills, ALRM, and AOR broaden reusable tools or con-

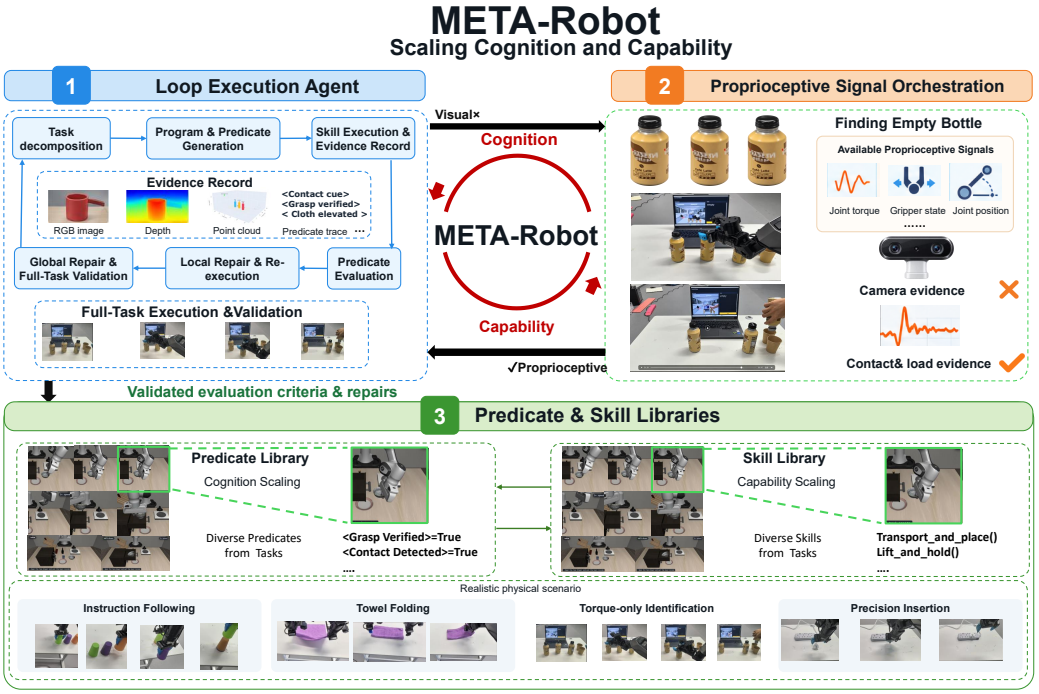


Figure 1: META-Robot starts from a task description. Its framework consists of three components: (1) Loop Execution Agent, which generates programs and predicates, evaluates execution evidence, and repairs failures; (2) Proprioceptive Signal Orchestration, which supplements insufficient visual observations with physical feedback; and (3) coupled Predicate and Skill Libraries, which consolidate validated criteria and repairs for cognitive and capability scaling. Detailed Skill–Predicate analyses are provided in the appendix A.5, with additional simulation and real-robot results reported in the experiments

troller implementations Zhang et al. (2026); Lu et al. (2026); Elmaaroufi et al. (2026); Wang et al. (2026b); Santos et al. (2026); Kumar (2026). Yet capability growth is typically decoupled from the evolution of evaluation criteria and evidence requirements. META-Robot instead couples hierarchical skills with action-level predicates and additional feedback channels, allowing capabilities and the mechanisms used to assess them to scale together.

3 METHOD

We introduce META-Robot, a novel framework for meta-level generalization to new tasks. The META-Robot framework is grounded in the scaling of cognition and capability, where evaluation criteria, feedback acquisition, available signals, and executable skills are jointly expanded, thereby reducing dependence on predefined scaffolding. META-Robot operates directly on action-level predicates throughout the process, instead of coarse execution outcomes, which better connects program behavior with changes in task conditions. Specifically, the closed-loop execution agent records observations, inputs, outputs, and pre- and post-execution predicate states for every skill invocation and subtask end. The predicate-state trace enables the system to consistently assess progress, attribute failures, and verify targeted repairs through local re-execution before full-task validation. We also introduce coupled action-level predicate and Skill Libraries and a proprioceptive signal orchestration mechanism to overcome the limitations of unstructured skill accumulation and vision-only verification. With these components, META-Robot can enrich its task evidence and cognition, consolidate validated repairs as reusable skills, and acquire additional physical evidence when existing observations are insufficient, progressively expanding both system cognition and capability. Fig. 2 illustrates an overview of the META-Robot framework.

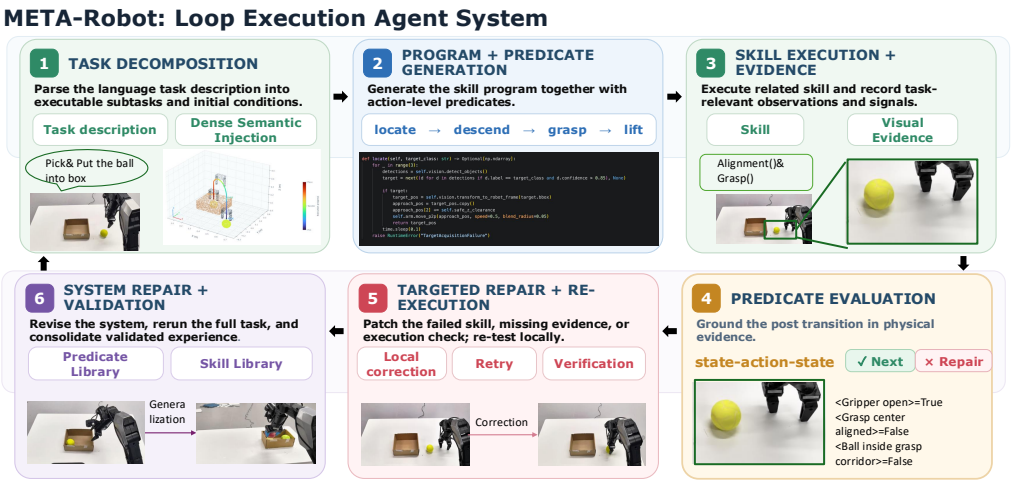


Figure 2: META-Robot Loop Execution Agent. Given a task description, the agent (1) decomposes it into executable subtasks and initial conditions; (2) jointly generates the task program and action-level predicates; (3) executes each skill while collecting task-relevant evidence; (4) evaluates predicate transitions to determine whether execution should continue or be repaired; (5) locally corrects and re-executes the failed skill, missing evidence routine, or verification check; and (6) performs system-level repair and full-task validation, consolidating validated predicates and skills for reuse. This procedure is repeated to form a closed execution–evaluation–repair loop.

3.1 LOOP EXECUTION AGENT SYSTEM

Inspired by software coding agents DeepSeek-AI (2026); Barbaste et al. (2026), we treat evaluation of the execution system itself as a key component of scaling, just as shown in Fig.2. Existing robotic agents often focus on task outcomes or intermediate execution feedback, with limited assessment of the mechanisms responsible for observation and evaluation. When failures arise beyond these mechanisms' coverage—the system's cognition and capability boundaries—repeated iteration alone may be insufficient. To solve this problem, META-Robot's loop execution agent system uses action-level predicates to assess the outcome of each action execution. Given a task description, dense semantic injection Smith et al. (2025) enriches its specification and decomposes it into a subtask trace. For each subtask, the agent generates action-level predicates from the task description and the LLM's own reasoning, establishing explicit criteria for execution assessment. These predicates are evaluated after each skill invocation and at subtask completion, providing fine-grained evidence of whether the intended action conditions have been satisfied.

To recover from an unsatisfied action-level condition, we initiate a bounded local repair loop whenever a required action-level predicate evaluates to False. To restore the expected state transition within a limited number of turns, the repair agent revises the code surrounding the failed action or applies iterative refinement procedures, like Picard iteration Ince (1956) and multi-directional approach search Panic (2026); Ortega & Rheinboldt (2000). To preserve diagnostic evidence when the failure cannot be localized or resolved locally, we record the latest predicate state and continue the remaining execution. To support task-level failure attribution, we consolidate all evaluations into a predicate transition trace. If the overall task fails, a global repair agent analyzes this trace together with execution logs and auxiliary evidence, formulates diagnostic hypotheses, and revises the task program or its supporting observation and verification mechanisms. To validate these system-level modifications, we re-execute the complete task, enabling improvements beyond isolated action-level repairs.

3.2 ACTION-LEVEL PREDICATE LIBRARY & SKILL LIBRARY

META-Robot maintains two coupled libraries: a Skill Library that stores reusable execution knowledge distilled from validated repairs, and an action-level predicate library that stores verifiable action predicates. Skills specify how to execute or repair an interaction, while associated predicates specify the conditions to be satisfied after execution. Predicates and skills are associated: each skill can be associated with multiple predicates, and different skills may share the same predicate.

Formal representation. To couple reusable skills with action-level evaluation criteria, we represent the Skill Library as a forest $\mathcal{L}_S = (\mathcal{V}, \mathcal{E})$, where edges encode parent–child specialization. Each skill is defined as

$$s_v = (f_v, \mathcal{C}_v, g_v, r_v, \mathcal{Q}_v, \pi_v, \mathcal{D}_v)\,, \tag{1}$$

where $(f_v, \mathcal{C}_v, g_v, r_v)$ specify its failure signature, applicability, intent, and repair mechanism, while $(\mathcal{Q}_v, \pi_v, \mathcal{D}_v)$ record its feedback requirements, implementation, and validation evidence.

To organize evaluation criteria consistently, the Predicate Library contains shared action-level predicates $\mathcal{P} = \{\phi_j\}_{j=1}^{M}$ and follows the same categorization as the Skill Library. Their correspondence is

$$\Gamma(v) \subseteq \mathcal{P}, \qquad \mathcal{R} = \{(v, \phi_j) \mid \phi_j \in \Gamma(v)\}, \tag{2}$$

allowing each skill to use multiple predicates and each predicate to be shared across skills.

To capture the effects of a skill invocation, the agent records

$$\mathbf{z}^{-}_{v,t} = [\phi_j(e^{-}_t)]_{\phi_j \in \Gamma(v)}, \qquad \mathbf{z}^{+}_{v,t} = [\phi_j(e^{+}_t)]_{\phi_j \in \Gamma(v)}, \tag{3}$$

where e^{-}_t and e^{+}_t denote evidence collected before and after execution. These transitions associate reusable skills with explicit changes in task conditions.

Parent–child organization and shared predicate indexing. To generalize validated repairs without imposing a predefined taxonomy, META-Robot organizes skills into parent–child relations. Parent nodes capture general repair principles, while child nodes specialize them to particular tasks, objects, sensing conditions, or embodiments. Let $\mathcal{A}(v)$ denote the contexts satisfying $\mathcal{C}_v$. A parent p and child c satisfy

$$\mathcal{A}(c) \subseteq \mathcal{A}(p), \qquad r_c = \mathrm{Specialize}(r_p; \delta_c), \tag{4}$$

where δ_c introduces additional conditions or implementation details. This relation represents semantic specialization rather than guaranteed success over the entire parent domain.

To align evaluation criteria with skill specialization, the Predicate Library is indexed by the same hierarchy. For a skill category b,

$$\mathcal{P}_b = \bigcup_{v \in \mathrm{Subtree}(b)} \Gamma(v) \tag{5}$$

To avoid duplication, predicates with identical semantics and evidence requirements are shared across skills and branches. A specialized skill inherits applicable predicates and introduces new ones only when the existing predicate set cannot characterize the interaction. For example, torque-guided thin-towel grasping may specialize contact-aware grasping, reuse contact detected, and introduce cloth elevated and grasp verified. Contact and successful acquisition remain separate predicates because they describe distinct physical conditions.

