AGENCY AT THE RIGHT SCALE: BIMANUAL MANIPULATION THROUGH ACTIVE OBSERVATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Robot agents must adapt their reasoning as they act on and acquire knowledge about the physical world. In robotic manipulation, this raises a central question: which decisions should remain open to the model as execution unfolds? We introduce *Agency at the Right Scale* (ARS), a bimanual robot-agent architecture that makes the scope of reasoning an online decision. ARS connects task-level reasoning with sustained local interaction, preserving established goals and arm roles during local correction while allowing physical feedback to trigger broader replanning. Active perception makes this coordination embodied: the agent moves a wrist camera to resolve uncertainty in an ongoing manipulation, then uses the acquired evidence to decide whether to continue correcting, execute an established operation, or revise the task arrangement. Reusable procedures retain established action structure across executions, leaving scene-dependent interpretation and recovery to online reasoning. We evaluate ARS on 18 RoboDojo tasks and use execution studies and physical-robot demonstrations to examine how observation, correction, and procedural reuse interact during manipulation. ARS connects what the robot has established through physical interaction to what it can commit to execution and what must remain open to model judgment.

1 INTRODUCTION

Robotic manipulation poses a fundamental challenge for reasoning agents: physical interaction changes both the world they must act on and the information available for deciding what to do next. A useful plan must therefore support progress while remaining open to revision through interaction. Frontier vision language models offer a foundation for such agency, bringing reasoning to robot control through programs, tools, and visual feedback (Liang et al., 2023; Fu et al., 2026; Chen et al., 2026a;d; Hu et al., 2026). These capabilities allow agents to intervene throughout execution, raising a question about the scope of that intervention: which decisions should remain open to model reasoning as physical interaction changes what is known and what remains unresolved?

Effective manipulation requires an agent to move between coarse progress and fine physical adjustment without losing the task it is trying to accomplish. Transporting a held tube toward a rack can be specified as a single operation, whereas aligning its tip with an occluded opening requires repeated observation and correction. A local correction should preserve the chosen target and grasp, yet a newly revealed obstruction may require a different approach or arm arrangement. The appropriate granularity therefore depends on what the current evidence resolves, rather than on a fixed division between planning and control.

The evidence needed to organize this reasoning is itself a product of physical interaction. A reasoning scope determines what an observation must resolve, while a new observation can change which decisions require further attention. In bimanual manipulation, this relationship is tangible: moving one arm's wrist camera can reveal geometry hidden by the other arm or a held object. During tube insertion, for example, a new view may support an alignment correction or reveal an obstruction that requires revising the approach. Obtaining such a view requires moving the robot, subject to reachability and workspace constraints, and incurs additional motion cost. It can support continued interaction within a scope or establish a reason to reconsider its surrounding commitments. We observe that GPT-6, acting through a direct tool interface, can spontaneously reposition an unoccupied arm's wrist camera to seek a clearer view of an ongoing manipulation. Such behavior further

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.

motivates the harness that supports model-selected observation strategies and connects the acquired evidence to continued interaction and task-level reconsideration.

We introduce *Agency at the Right Scale* (ARS), a bimanual robot-agent architecture that adaptively combines coarse task execution and fine visual interaction according to current observations and execution feedback. Physical observation informs both the next action and which established decisions need reconsideration. We define a *decision scale* by the commitments open to model revision: local correction preserves the assigned goal and arm roles, while task-level reconsideration can change the arrangement that defines the interaction. A *Task-Level Agent* can reconsider targets, approaches, subgoal order, and arm roles. It organizes execution through perception and manipulation APIs, programs, and reusable skills, and can delegate a bounded goal to a *Local Interaction Agent*. Local sustains observation and action under that goal and its assigned arm roles, returning when the interaction is complete or broader intervention is needed. Both agents acquire physical evidence and interpret execution outcomes to decide whether to continue reasoning within the current scope, transfer responsibility, or execute a specified operation. Such operations can range from individual corrections to multi-step procedures; their execution span is distinct from the scope of decisions open to revision. Shared memory and execution reports preserve context across transfers.

Across executions, *offline skill consolidation* changes which decisions require online reasoning. When a recurring interaction can be expressed using fresh perceptual inputs and explicit control logic, its intermediate choices can be incorporated into a reusable skill. The Task-Level Agent can then invoke that skill while retaining responsibility for judging its applicability and assessing its outcome. A procedure may still include fine-grained sensing and correction, while choices previously made through repeated model reasoning are handled by its control logic. Skill consolidation thus extends ARS across experience: reusable procedures handle established parts of an interaction, while scene-dependent decisions remain open to online reasoning. Figure 1 summarizes the architecture.

This perspective also has implications for policy learning: successful execution alone does not establish that the recorded observations contain enough information to reproduce the demonstrated actions. An expert may act on visual access or prior knowledge unavailable to the learner. An embodied agent's attempts to acquire missing views can therefore inform what evidence a demonstration collection routine should record. We examine this issue through paired tube-insertion demonstrations from a scripted expert, holding initial layouts and manipulation logic fixed while varying physical view acquisition. This study extends our focus from evidence acquired during execution to evidence available in demonstrations.

Our contributions are:

- **Active observation as part of bimanual manipulation.** ARS integrates physical view acquisition with manipulation, allowing one arm to maintain the ongoing interaction while the other acquires visual evidence for subsequent corrections.
- **Adaptive coordination of coarse execution and fine interaction.** ARS uses current physical evidence to choose between executing a specified operation, sustaining local observation and correction, and reconsidering the surrounding task arrangement. Explicit local goals preserve established commitments during fine interaction; the return path allows new evidence to reopen task-level decisions.
- **Skill consolidation across decision scales.** ARS incorporates recurring manipulation procedures into parameterized skills for task-level reuse. Execution cases illustrate how these skills replace intermediate model decisions while retaining fresh perceptual inputs, and explicit cost accounting characterizes the resulting reduction in model invocations.

2 Related Work

Robot agents through programs and grounded interfaces. Language-model robot agents connect semantic reasoning to perception and control through executable interfaces. CaP generates programs that compose robot primitives (Liang et al., 2023), while Inner Monologue incorporates execution feedback into language-based planning (Huang et al., 2022). VoxPoser and ReKep ground model outputs in spatial value maps and relational keypoint constraints, respectively, supporting closed-loop motion generation (Huang et al., 2023; 2024). CaP-X systematically studies coding agents across control abstractions and interaction settings, including feedback and skill synthesis (Fu et al.,

2026). VIA provides a visual control interface (Hu et al., 2026), and ETA alternates a tool invocation with returned evidence and a fresh observation (Chen et al., 2026d). Together, these works establish programs, spatial representations, and interactive tools as interfaces for robot agency. ARS builds on these interfaces to connect task-level execution with sustained local observation–action reasoning, allowing current physical evidence to inform which decisions remain open during execution.

Hierarchical control and adaptive reasoning. Hierarchical systems separate task selection from motor execution. SayCan combines language-model relevance with learned skill affordances (Ahn et al., 2022), and Hi Robot uses high-level reasoning to interpret instructions and feedback before invoking low-level policies (Shi et al., 2025). Steerable Policies expose commands at several abstraction levels (Chen et al., 2026c); OneTwoVLA adaptively switches between reasoning and action generation (Lin et al., 2026). HarnessVLA places semantic re-grounding and re-staging in an agent that combines analytic primitives with a frozen VLA for contact-rich execution (Zhang et al., 2026b). These approaches address related choices about action abstraction, reasoning frequency, and responsibility for recovery. ARS adaptively combines coarse task execution and fine interaction through explicit delegation and return: Task selects direct execution, a reusable procedure, or a local goal; Local acquires evidence and corrects withinthat goal, returning when the task arrangement needs reconsideration. This couples execution granularity to the scope of subsequent decisions.