Incremental consolidation. To convert validated repairs into reusable capabilities, the agent distills each repair ρ into a candidate skill $\hat{s}$ and compares its failure signature, applicability conditions, execution intent, and corrective mechanism with related library entries. Equivalent repairs are merged, specialized repairs become child nodes, and distinct mechanisms create new branches. Changes to applicability or implementation semantics are revalidated before admission.

To preserve consistent evaluation, the agent jointly consolidates the predicates associated with the new skill. Existing predicates are reused when their semantics and evidence requirements match; otherwise, new predicates are proposed, validated, and added to $\mathcal{P}$. The corresponding mapping $\Gamma(v)$ is then updated. Reorganizing the skill hierarchy changes only predicate indexing, while shared predicate identities remain unchanged.

To transfer validated knowledge to a new context x, the agent retrieves the most specific applicable skill whose feedback requirements are available:

$$v^\star \in \arg \max_{\substack{v \in \mathcal{V} \\ x \in \mathcal{A}(v) \\ \mathcal{Q}_v \subseteq \mathcal{Q}(x)}} \operatorname{depth}(v). \tag{6}$$

The selected skill is retrieved together with $\Gamma(v^\star)$, providing both execution guidance and action-level evaluation criteria. If no specialized skill applies, a compatible parent guides the generation and validation of a new implementation. The whole algorithm flow pseudocode can be seen in appendix A.3

3.3 PROPRIOCEPTIVE SIGNAL ORCHESTRATION MECHANISM

To enable the agent to reason about the robot's own perceptual capabilities, META-Robot introduces a mechanism for the synergistic use of proprioceptive signals. Instead of relying on fixed observation interfaces, the system identifies available proprioceptive signals—such as joint states, motor currents, force/torque measurements, and contact-related feedback—by examining specifications tied to the robot's embodiment (e.g., simulator configurations, controller parameters, and robot description files like URDF). The system then associates these signals with their physical semantics and utilizes them as actionable sources of evidence during execution.

When a task indicates that visual information alone is insufficient to achieve the goal, the system incorporates these identified signals into the execution and verification processes. Validated proprioceptive applications are refined and categorized within the skill library (e.g., contact-aware grasping, slip detection, or force-sensitive manipulation), while the predicate library is expanded to include corresponding action-level criteria that define the state conditions required before and after an interaction. For instance, the assessment of a grasping action is no longer limited to a visual check of the gripper's closed state; it can also be determined based on contact conditions, motor load, or force stability. This is particularly crucial for failure cases that are visually ambiguous (such as "near-miss" grasps) or for tasks involving deformable and thin objects (like folding a towel, just as shown in Fig.3), where image evidence alone often fails to confirm whether stable contact or effective manipulation has been achieved.

Through this process, META-Robot moves beyond treating proprioception merely as a static auxiliary input; instead, it continuously explores how embodiment-specific signals can support novel execution strategies and verification criteria, thereby improving the cognition and capabilities of the system. As these signal-based skills and predicate-based criteria accumulate, the system gains access to interaction state information that is difficult to infer visually, leading to significant improvements in meta-level generalization to more complex tasks.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluated META-Robot on LIBERO-Pro Zhou et al. (2025) with π_0 Black et al. (2024), $\pi_{0.5}$, Intelligence et al. (2025), OpenVLA Kim et al. (2024), etc, in simulation environment and on the Galaxea R1 Lite Galaxea Dynamics (n.d.) platform in real-world experiments. For simulation, we used GPT-6 Astra OpenAI (2026b) and DeepSeek-V4-Flash Xu et al. (2026) as coding agent. We tested several different code models to support, please see appendix A.3. Given a natural-language task description, the agent generates executable Python programs and iteratively refines them using action-level predicate state-transition traces. The predicate and skill libraries provide reusable action evaluation criteria and implementations to support policy generation and repair.

For real-world experiments, we use GPT-5.6-sol OpenAI (2026a) on R1 Lite. Our tasks cover basic pick-and-place, sequential multi-object manipulation, instruction following, precise plug insertion, thin-towel folding, and breakfast-item stacking. We additionally consider identifying an empty bottle among three visually identical opaque bottles containing different amounts of water, using joint-torque feedback to distinguish their loads. The system combines head-mounted camera observations with available proprioceptive signals, including joint torques and gripper feedback. These signals support interaction judgments that are difficult to establish visually, such as detecting contact with

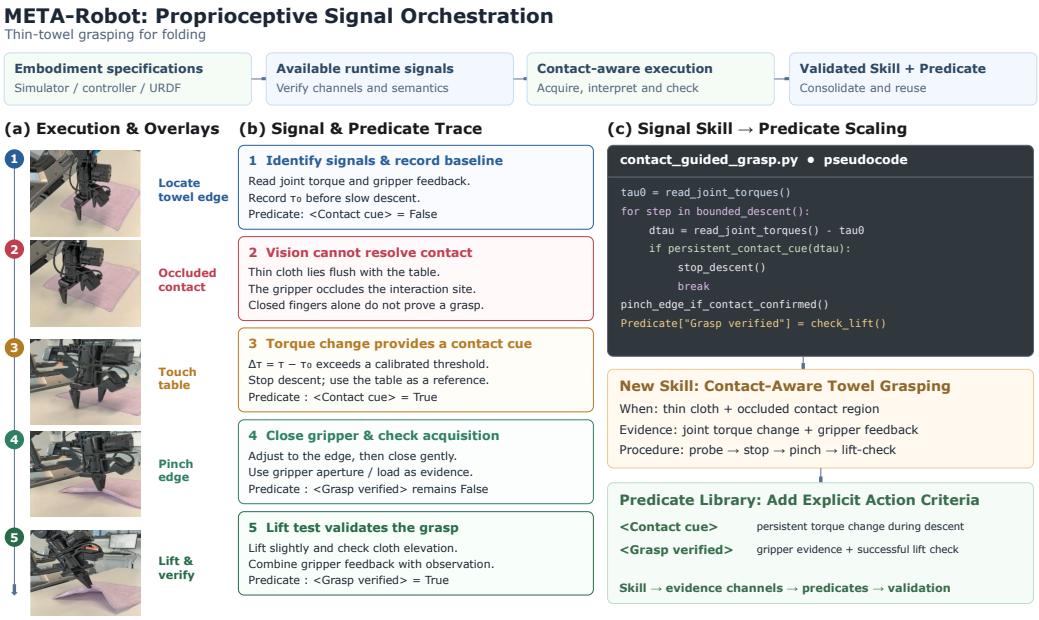


Figure 3: Proprioceptive Signal Orchestration Mechanism Example: in thin-towel grasping, torque change provides a contact cue, while gripper feedback and a lift check verify successful acquisition to complement insufficient visual evidence during execution and verification. Validated signal usage is then consolidated into a reusable skill together with corresponding action predicates, jointly expanding sensing, execution, and verification capabilities

the tabletop when grasping a thin towel. Across both settings, META-Robot evaluates execution through predicate checks after skill calls and subtasks, attempts bounded local repairs, and uses accumulated predicate traces to guide system-level revision and full-task re-execution. Adaptation is performed through program and library updates without model parameter training.

4.2 Main Evaluation Results

Tab. 1 compares META-Robot with prior methods across the Object, Goal, and Spatial suites under position and task perturbations, evaluated over 50 seeds. META-Robot achieves its largest gains on Spatial, where the GPT-6 variant reaches 89.25% on Pos and 95.8% on Task, outperforming ASPIRE by 38.25 and 35.8 percentage points, respectively. On Goal–Task, DeepSeek-V4-Flash and GPT-6 achieve 64.6% and 69.8%, compared with ASPIRE's 45.0%. Performance also remains strong on Object, with GPT-6 reaching 99.6% on Pos and 99.0% on Task. On Goal–Pos, META-Robot with GPT-6 achieves 85.8%, outperforming ASPIRE's 81.0% by 4.8 percentage points. Overall, both META-Robot variants perform competitively across the benchmark, while GPT-6 consistently achieves the stronger results, particularly on spatial perturbations and Goal–Task. Per-task success rates are reported in the appendix.

Table 1: LIBERO-Pro evaluates the success rate of each model across Object, Goal, and Spatial tasks

Category	Task Type	OpenVLA	π_0	$\pi_{0.5}$	CaP-Agent0	ASPIRE	Playful / RATS	META-Robot (DeepSeek-V4-Flash)	META-Robot (GPT-6 Astra)
Object	Pos	0.00	0.00	17.00	22.00	98.00	61.00	98.00	99.60
	Task	0.00	0.00	1.00	18.00	95.00	63.00	88.60	99.00
Goal	Pos	0.00	0.00	38.00	26.00	81.00	43.00	72.20	85.80
	Task	0.00	0.00	0.00	17.00	45.00	36.00	64.60	69.80
Spatial	Pos	0.00	0.00	20.00	12.00	51.00	29.00	83.00	89.25
	Task	0.00	0.00	1.00	14.00	60.00	31.00	88.40	95.80

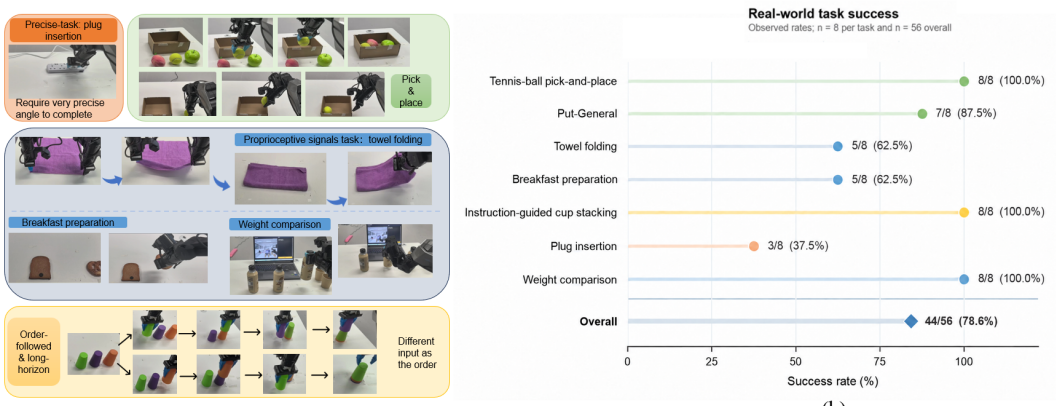


Figure 4: (a) illustrates the robot performing diverse real-world tasks, including precise insertion, pick-and-place, proprioceptive signals task and long-horizon instruction following. (b) shows the success rate of these tasks.

4.3 Real-Robot Evaluation

Fig. 4 reports META-Robot's performance across seven real-world tasks (with eight trials per task) was as follows: the system succeeded in 44 out of 56 total trials (a success rate of 78.6%). Specifically, the tennis ball pick-and-place, instruction-guided cup stacking, and weight comparison tasks all achieved a 100% success rate, while the "Put-General" task had a success rate of 87.5%. The success rates for towel folding and breakfast preparation dropped to 62.5% (notably, due to the thin, soft, brittle, and slippery nature of towels and cookies, the success rate for these two tasks was 0% without proprioceptive feedback, but improved significantly once such feedback was incorporated; see the appendix for a detailed analysis). The plug insertion task proved the most challenging, with a success rate of only 37.5%. These results indicate that while the robot excels at tasks such as pick-and-place, instruction following, and weight comparison, manipulating deformable objects and achieving contact alignment remain its primary limitations. More details can be seen in A.1.