Embodied active perception and relative control. Active perception makes sensing part of control. ActiveGrasp selects views while deciding whether to continue observing or grasp (Breyer et al., 2022), and Observe Then Act learns asynchronous viewpoint and manipulation policies (Wang et al., 2025). Bimanual active-perception systems use one arm to observe the other's interaction (He et al., 2026; Wang et al., 2026b); ObAct uses reconstructed views to select a pose that the observer arm physically reaches. See2Act couples action denoising with camera-pose refinement and demonstrates physical wrist-camera acquisition (Wang et al., 2026a). ActiveVLA instead selects virtual views and zooms within available 3D evidence (Liu et al., 2026). Visual servoing provides the established foundation for using visual errors to guide local motion (Chaumette & Hutchinson, 2006). ARS exposes camera-relative translations and rotations from a selected head or wrist view, with the Runtime converting these commands into bounded robot motion. Its focus is the agent's joint choice of physical view acquisition and local correction, together with whether the acquired evidence supports continued interaction, grounded execution, or a return to task reasoning.

Program structure and reusable procedural experience. Executable skill libraries allow agents to retain procedures across tasks. Voyager develops and retrieves code skills through feedback-driven exploration in Minecraft (Wang et al., 2023). In robotics, GaP constructs execution graphs from modular skills and refines them through simulation rehearsal (Chen et al., 2026a). RATs distills self-directed robot play into a skill library that is frozen for downstream evaluation (Zhang et al., 2026a), while ASPIRE develops reusable procedures through program repair and execution-based search (Lu et al., 2026). ARS uses offline consolidation to replace recurring intermediate model decisions with parameterized procedures. During an episode, a procedure can use fresh perception and feedback control without intermediate agent deliberation. It returns to Task on completion or exception; Task then assesses the outcome and decides whether to reuse a procedure or delegate unresolved interaction to Local. Reuse therefore changes how much established behavior can execute before model reasoning is needed again.

3 METHODOLOGY

3.1 OVERVIEW

ARS organizes reasoning around the choices that remain unresolved during execution. Its guiding principle is to preserve established decisions while allowing the model to reconsider them in light of physical feedback. ARS connects a Task-Level Agent and a Local Interaction Agent through a shared Robot Runtime. We define the framework as

$$\mathcal{F} = (\pi_{\text{Task-level}}, \pi_{\text{Local}}, \mathcal{R}, \mathcal{K}), \tag{1}$$

where $\pi_{\text{Task-level}}$ and π_{Local} are the two agent policies. The Robot Runtime $\mathcal{R}$ provides perception, motion execution, and outcome reports through agent-specific tool interfaces, and the skill library $\mathcal{K}$ supplies reusable procedures. The library is fixed within an episode and can be extended by offline skill construction between episodes.

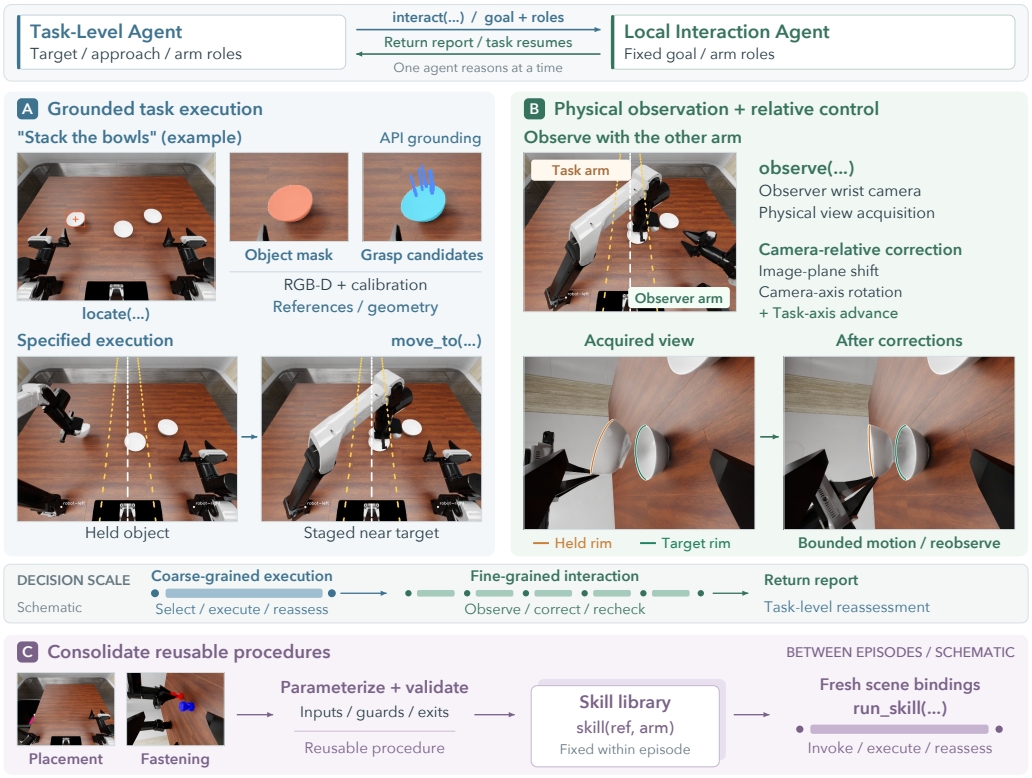

Figure 1: ARS couples two decision spaces through a shared robot Runtime.

Both agents follow a ReAct loop (Yao et al., 2023), with only one agent deliberating at a time. Let $z \in \{\text{Task-level}, \text{Local}\}$ denote the active agent. Its context h_t^z contains the task instruction, current head and wrist images and robot measurements, shared task notes and geometric references, and its own recent interaction history. The interaction contract c_t constrains Local to a specified goal, arm roles, and action limits; it is empty at the task level.

While agent z retains control, it selects a tool call and arguments a_t, receives Runtime feedback o_{t+1}, and updates its context:

$$a_t = \pi_z(h_t^z; c_t), \quad o_{t+1} = \mathcal{R}(a_t), \quad h_{t+1}^z = \text{Update}(h_t^z, a_t, o_{t+1}). \tag{2}$$

Here $\mathcal{R}(a_t)$ abbreviates execution in the current environment and the resulting feedback, including execution outcomes and any observations acquired; t indexes model decisions rather than low-level control steps.

Coordinating these loops requires deciding which choices still need model judgment: whether to revise task-level commitments, resolve physical details within a fixed local goal, or execute an already specified operation. Section 3.2 defines these decision scopes and explains how the agents use current evidence to continue reasoning, transfer control, or commit an operation to the Runtime. Section 3.3 describes how physical observation acquires new evidence, and Section 3.4 explains how offline skill construction produces reusable procedures for subsequent episodes.

3.2 Decision Spaces and Online Selection

The two decision spaces specify which choices each agent may revise. The Task-Level Agent can change targets, subgoal order, grasp arrangements, coarse approaches, and arm roles. Local retains the assigned goal and roles while choosing the arm to move, the reference view, and the permitted observation or motion correction. An arm may both observe and manipulate within its assigned role. Appendix A, Table 2, lists the tools available to each agent and their access restrictions.

The Task-Level Agent transfers control to the Local Interaction Agent through `interact`, a delegation call carrying the contract c_t: the goal, arm roles, permitted actions, limits, and completion condition. Task pauses while Local runs its ReAct loop under this contract. Local uses `return_to_task` to report completion or an unresolved problem, including a need to revise task-level commitments. Budget or round limits can also terminate the local session without implying completion. Task then resumes its saved context with the local report, current observations, and robot measurements. The contract defines which decisions remain fixed during local reasoning. Evidence of an alignment error can guide further correction within that scope; evidence that progress requires a different goal or arm arrangement can motivate a return to Task. Task interprets the returned evidence and decides which commitments to revise before executing or delegating again. These choices are made by the agents through the existing tool interface.