4.4 Zero-Shot Transfer to Unseen Tasks

We evaluate transfer to LIBERO-Pro Long using fixed predicate and skill libraries, without additional debugging or library updates, as shown in Fig. 5. We randomly retain {25%, 50%, 75%, 100%} of the skills and their associated predicates. On Task perturbations, success is 0.0%, 31.8%, 40.2%, and 37.0% at the four retention ratios, respectively. The corresponding results on Swap perturbations are 0.0%, 21.2%, 8.0%, and 8.2%. These results show that accumulated predicate–skill knowledge supports partial zero-shot transfer, but its benefit varies substantially with the perturbation type and retained library composition. In particular, the non-monotonic Task trend and the weaker Swap results indicate that retaining more entries does not necessarily improve retrieval or execution. However, because each ratio is evaluated using a randomly retained subset, the experiment cannot determine whether the differences arise from redundant context, missing task-relevant entries, or sampling variation. Repeated subsampling across multiple library masks is therefore required before interpreting 75% retention as an optimal trade-off. Per-task results in Figure. A.2 further show that gains are concentrated in a subset of tasks, suggesting that

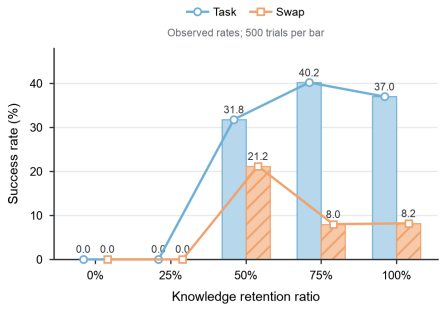


Figure 5: Zero-shot transfer to LIBERO-Pro Long under different library retention ratios. Success rates are averaged over ten tasks.

transfer depends on whether the retained skills and predicates cover each task's required conditions rather than on library size alone.

4.5 Ablation Studies

4.5.1 Action-level Predicate Ablation Test

As shown in Tab. 2, we evaluate predicate contributions by removing them across all six LIBERO-Pro splits. On task perturbations, success falls from 99.0% to 85.0% for Object, 69.8% to 56.0% for Goal, and 95.8% to 76.0% for Spatial. On Pos perturbations, it falls from 99.6% to 91.0%, 85.8% to 69.0%, and 89.25% to 72.0%, respectively. These consistent drops show that action-level predicates support execution assessment, failure localization, and targeted repair. The Spatial-Pos comparison warrants caution because the full-system result covers eight of the available tasks.

Table 2: Ablation Study of META-Robot across Different Tasks on LIBERO-Pro: We compare the task success rates (%) of full META-Robot and its variant without action-level predicates across the Object, Goal, and Spatial suites under task and position perturbations. Removing predicates consistently reduces performance across all six settings, demonstrating their contribution to execution assessment and targeted repair.

Category	Task Type	Full META-Robot (%)	Without Predicate (%)
Object	Task	99.00	85.00
	Pos	99.60	91.00
Goal	Task	69.80	56.00
	Pos	85.80	69.00
Spatial	Task	95.80	76.00
	Pos	89.25	72.00

4.5.2 Proprioception Ablation test

Libero-Pro lacks tasks that heavily rely on proprioceptive signals. Therefore, we specifically designed a new simulation task involving the grasping of a rigid, thin object for ablation testing; the success rate was 50% without proprioceptive signals and 78% with them, as shown in Fig. 11. The whole process can be divided into 4 steps: 1. The gripper positions itself and approaches the sheet; 2. it descends slowly to locate the point of contact; 3. it uses internal feedback to determine whether the gripper has made contact with the sheet or the tabletop and adjusts the grasping position; 4. it closes the gripper, lifts the sheet, and verifies the success of the grasp. Since the sheet lies flat against the tabletop, the contact point is difficult to determine based on visual data alone; therefore, this process requires assistance from internal system signals. In addition, we conducted a comparative analysis between the biscuit-grasping task performed during the real-world breakfast-making scenario and the task at hand; please refer to the appendix A.1 for further analysis and illustrations regarding this ablation study.

5 Conclusion

This paper presented META-Robot, a self congnition and capbility scaling robotic agent that jointly improves execution, observation, and evaluation of task and system itself. Predicate state-transition traces support failure diagnosis and targeted repair, while coupled Predicate and Skill Libraries consolidate reusable criteria and procedures. Proprioceptive signal orchestration further incorporates physical evidence beyond vision. Experiments on LIBERO-Pro and the Galaxea R1 Lite demonstrate effective manipulation and transfer to held-out long-horizon tasks without additional debugging. The varying transfer performance also shows that more accumulated knowledge does not necessarily yield better generalization, emphasizing the importance of selecting task-relevant experience. More broadly, META-Robot supports joint cognitive and capability scaling by refining not only executable skills, but also the evidence channels and verification criteria used to assess them. These results motivate robotic self scaling through the coordinated refinement of execution, sensing, and evaluation. Future work will study relevance-aware knowledge selection, automatic signal calibration, and broader continual evaluation across tasks, embodiments, and contact-rich settings.

AI use statement

Generative AI was used to assist the research and writing workflow. We used generative AI tools to assist with drafting and editing text, identifying relevant literature, providing feedback on experimental design and evaluation protocols, and examining the interpretation and limitations of the reported results. The authors reviewed the AI-assisted text and suggestions, checked cited sources and empirical claims against the original publications and experimental records, and determined which suggestions to incorporate. The authors take full responsibility for the methodology, implementation, reported results, references, and conclusions of this work, including all content produced with the assistance of generative AI.

REFERENCES

Ashay Athalye, Nishanth Kumar, Tom Silver, Yichao Liang, Jiuguang Wang, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. From pixels to predicates: Learning symbolic world models via pretrained vlms. *IEEE Robotics and Automation Letters*, 2026.

Paul Barbaste, Tristan Darrigol, Germain Vu, and Tom Wiltberger. Harness engineering: Anatomy, architecture, and evolution of coding agents–a source-code study of eleven systems. *arXiv preprint arXiv:2609.00006*, 2026.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*, 2024.

DeepSeek-AI. Deepseek harness: Everything is a plugin. `https://github.com/deepseek-ai/deepseek-harness`, 2026.

Karim Elmaaroufi, Justin Svegliato, Sarunas Kalade, Graham Schelle, Sanjit A Seshia, and Matei Zaharia. Rho: Your coding agent is secretly a roboticist. *arXiv preprint arXiv:2606.16458*, 2026.

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.

Galaxea Dynamics. R1 Lite hardware introduction. Galaxea Dynamics Documentation, n.d. URL `https://deepwiki.com/userguide-galaxea/Product_User_Guide/2.3.1-r1-lite-hardware-introduction`. Accessed September 26, 2026.

Edward L Ince. *Ordinary differential equations*. Courier Corporation, 1956.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Tomoyuki Kagaya, Subramanian Lakshmi, Yuxuan Lou, Thong Jing Yuan, Jayashree Karlekar, Sugiri Pranata, Natsuki Murakami, Akira Kinose, and Yang You. Memory transfer planning: LLM-driven context-aware code adaptation for robot manipulation. *arXiv preprint arXiv:2509.24160*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model, 2024. *URL https://arxiv. org/abs/2406.09246*, 1(2):4, 2024.

Vaishak Kumar. Act-observe-rewrite: Multimodal coding agents as in-context policy learners for robot manipulation. *arXiv preprint arXiv:2603.04466*, 2026.

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 *2023 IEEE International conference on robotics and automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

Yichao Liang, Nishanth Kumar, Hao Tang, Adrian Weller, Joshua B Tenenbaum, Tom Silver, João F Henriques, and Kevin Ellis. Visualpredicator: Learning abstract world models with neuro-symbolic predicates for robot planning. In *International Conference on Learning Representations*, volume 2025, pp. 60416–60444, 2025.

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*, 2026.

Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, et al. Aspire: Agentic/skills discovery for robotics. *arXiv preprint arXiv:2607.00272*, 2026.

Yuan Meng, Zhenguo Sun, Max Fest, Xukun Li, Zhenshan Bing, and Alois Knoll. Growing with your embodied agent: A human-in-the-loop lifelong code generation framework for long-horizon manipulation skills. *arXiv preprint arXiv:2509.18597*, 2025a.

Yuan Meng, Xiangtong Yao, Haihui Ye, Yirui Zhou, Shengqiang Zhang, Zhenguo Sun, Xukun Li, Zhenshan Bing, and Alois Knoll. Embodied long horizon manipulation with closed-loop code generation and incremental few-shot adaptation. *arXiv preprint arXiv:2503.21969*, 2025b.

OpenAI. GPT-5.6 Sol model. OpenAI API documentation, 2026a. URL `https://developers.openai.com/api/docs/models/gpt-5.6-sol`. Accessed September 26, 2026.

OpenAI. GPT-6 Astra model. OpenAI API documentation, 2026b. URL `https://developers.openai.com/api/docs/models/gpt-6-astra`. Accessed September 26, 2026.

James M Ortega and Werner C Rheinboldt. *Iterative solution of nonlinear equations in several variables*. SIAM, 2000.

Stefan R Panic. Directional gradient and curvature approximation via legendre quadrature in unconstrained optimization. *Applied Numerical Mathematics*, 220:294–309, 2026.

Vitor Gaboardi dos Santos, Ibrahim Khadraoui, Ibrahim Farhat, Hamza Yous, Samy Teffahi, and Hakim Hacid. Alrm: Agentic llm for robotic manipulation. *arXiv preprint arXiv:2601.19510*, 2026.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025.

Laura Smith, Alex Irpan, Montserrat Gonzalez Arenas, Sean Kirmani, Dmitry Kalashnikov, Dhruv Shah, and Ted Xiao. Steer: Flexible robotic manipulation via dense language grounding. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16517–16524. IEEE, 2025.

Jianzong Wang, Botao Zhao, Yayun He, Junqing Peng, and Xulong Zhang. Evolvable embodied agent for robotic manipulation via long short-term reflection and optimization. *arXiv preprint arXiv:2604.13533*, 2026a.

Wei Wang, Wenqiao Zhang, Yutong Lin, Yuqian Yuan, Tianwei Lin, Jinhao Mao, Zhenxuan Fan, Mingjian Gao, Yang Dai, Wentong Li, et al. Embodiedskills: A unified framework for orchestrating, training, and deploying vla agents. *arXiv preprint arXiv:2609.01281*, 2026b.

Wenli Xiao, Jia Xie, Tonghe Zhang, Haotian Lin, Letian Fu, Haoru Xue, Jalen Lu, Yi Yang, Cunxi Dai, Zi Wang, et al. Enpire: Agentic robot policy self-improvement in the real world. *arXiv preprint arXiv:2606.19980*, 2026.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Haoqi Yuan, Yu Bai, Yuhui Fu, Bohan Zhou, Yicheng Feng, Xinrun Xu, Yi Zhan, Börje F Karlsson, and Zongqing Lu. Being-0: A humanoid robotic agent with vision-language models and modular skills. *arXiv preprint arXiv:2503.12533*, 2025.

Junyi Zhang, Jiaxin Ge, Hanjun Yoo, Letian Fu, Zihan Yang, Yaowei Liu, Raj Saravanan, Shaofeng Yin, Justin Yu, Dantong Niu, et al. Playful agentic robot learning. *arXiv preprint arXiv:2606.19419*, 2026.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.