Online selection occurs through the active model's choice of the next tool and its arguments from the context defined in Section 3.1. Task can use individual APIs, invoke a skill, or delegate; Local can continue with a permitted action or return. Ordinary API calls provide feedback to the calling agent, while delegation and return change who reasons next. Skill invocation retains task-level authority but allows a specified procedure, including perception and feedback control, to execute until its endpoint or an exception without intermediate model decisions. Thus, tool choice determines both control transfer and how much execution occurs before the next model decision. Decision scope and execution length are distinct: Local may sustain many model decisions under one contract, while a task-level skill call may execute many actions before returning feedback. Skill construction and reuse are described in Section 3.4.

3.3 Active Perception Across Decision Scales

A decision scope identifies what must be resolved while physical observation changes the evidence available to resolve it. ARS therefore places view acquisition within the agent's action choices. From current observation and valid memory, the model judges whether to act on the available evidence or seek a view that could clarify the next decision. Such uncertainty can arise from occlusion, insufficient visual detail, or a viewing direction that leaves position or orientation ambiguous. Observation moves a real wrist camera, so the requested view must be compatible with the current grasp, reachable arm configurations, and remaining motion budget.

The Task-Level Agent observes to inform subsequent manipulation decisions. It inspects objects or target regions to determine their spatial properties, choose an approach, and assign manipulation and observation roles. RGB-D and camera calibration lift observed features into base-frame references that remain in task context as memory. Later actions can recall these references across intervening manipulation; stale or inconsistent references require renewed perception. The Task-Level Agent also positions an observer to provide an informative initial view for local alignment. When the acquired evidence sufficiently specifies an operation, the Task-Level Agent can execute it directly; otherwise, it can delegate the remaining alignment problem to Local Interaction.

The Local Interaction Agent uses observation to resolve alignment within the established goal and arm roles. It selects the head or wrist view that best reveals the relevant spatial relation and, when necessary, repositions a permitted arm to obtain a more informative view while preserving the grasp. The arm supplying the reference view need not be the arm being controlled: an observer's wrist image can guide a correction by the holding arm, while a receiver's wrist can guide its own approach during takeover. For camera-relative corrections, the selected view defines the directions of translation and the axes of rotation. The agent therefore chooses both where to look and how to express the correction; the Runtime maps the request into bounded robot motion using depth and calibration. Subsequent views reveal the effect of the action and guide further correction or advancement along the specified contact axis. If the evidence instead indicates that progress requires a change outside the contract, Local can return the unresolved problem to Task. Observation thus supports both continued interaction within a scope and reconsideration of the commitments that define it. Section 4.2 and Appendix C illustrate how these observation mechanisms support subsequent manipulation and local alignment across tasks.

3.4 Recursive Self-Improvement through Skill Consolidation

Reuse extends the same principle across episodes: established operation structure can be carried forward, while decisions that depend on the new situation remain open. Offline skill construction uses a coding agent to turn execution experience into parameterized procedures over the available APIs. This expands the skill library between episodes, closing the RSI loop without changing the online model weights. The builder analyzes traces from both decision scales. Task-level traces supply perception and manipulation procedures; local traces supply bounded action sequences within an established interaction. Each trace records API calls, arguments, decision rationales, observation references, and execution outcomes. Calls and outcomes establish what was executed; rationales help identify the purpose and dependencies of candidate sequences. The builder identifies reusable structure within each scope, treating delegation and return as segment boundaries. Repetition can suggest a candidate, but the criterion for consolidation is whether its intermediate choices can be expressed through parameters and explicit program logic. A step requiring fresh model interpretation ends the segment, preserving an online decision point.