A Appendix

A.1 The analysis of each real-robot task

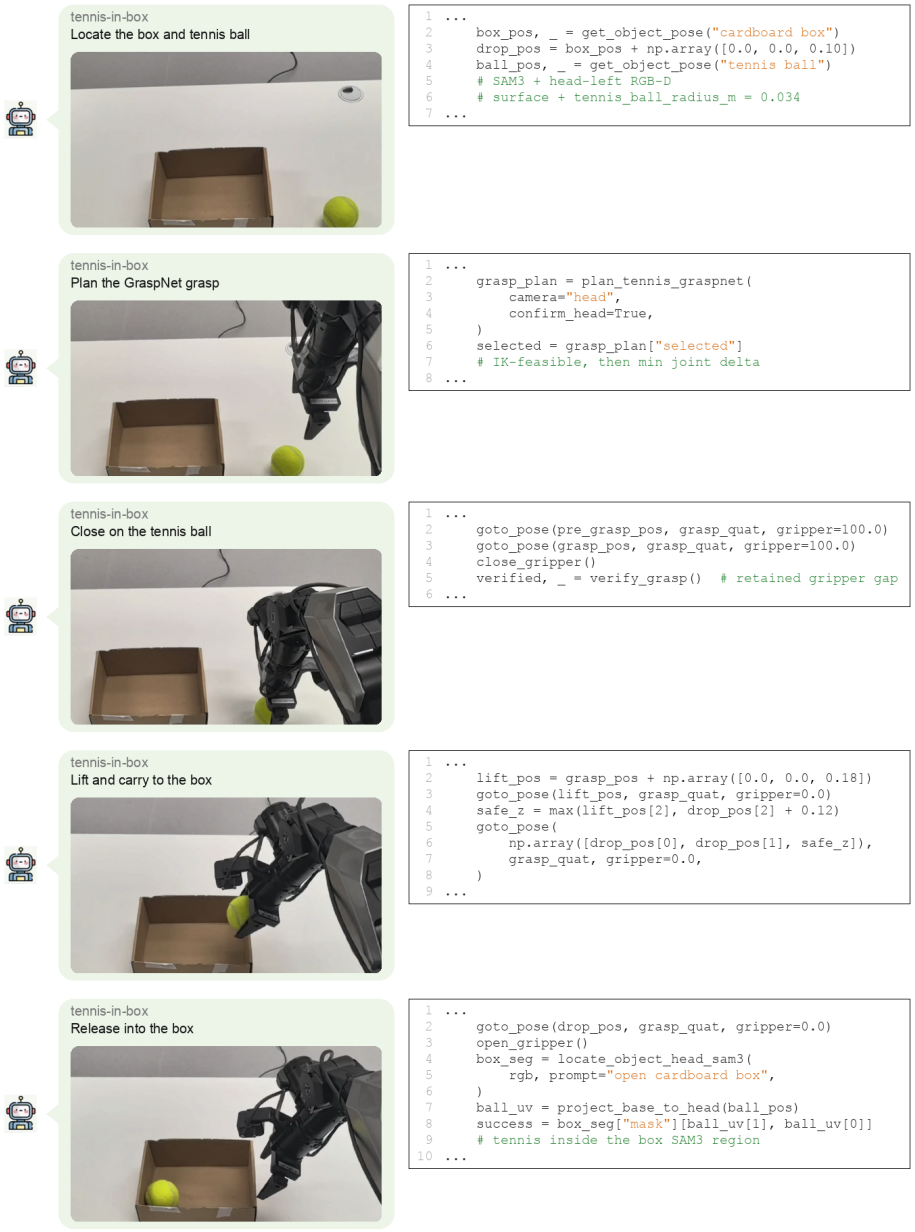


Figure 6: We validate the pick-and-place ability of META-Robot on a real-world Galaxea R1 Lite. A yellow tennis ball and an open cardboard box sit on the table; only the right arm moves. The operator names neither a grasp nor a drop point. We find META-Robot can easily complete this task. We find that META-Robot can pick and place from two live object poses rather than from a saved trajectory or large training data.

META-Robot decomposes the task into locate, plan, pinch, carry, and verify. Head RGB-D and SAM3 recover the box and the ball; the ball center is the stereo surface plus the 3.4 cm radius. The drop pose is 10 cm above the box center. The agent then selects a grabbing position which first passes the IK harness with the smallest joint motion. The arm goes to pre-grasp, then grasp, and closes. The agent accepts the grasp only if the gripper gap is neither empty-closed nor fully open.

The loaded arm lifts 18 cm, crosses at a safe height, and opens at the drop pose. After that, it checks that the ball lies within the box.

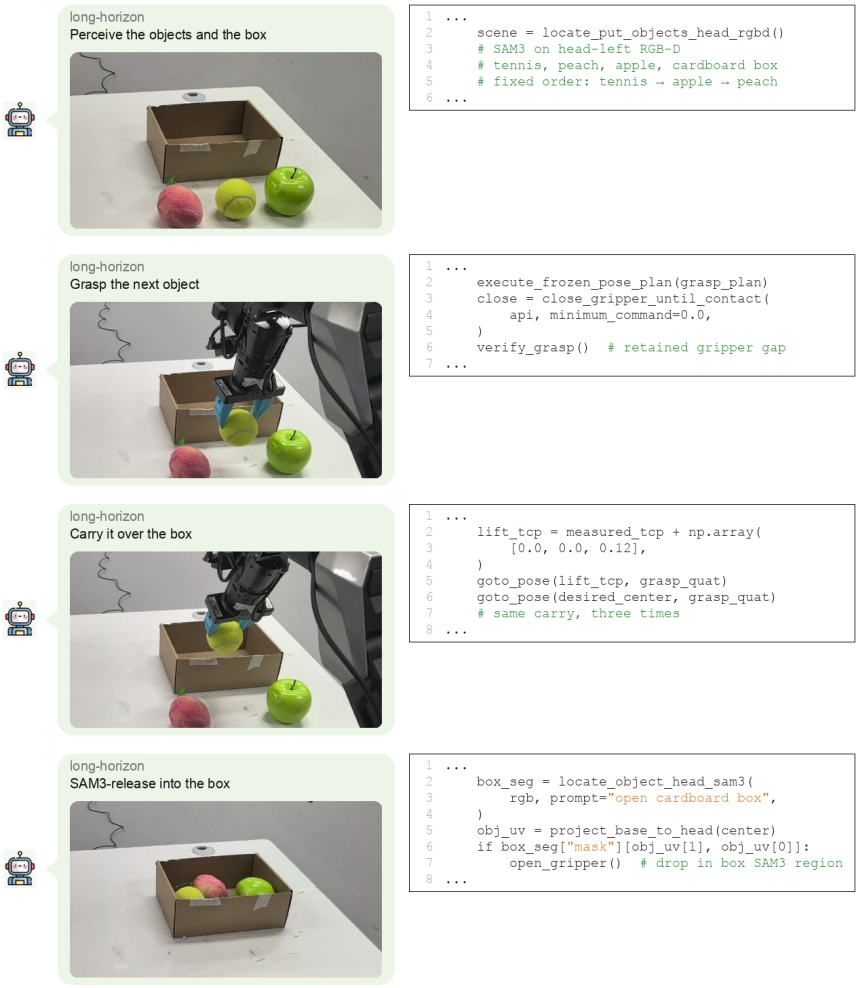


Figure 7: We validate META-Robot's long-horizon manipulation capability on a real-world Galaxea R1 Lite. A peach, a tennis ball, an apple, and an open cardboard box sit on the table; only the right arm moves. This task shows the robot's ability to perform long-horizon pick-and-place tasks.

This demo uses the tennis ball, followed by the apple and then the peach. No object is assigned a fixed spatial slot; only the order is specified. The robot is instructed to pick up the items and place them into the box in this order.

META-Robot decomposes the task into read-order, perceive, grasp, carry, and SAM3-release, repeated three times. One head-left RGB-D frame and SAM3 recover the three objects and the box. Each phase pinches the next object, verifies a retained gripper gap, and carries it over the box. After the third drop, all three objects sit in that same region.

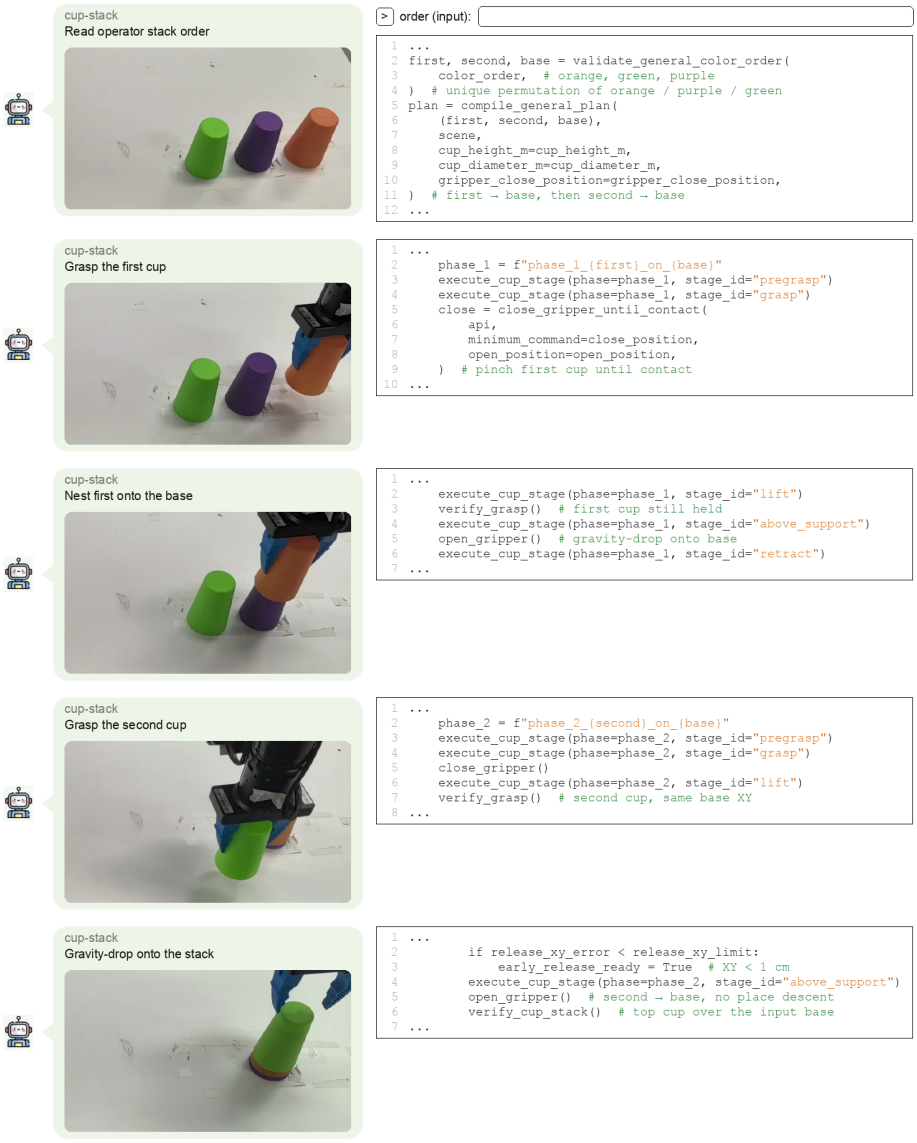
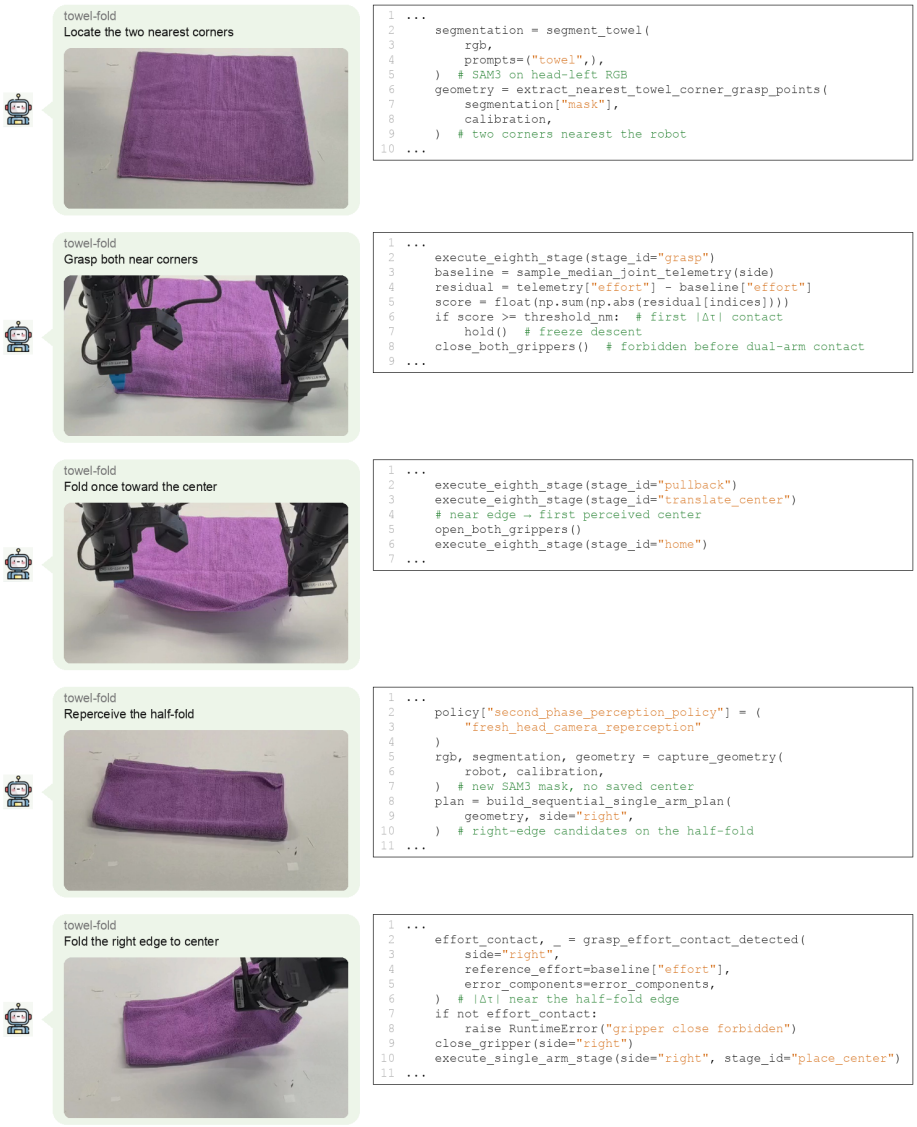