A coding agent converts each candidate into a parameterized program with declared inputs, outputs, applicability conditions, and termination behavior. Object identities, arm assignments, and spatial arguments are exposed as inputs or recomputed through perception, while reusable operation structure is retained. Each candidate keeps links to its source calls and evidence, making the construction inspectable. The resulting procedure is exercised through the same Runtime as its source operations, with its applicability conditions and termination behavior retained at the interface.

During online rollout, the Task-Level Agent selects a skill and binds its parameters using current observations or valid memory. Guards check the procedure's applicability, arm permissions, and motion limits. The procedure runs without intermediate agent deliberation, while perception and feedback control may continue, and returns to the Task-Level Agent upon completion or exception. Subsequent evidence determines whether to repeat the skill or delegate unresolved alignment to Local. A reused procedure replaces intermediate model calls with explicit program logic. Applicability, repetition, and any need for renewed local interaction remain task-level decisions. Section 4.3 connects these boundaries to task-level reuse and its amortized reasoning cost.

4 Experiments

Our evaluation addresses four research questions through quantitative comparisons and qualitative analysis: (1) Does active observation improve task success? (2) How does adaptive scale affect the trade-off between success and decision steps? (3) How do observations guide coarse and fine manipulation? (4) How can recurring action sequences be packaged as reusable procedures?

Tasks and observations. We study bimanual manipulation in RoboDojo (Chen et al., 2026b). The overall evaluation covers 18 tasks spanning generalization, memory, precision, long-horizon, and open-task regimes (Appendix B). Agents receive camera observations and calibration, measured robot state, and task memory. Task completion and process scores are supplied by the native evaluator for reporting.

Metrics. We report full-task success rate (SR) and mean process score on a 0-100 scale. For the ablations, we jointly report SR and the mean number of *decision steps*: the total Task and Local model decisions in an episode, excluding auxiliary perception-model calls. Means include both successful and unsuccessful episodes. All ablation conditions use the same task-specific physical action budgets, including motion used to acquire observations. Decision steps measure agent deliberation frequency; they are distinct from robot actions and elapsed time.

4.1 Overall Performance and Decision-Space Selection

ARS achieves 68.5% full-task success and a mean process score of 76.9, compared with 22.2% and 29.8 for our GPT-6 reimplementation of the Robocurve interface (Table 1). ARS improves both metrics in every task group, with an overall success-rate gain of 46.3%. This comparison characterizes the complete agent configurations; the targeted ablations below examine observation support and scale selection within ARS.

	ARS (GPT-6 backbone)		GPT-6 + Robocurve†	
Task group	SR (%) ↑	Score ↑	SR (%) ↑	Score ↑
Generalization	60.0	70.0	6.7	16.0
Memory	80.0	80.0	40.0	40.0
Precision	44.4	59.4	0.0	17.2
Long-horizon	75.0	89.2	25.0	34.6
Open-task	100.0	100.0	66.7	66.7
Overall	**68.5**	**76.9**	**22.2**	**29.8**

Table 1: Full-task success rate (SR) and mean process score (0-100) across 18 RoboDojo tasks. Scores include all evaluations and credit partial progress. †Our reimplementation of the Robocurve interface with a GPT-6 backbone.

4.2 How Does Physical Observation Support Manipulation?

Physical observation connects task planning to fine control by supplying evidence for decisions at both scales. The central issue is when to acquire that evidence, how long it remains useful, and which subsequent action it supports. We examine these effects through task-level observation and memory in stacking, and observation-guided correction during insertion.

Observation can support an action long after the view was acquired. In block stacking (Figure 2), wrist inspection grounds the support location in a shared coordinate frame. Retaining this reference in task memory keeps the placement target available across intervening manipulation; renewed perception updates it as the stack changes. The sequence illustrates how a physical observation becomes a persistent input to the decision chain, supporting later placement without a separate local alignment phase.

During insertion, obtaining corrective evidence requires coordinating observation with an ongoing grasp. The holding arm obstructs the head view of the contact region, while the tube limits its own wrist view (Figure 3). Moving the opposite wrist reveals the tube-hole relation while the task arm maintains its hold, providing a reference for lateral correction before advancement. This connects two scales of observation: Task prepares the arm roles and initial observation pose, while Local addresses the alignment uncertainty that remains. The fastening sequence illustrates the same visibility-correction relationship (Appendix C.1).In the GPT-6 zero-shot condition, we occasionally observe the model actively seeking better views during manipulation (Appendix C.4). For example, during tube insertion, GPT-6 identified that the gripper obscured the tube tip in the head view, repositioned the free wrist camera, and subsequently issued an alignment correction informed by the wrist view. These instances offer a glimpse of embodied strategic behavior in a general-purpose frontier model, without task-specific training or a dedicated active-perception mechanism in the interface. They motivate a harness that supports such strategies and connects acquired evidence to sustained manipulation. ARS provides this organization through Task, which establishes the goal, arm roles, and initial observation, and Local, which acquires further evidence and adjusts actions within that scope, returning to Task when broader reconsideration is needed.

The value of observation lies in the execution it supports: retained evidence can guide later actions, while missing contact geometry can justify additional motion before correction.

4.3 Guarded Skill Reuse

Can procedural reuse reduce model deliberation while preserving task completion? We examine two sources of reusable structure: task-level API sequences and repeated local action cycles. In both cases, reuse replaces intermediate model decisions with parameterized execution, while leaving unresolved physical judgments to the agent.

Task-level reuse consolidates familiar API sequences while refreshing their inputs from the current scene. For example, object grounding, segmentation, and grasp proposal can execute as one procedure rather than requiring a separate model selection at each step. This removes repeated orchestration, while retaining instance-specific perception. We distinguish the num-

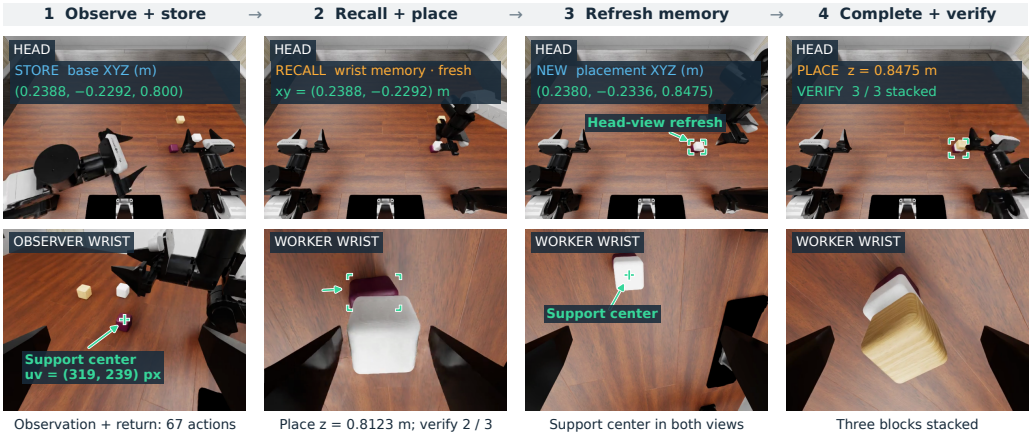

Figure 2: **Observe, remember, and place.** Wrist RGB-D grounds a stored placement target; expired memory triggers head-view refresh before completing the stack. In-image labels summarize operations and measured references