Figure 8: We validate the order-following ability of Meta-Robot on a real-world Galaxea R1 Lite using three inverted plastic cups of the same size—orange, purple, and green. The focus of this task is the robot's ability to follow the specified order, rather than the individual grasping and transfer motions. This real-world task evaluates generalization across different specified color orders.

The cups already sit rim-down, so they nest rather than pile. We specify a unique color order (first, second, base): the first cup is placed onto the base, then the second cup is placed onto the same base. The three cups are initially arranged in a line in a random color order. Each time we input an order of the three colors.

META-Robot decomposes the task into read-order, perceive, grasp, transfer, and gravity-release. Head RGB-D and SAM3, gated by HSV, recover each inverted cup; the upward base is measured from stereo depth, and the downward rim is inferred from the known cup height. After compilation, phase 1 is first-on-base and phase 2 is second-on-base; both releases target the original base X-Y, because nested inverted cups keep the same rim height.




```
...
    segmentation = segment_towel(
        rgb,
        prompts=("towel",),
    )  # SAM3 on head-left RGB
    geometry = extract_nearest_towel_corner_grasp_points(
        segmentation["mask"],
        calibration,
    )  # two corners nearest the robot
...
```

```
...
    execute_eighth_stage(stage_id="grasp")
    baseline = sample_median_joint_telemetry(side)
    residual = telemetry["effort"] - baseline["effort"]
    score = float(np.sum(np.abs(residual[indices])))
    if score >= threshold_nm:  # first |Δτ| contact
        hold()  # freeze descent
    close_both_grippers()  # forbidden before dual-arm contact
...
```

```
...
    execute_eighth_stage(stage_id="pullback")
    execute_eighth_stage(stage_id="translate_center")
    # near edge → first perceived center
    open_both_grippers()
    execute_eighth_stage(stage_id="home")
...
```

```
...
    policy["second_phase_perception_policy"] = (
        "fresh_head_camera_reperception"
    )
    rgb, segmentation, geometry = capture_geometry(
        robot, calibration,
    )  # new SAM3 mask, no saved center
    plan = build_sequential_single_arm_plan(
        geometry, side="right",
    )  # right-edge candidates on the half-fold
...
```

```
...
    effort_contact, _ = grasp_effort_contact_detected(
        side="right",
        reference_effort=baseline["effort"],
        error_components=error_components,
    )  # |Δτ| near the half-fold edge
    if not effort_contact:
        raise RuntimeError("gripper close forbidden")
    close_gripper(side="right")
    execute_single_arm_stage(side="right", stage_id="place_center")
...
```

Figure 9: We validate the ability of META-Robot to fold a soft object on a real-world Galaxea R1 Lite using one flat purple cloth on the table. This task requires the use of proprioceptive signals as the towel is very thin and is quite difficult to grasp using visual feedback alone. META-Robot gives the plan: a bimanual first fold that turns the square into a half-width rectangle, then a right-arm second fold that turns that rectangle into a quarter. Both arms move in the first half; only the right arm moves in the second. The first episode makes the half-fold; the second episode makes the quarter. We validate its ability to handle complex tasks.

META-Robot decomposes the task into perceive-corners, bimanual-grasp, first-fold, reperceive, and second-fold. Head RGB and SAM3 recover the largest towel mask; a fitted rectangle's four corners are projected onto the real external contour, ranked by XY distance to the robot base, and the nearest pair becomes the left and right grasps. After the first fold changes the cloth's shape, the original corner pair is discarded. A fresh head frame re-segments the half-fold; META-Robot now treats it as a new rectangle and samples grasp candidates only on the current right edge, not the original four corners.

The core of this task is to verify its ability to deal with soft, thin objects, which requires the power of perception. One other important thing is to accurately grasp the edge of the towel and ensure there

are no small areas of folding meanwhile. META-Robot uses the signal of joint torques to assist in determining the appropriate contact between the gripper and the table surface.

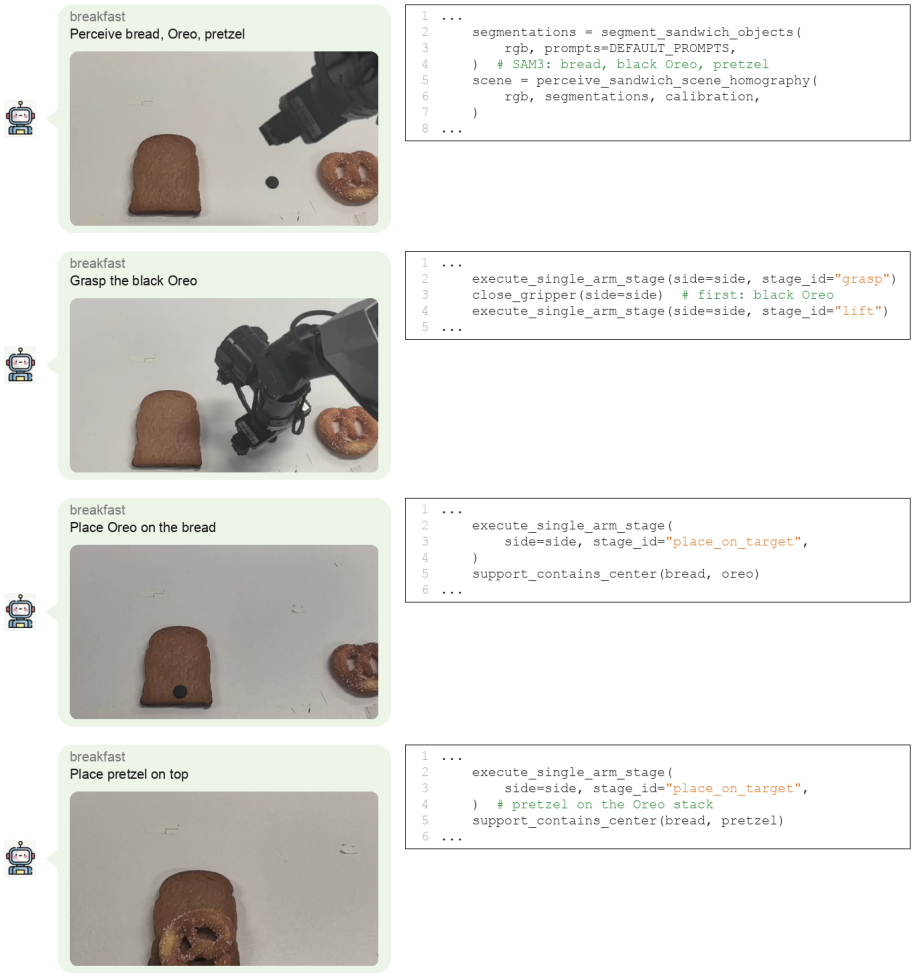


Figure 10: We validate the breakfast task of META-Robot on a real-world Galaxea R1 Lite. A brown toast slice, a small black Oreo, and a pretzel sit on the table; only one arm moves. We use a fixed two-layer order: Oreo onto the bread, then pretzel on top of that stack. This task is also an evaluation of the system's use of proprioceptive feedback.

META-Robot decomposes the task into perceive, grasp, place-on-support, and verify, once per layer. One RGB frame and SAM3 recover the bread, the Oreo, and the pretzel to detect their positions. The first phase pinches the black Oreo and places it onto the toast. After a fresh re-perception, the second phase pinches the pretzel and places it on top of the Oreo.

In fact, the sub-task of grasping a cookie for breakfast shares similarities with the thin-wafer grasping task we simulated; the challenge in both cases lies in determining how to grasp objects whose grasp points are difficult to pinpoint using vision alone. While both tasks rely on detecting contact between the gripper and the table surface, grasping a cookie introduces the additional requirement of controlling the gripper's closing force.

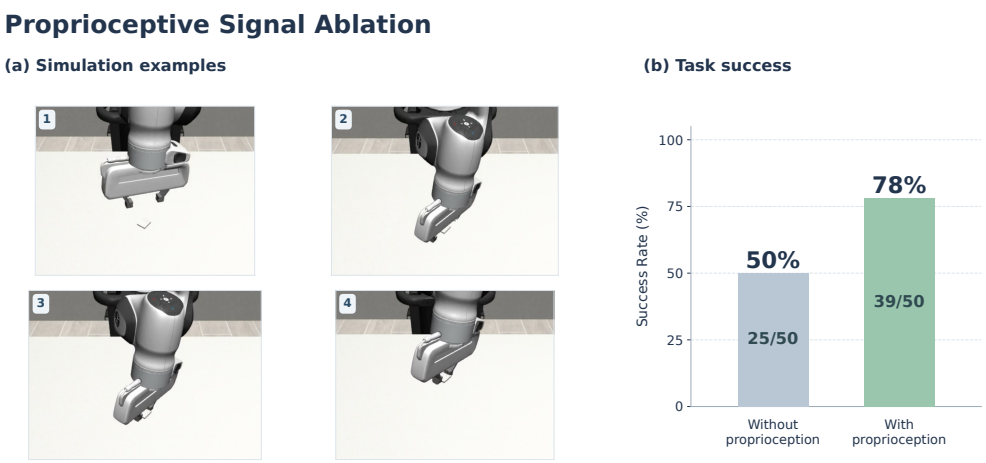
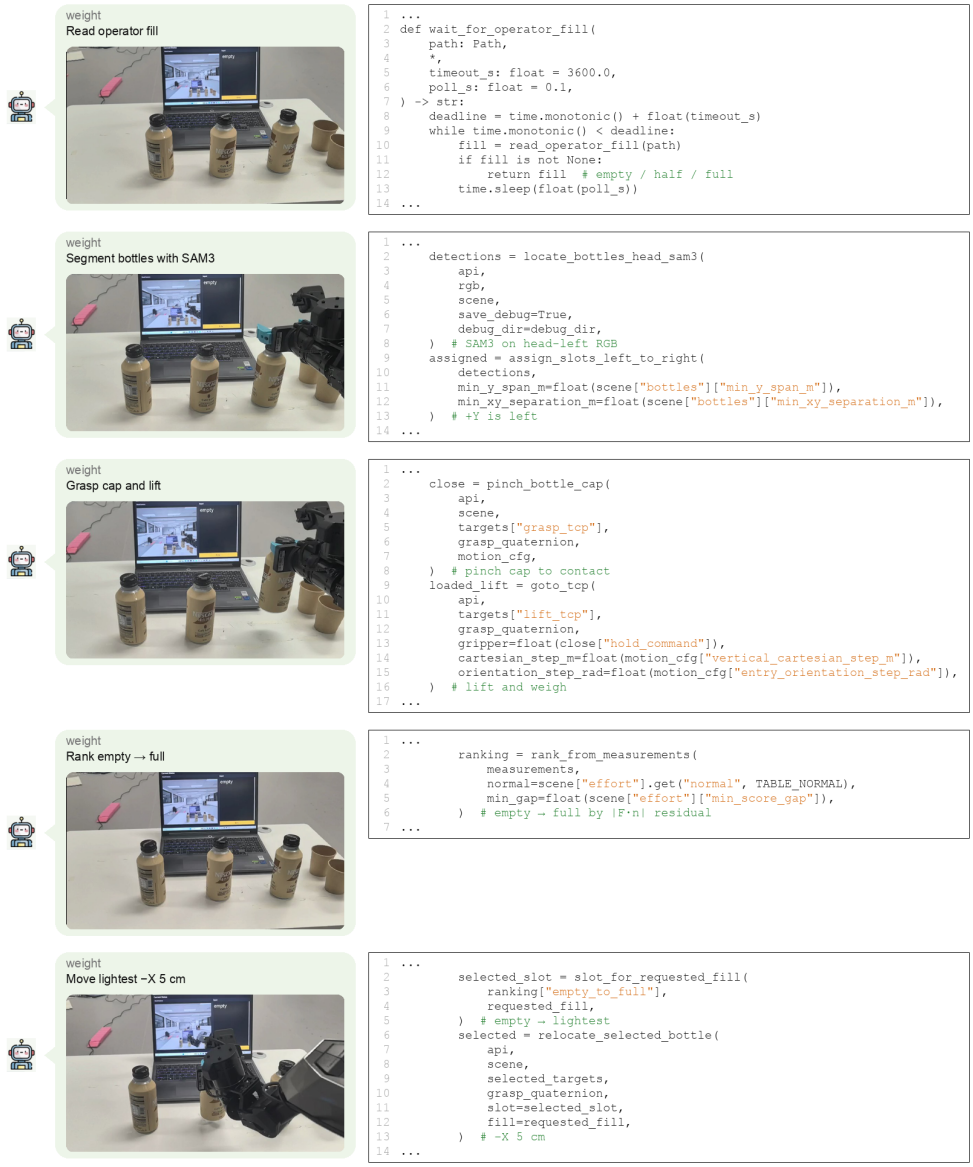


Figure 11: Effect of Proprioceptive Feedback on Manipulation Success: (a) shows the execution process during the simulation, while (b) displays the success rates before and after the ablation of the agent's signals.




```
...
def wait_for_operator_fill(
    path: Path,
    *,
    timeout_s: float = 3600.0,
    poll_s: float = 0.1,
) -> str:
    deadline = time.monotonic() + float(timeout_s)
    while time.monotonic() < deadline:
        fill = read_operator_fill(path)
        if fill is not None:
            return fill  # empty / half / full
        time.sleep(float(poll_s))
...
```

```
...
    detections = locate_bottles_head_sam3(
        api,
        rgb,
        scene,
        save_debug=True,
        debug_dir=debug_dir,
    )  # SAM3 on head-left RGB
    assigned = assign_slots_left_to_right(
        detections,
        min_y_span_m=float(scene["bottles"]["min_y_span_m"]),
        min_xy_separation_m=float(scene["bottles"]["min_xy_separation_m"]),
    )  # +Y is left
...
```

```
...
    close = pinch_bottle_cap(
        api,
        scene,
        targets["grasp_tcp"],
        grasp_quaternion,
        motion_cfg,
    )  # pinch cap to contact
    loaded_lift = goto_tcp(
        api,
        targets["lift_tcp"],
        grasp_quaternion,
        gripper=float(close["hold_command"]),
        cartesian_step_m=float(motion_cfg["vertical_cartesian_step_m"]),
        orientation_step_rad=float(motion_cfg["entry_orientation_step_rad"]),
    )  # lift and weigh
...
```

```
...
        ranking = rank_from_measurements(
            measurements,
            normal=scene["effort"].get("normal", TABLE_NORMAL),
            min_gap=float(scene["effort"]["min_score_gap"]),
        )  # empty → full by |F·n| residual
...
```

```
...
        selected_slot = slot_for_requested_fill(
            ranking["empty_to_full"],
            requested_fill,
        )  # empty → lightest
        selected = relocate_selected_bottle(
            api,
            scene,
            selected_targets,
            grasp_quaternion,
            slot=selected_slot,
            fill=requested_fill,
        )  # -X 5 cm
...
```

Figure 12: We validate the weighing capability of META-Robot on a real-world Galaxea R1 Lite using three visually identical plastic bottles whose fill levels—empty, half, and full—cannot be recovered from RGB appearance. META-Robot uses its proprioceptive signal to realize this task with high accuracy.

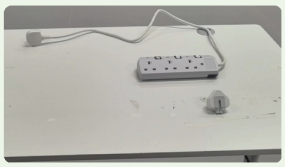
The operator specifies the desired fill in a Current Status window; Enter submits empty, half, or full and starts motion. No bottle is labeled, and the prompt provides no spatial hint for which slot is lightest.

By its proprioceptive signal orchestration mechanism, META-Robot decomposes the task into perceive, weigh, rank, and relocate. SAM3 segments the standing bottles from the head-left camera; slots are assigned left-to-right in the robot base, while the right arm weighs them from right to left.

The key is the signal of joint torques. Each weighing cycle pinches the cap, lifts, and estimates table-normal force from joint effort via $\tau \approx J^{\mathrm{T}} F$. The meta-robot then computes a load score: $|F \cdot n|_{\text{loaded}} - |F \cdot n|_{\text{empty}}$. After three measurements, META-Robot generates three functions: it first sorts the bottles from empty to full, then maps the operator token onto that ranking from empty to the

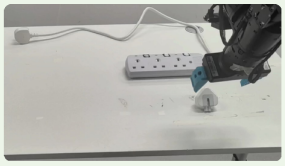
lightest, half to the middle, full to the heaviest, after that it shifts that bottle 5 cm along robot-reverse, which corresponds to the selection order.

plug-insert
Segment plug and strip with SAM3



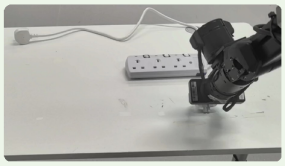
```
...
        socket_seg = api._locate_object_head_sam3(
            rgb,
            prompt=socket_cfg["sam3_prompt"],
            include_mask=True,
        )  # white power strip
        plug_seg = locate_compact_plug_sam3(
            api,
            rgb,
            plug_cfg["sam3_prompt"],
            socket_box=socket_seg.get("box"),
        )  # small three-pin plug
...
```

plug-insert
Grasp the plug



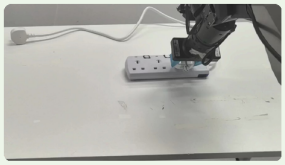
```
...
        close_result = close_gripper_until_contact(
            api,
            minimum_command=0.0,
            open_position=OPEN_GRIPPER_COMMAND,
        )
        if close_result.get("contact_detected"):
            squeeze_command = max(
                0.0, float(close_result["hold_command"]) - 12.0
            )  # extra pinch so the plug cannot skate
...
```

plug-insert
Lift and transfer to the strip



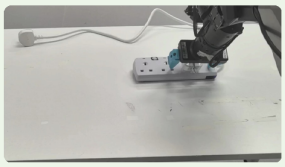
```
...
        lift_position = np.array(
            [float(current_tcp[0]), float(current_tcp[1]), lift_z],
            dtype=np.float64,
        )  # clear the table first
        robot.move_joints_feedforward(
            transfer_joints,
            duration_s=transfer_duration_s,
            context="plug_insert_free_transfer",
        )  # free-orientation stream to the hole
...
```

plug-insert
Aim tool Z along the strip normal



```
...
        socket_center_base, socket_normal_base, socket_inliers =
            estimate_plane_ransac(
            socket_points_base,
            threshold_m=float(socket_cfg["plane_ransac_threshold_m"]),
            min_inliers=int(socket_cfg["plane_min_inliers"]),
        )  # SAM3 strip face
        insert_quat, insert_axis = insertion_quaternion_from_socket_normal(
            table_quaternion, socket_normal_base
        )  # tool Z = -normal, into the face
...
```

plug-insert
Insert into the right-first hole



```
...
        hole_info = estimate_right_first_hole(
            socket_points_base[socket_inliers],
            socket_normal_base,
            inset_m=hole_inset_m,
            hole_z_m=table_z + strip_thickness_m,
        )
        while remaining > plug_pin_standoff_m + 1e-4:
            remaining = max(plug_pin_standoff_m, remaining -
                insertion_step_m)
            target_position = tcp_on_insert_axis(remaining)
...
```

Figure 13: We validate a precision plug-insertion task of META-Robot on a real-world Galaxea R1 Lite. A small three-pin plug and a white power strip are placed at randomized poses on the table. This task shows the use of proprioceptive signal in precise task.

META-Robot decomposes the task into perceive, grasp, transfer, align, and insert. SAM3 first segments the strip and the compact plug. The outward face normal comes from the plane of the detected power strip. The target is the robot-right first hole, inset from the right end of the strip.

After grasping the plug at its angle, the arm lifts, then streams with free orientation to the hole. Near the target hole of the power strip, meta-robot adjusts the angle of the gripper to make sure the plug is perpendicular to the surface normal of the power strip in the opposite direction. The angle alignment is the most difficult part. META-Robot aligns the plug's pin with the hole based on

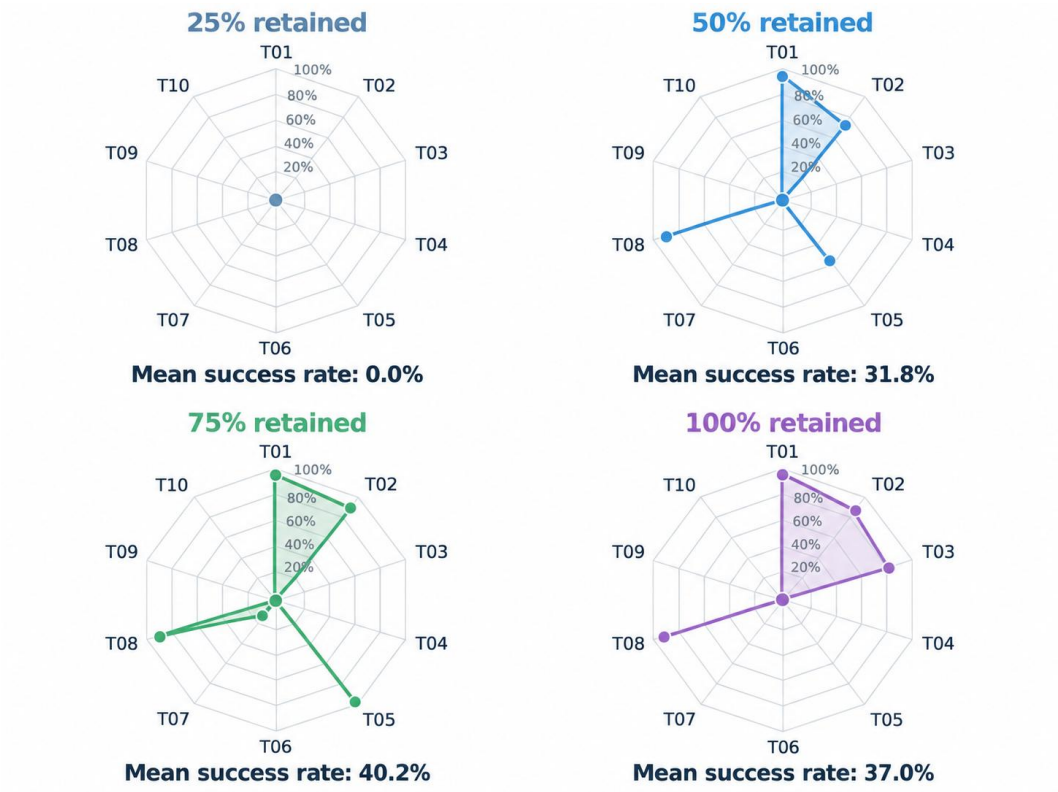


Figure 14: Zero-shot transfer to LIBERO-Pro Long tasks: we evaluate ten representative manipulation tasks under 25%, 50%, 75%, and 100% information retention. The tasks include T01 soup-and-sauce placement, T02 cheese-and-butter placement, T03 stove/moka-pot interaction, T04 bowl-in-drawer placement, T05 mug-to-plate placement, T06 book-to-caddy placement, T07 mug-and-pudding arrangement, T08 soup-and-cheese placement, T09 moka-pot placement, and T10 mug-in-microwave placement. Each radar plot reports per-task success rates, with the mean success rate shown below.

the initially identified position by SAM3. The insertion then advances along that axis until the pin standoff, opens the gripper, and returns home.

A.2 SUBTASK ANALYSIS OF ZERO-SHOT TRANSFER

A.3 API CHOICE

GPT-6 Astra reaches 45%, above the highest historical point shown in the figure, Gemini 3 Pro at about 32%. DeepSeek-V4-Flash reaches 31%, close to that earlier result. The added evaluations therefore show a clear gain for GPT-6 Astra, while DeepSeek-V4-Flash performs at a similar level to the strongest model highlighted in the original plot.

A.4 SKILL LIBRARY & PREDICATE LIBRARY ALGORITHM

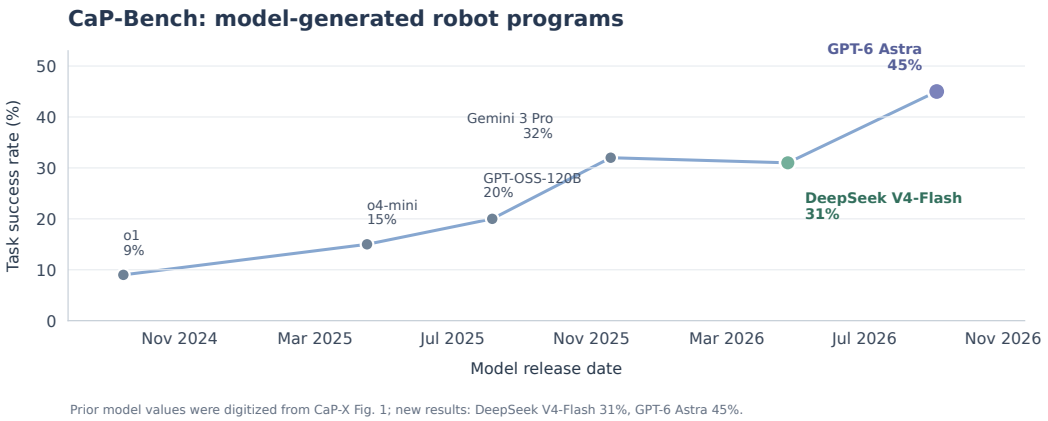


Figure 15: We extend the base model comparison shown in CaP-X Fu et al. (2026)with DeepSeek-V4-Flash and GPT-6 Astra, reporting task success rates of 31% and 45%, respectively.

Algorithm 1 Coupled Skill and Action-Level Predicate Library Consolidation

Require: Skill forest $\mathcal{L}_S$, action-level predicate library $\mathcal{P}$, associations Γ, candidate repair ρ, and execution evidence $\mathcal{D}$
Ensure: Updated $\mathcal{L}_S$, $\mathcal{P}$, and Γ

if $\neg\,\mathrm{Validated}(\rho, \mathcal{D})$ **then**
  return $(\mathcal{L}_S, \mathcal{P}, \Gamma)$
end if
$(\hat{s}, \widehat{\mathcal{P}}) \leftarrow \mathrm{Distill}(\rho, \mathcal{D})$
$\mathcal{V}_{\mathrm{rel}} \leftarrow \mathrm{RetrieveRelated}(\hat{s}, \mathcal{L}_S)$
$(\mathcal{L}'_S, v) \leftarrow \mathrm{UpdateSkillTree}(\hat{s}, \mathcal{V}_{\mathrm{rel}}, \mathcal{L}_S)$
if $\mathrm{NeedsValidation}(\mathcal{L}'_S, v) \wedge \neg\,\mathrm{ValidateSkillUpdate}(\mathcal{L}'_S, v, \mathcal{D})$ **then**
  return $(\mathcal{L}_S, \mathcal{P}, \Gamma)$
end if
$\mathcal{P}' \leftarrow \mathcal{P}$
$\Gamma' \leftarrow \Gamma$
for $\hat{\phi} \in \widehat{\mathcal{P}}$ **do**
  $\phi \leftarrow \mathrm{FindEquivalentPredicate}(\hat{\phi}, \mathcal{P}')$
  if $\phi = \emptyset$ **then**
    if $\neg\,\mathrm{ValidatePredicate}(\hat{\phi}, \mathcal{D})$ **then**
      continue
    end if
    $\phi \leftarrow \hat{\phi}$
    $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{\phi\}$
  end if
  $\Gamma'(v) \leftarrow \Gamma'(v) \cup \left\{\left(\phi, \mathrm{EvalMeta}(\hat{\phi}, \mathcal{D})\right)\right\}$
end for
$(\mathcal{P}', \Gamma') \leftarrow \mathrm{ReindexPredicates}(\mathcal{P}', \Gamma', \mathcal{L}'_S)$
return $(\mathcal{L}'_S, \mathcal{P}', \Gamma')$

A.5 Skill and Action Level Predicate details analysis

Localization The localization skill converts open-vocabulary RGB-D observations into task-bound entities before physical execution. Candidate masks are obtained using task-relevant prompt variants, filtered according to geometric consistency, and projected into the robot's world frame. The action predicates visible, localized, and role bound represent progressively stronger claims: detecting an object does not imply that its metric pose is known, and estimating its pose does not yet establish its role in the current task. Relational predicates such as inside and at target pose are eval-

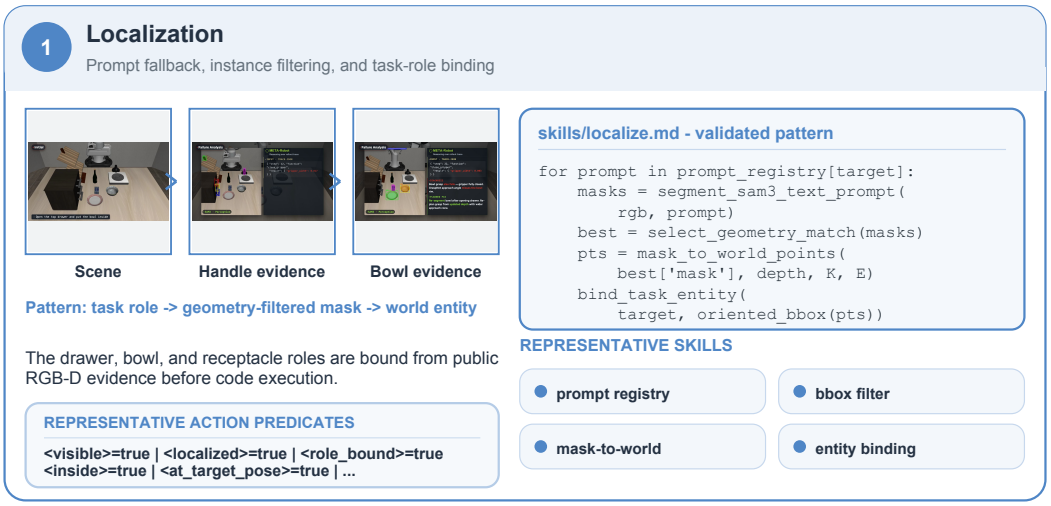


Figure 16: Localization and task-role binding: Open-vocabulary RGB-D masks are geometrically filtered and projected into the robot frame, binding language-specified roles to metric entities and exposing validated localization predicates before downstream execution.

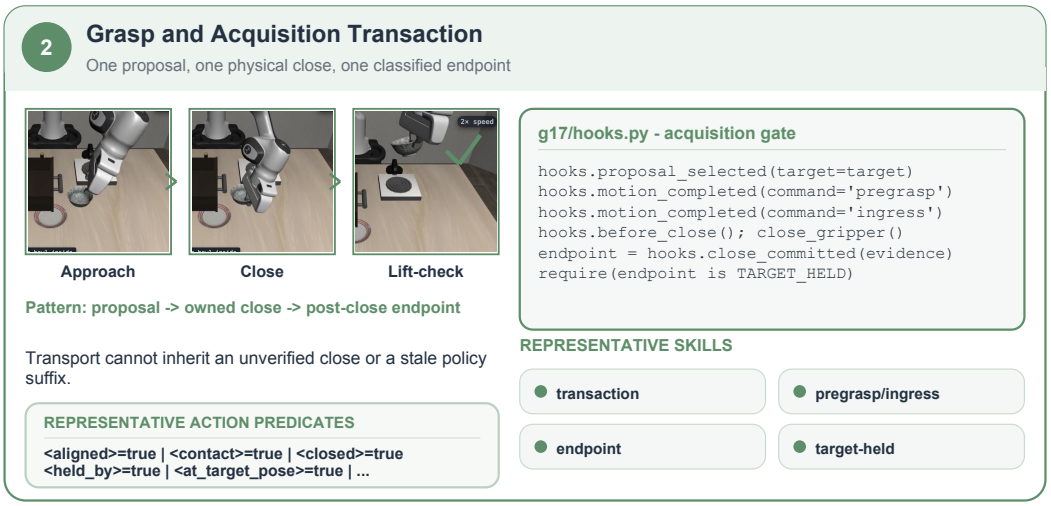


Figure 17: Grasp_and_Acquisition: A single committed gripper closure is verified by post-close visual and proprioceptive evidence. Transport is admitted only when this evidence establishes TARGET-HELD; contact or closure alone is insufficient.

uated only after this binding step, preventing a semantically plausible but geometrically incorrect detection from propagating to downstream skills.

Grasp_and_Acquisition Grasp execution is represented as a transaction consisting of approach, one committed physical close, and a post-close endpoint classification. The predicates aligned, contact, and closed describe intermediate physical conditions, whereas held by denotes verified acquisition by the gripper. These predicates remain separate because finger closure or transient contact alone does not establish a successful grasp. Transport is admitted only when the endpoint is classified as TARGET_HELD, using post-close visual and proprioceptive evidence. This transaction boundary prevents downstream skills from inheriting an unverified grasp or reusing stale observations collected before the closing action.

Proprioception_and_Dynamic_Action Predicates Dynamic action predicates are evaluated from event-aligned sensor evidence rather than from isolated observations. For each relevant action, the

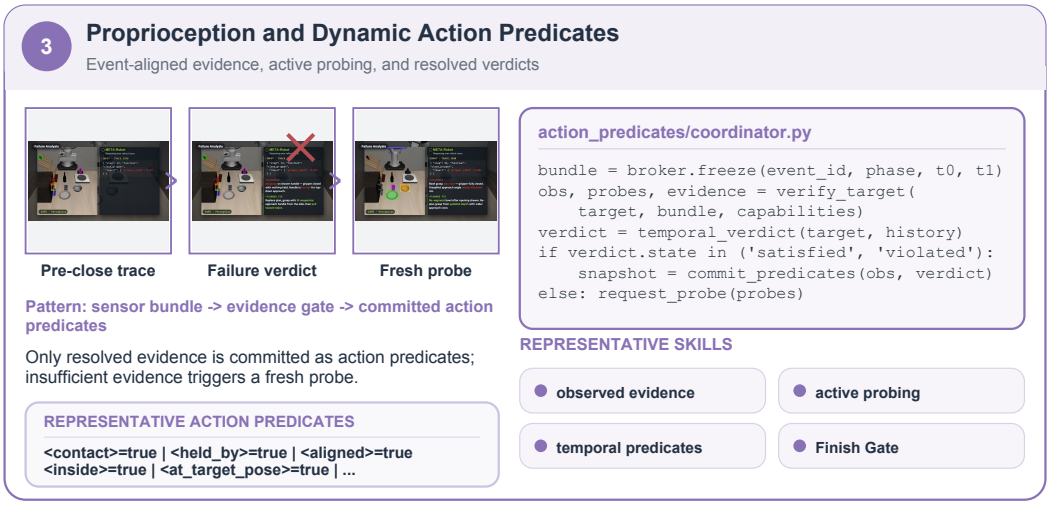


Figure 18: Proprioception_and_Dynamic_Action_Predicates: Event-aligned vision, gripper state, and joint signals resolve contact, holding, alignment, and placement predicates at action boundaries; insufficient evidence triggers an active probe.

system freezes a bounded evidence window containing visual observations, gripper state, joint measurements, and available torque signals. Temporal verification then resolves predicates such as contact, held by, aligned, inside, and at target pose. Only predicates supported by sufficient evidence are committed to the execution state. When the available evidence is insufficient, the system requests an active probe or a fresh observation instead of assigning an uncertain result. This mechanism allows action predicates to remain physically interpretable while supporting closed-loop verification under partial observability.

Transport_with_Recovery The transport skill inherits the verified acquisition state and admits motion only when held by remains satisfied. The inherited predicate avoids duplicating grasp verification while preserving a strict dependency between acquisition and transport. During transit, predicates such as aligned, at target pose, contact, and inside are updated at explicit observation boundaries. If the acquisition endpoint does not support transport, the controller selects a bounded recovery route, such as inverse ingress followed by fresh observation, rather than continuing optimistically. Consequently, each transport attempt either proceeds from a verified object-held state or terminates through a finite recovery transition.

Manipulation_and_ReleaseReady Manipulation and release are separated by an explicit ReleaseReady gate. In the illustrated drawer task, open describes the articulated state of the receptacle, while held by, inside, on, and contact characterize the evolving relation between the object, gripper, and destination. Importantly, the placement frame denotes a pre-release configuration: the target remains held while its over-receptacle geometry and intended placement relation are evaluated. The gripper is opened only after the required action predicates jointly satisfy ReleaseReady. This fail-closed design prevents premature release and distinguishes geometric placement readiness from the irreversible act of releasing the object.

Long_Horizon_Closure Long-horizon execution is evaluated against a revisioned action-predicate specification containing milestone, terminal, and invariant conditions. Task-dependent predicates such as on, inside, inserted, stacked on, and activated provide a common interface for evaluating heterogeneous manipulation outcomes. The Finish Gate admits completion only when the latest evidence satisfies all required predicates and execution milestones. Successful traces are then packaged with their action-predicate states, provenance, and evidence hashes before being proposed for reuse. A proposal cannot directly modify the canonical skill library; promotion requires an additional review gate, separating task completion from the stronger claim that an execution pattern is reliable and reusable.

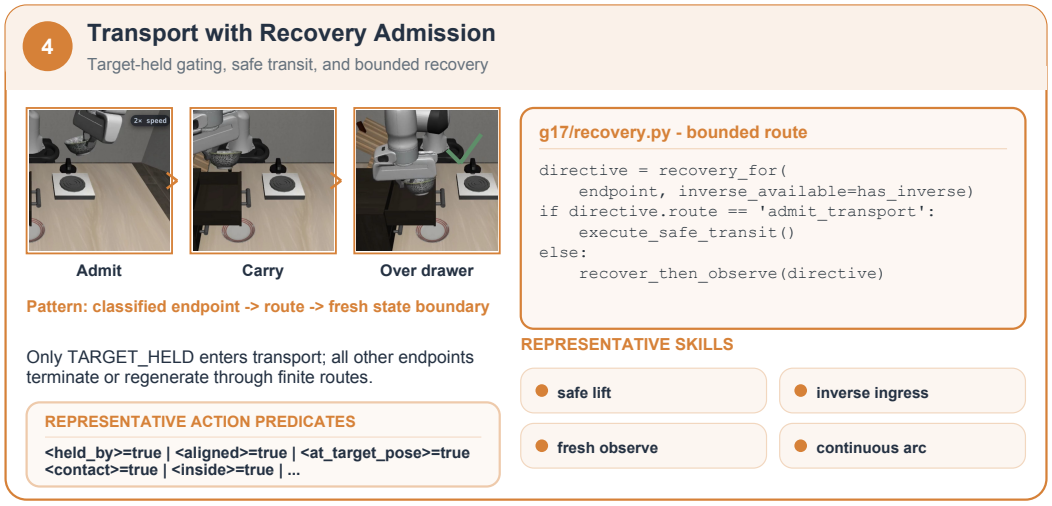


Figure 19: Transport_with_Recovery: Only a verified TARGET-HELD endpoint enters safe transit. Invalid endpoints invoke bounded inverse-ingress or fresh-observation routes, preventing stale acquisition evidence from propagating into subsequent manipulation.

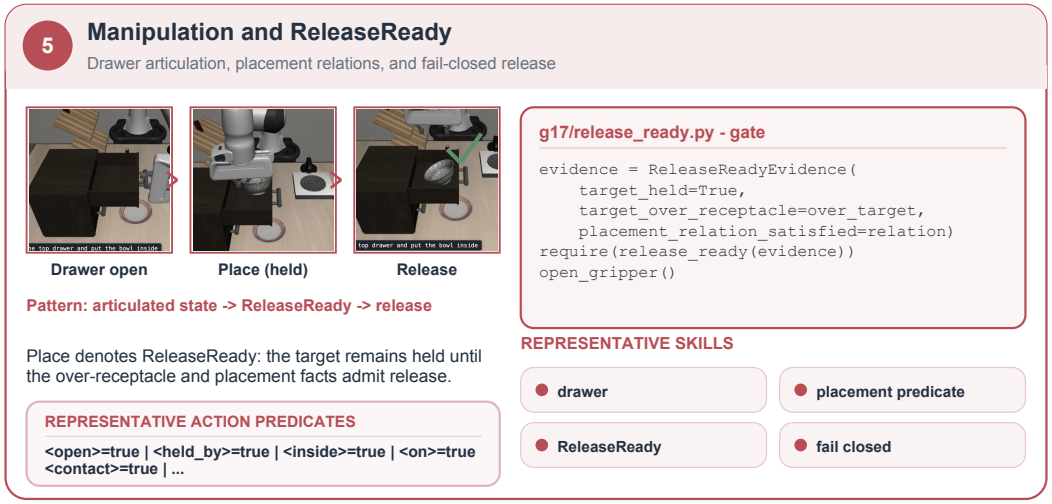


Figure 20: Manipulation_and_ReleaseReady: Drawer state, over-receptacle geometry, and placement relations are evaluated while the object remains held. Release occurs only when the required action predicates jointly satisfy ReleaseReady.

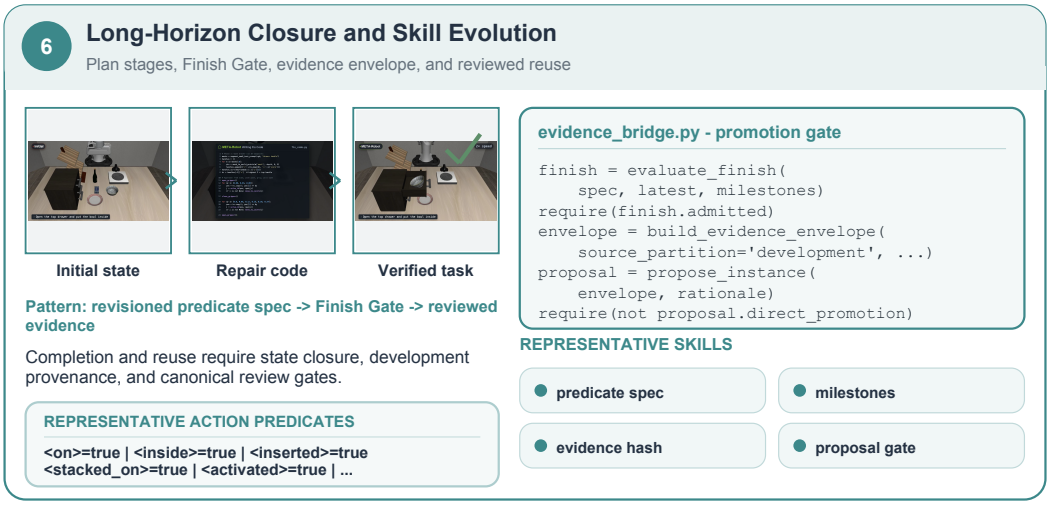


Figure 21: Long_Horizon_Closure: The Finish Gate checks milestone, terminal, and invariant predicates. Successful traces are packaged with code provenance and evidence hashes, then reviewed before updating the reusable skill and predicate libraries.