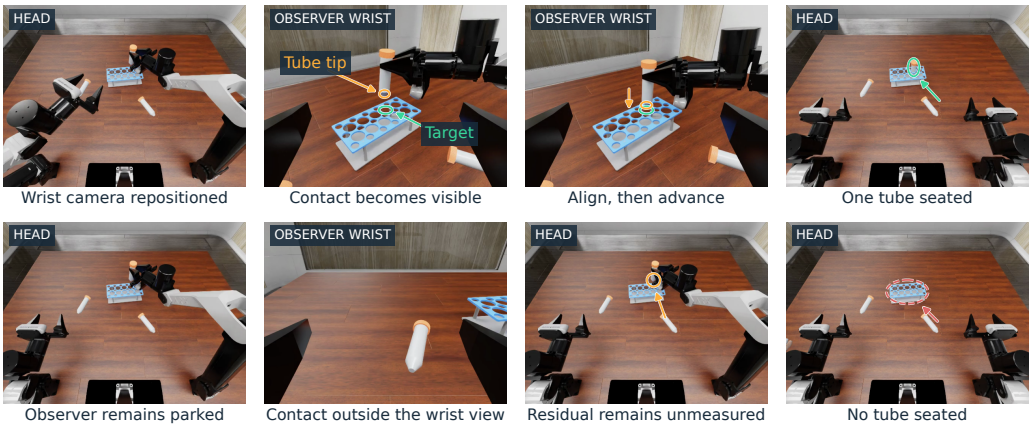

Figure 3: **Observation supplies actionable contact evidence.** Top: wrist-camera repositioning exposes the tube–hole relation for analytic correction, followed by scripted release and single-tube seating. Bottom: no observer motion, unmeasured contact residual, and no seated tube.

ber of model decisions from cumulative model-call time (Figure 4): fewer decisions need not yield proportional time savings when perception and the remaining calls differ in cost. Local action cycles provide a finer-grained source of reuse. In nut fastening, grasping, turning, opening, and resetting form a recurring sequence, whereas alignment and progress require renewed visual assessment (Figure 5). A useful skill therefore ends where fresh interpretation becomes necessary. The cycle executes internally and returns control so that the agent can decide whether to repeat it, correct the alignment, or change its approach. Consolidating such a cycle makes its motion structure available as a reusable operation without fixing those subsequent decisions.

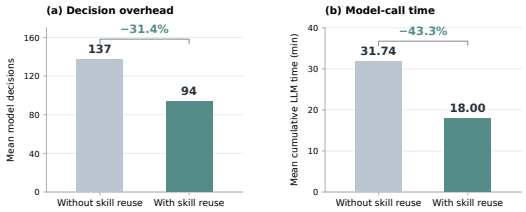

Figure 4: **Reasoning overhead with and without skill reuse.** Panels report mean model decisions and cumulative LLM-call time, including APIs.

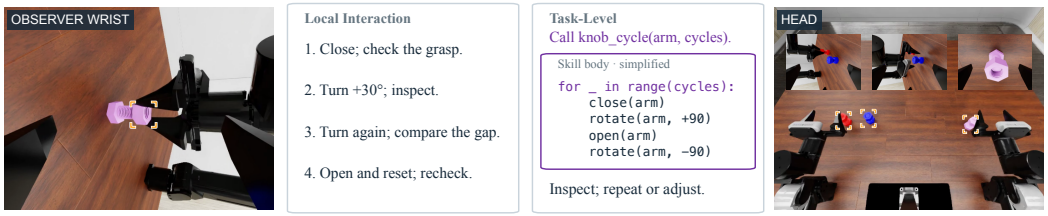

Figure 5: Consolidating recurring local actions into a skill

4.4 Real-World Experiments

We evaluate ARS on a physical dual-arm platform for *stack blocks* and *fill pen holder*, retaining the Opus 5 controller, prompts, and skill library used in simulation. Figure 6 shows the robot stacking the blue block on the orange block and releasing it, and placing the yellow pen into a cup before aligning the grasped red pen with the opening. These results suggest that task-level procedures developed in simulation can transfer to physical manipulation, and show the cross-embodiment generalization.

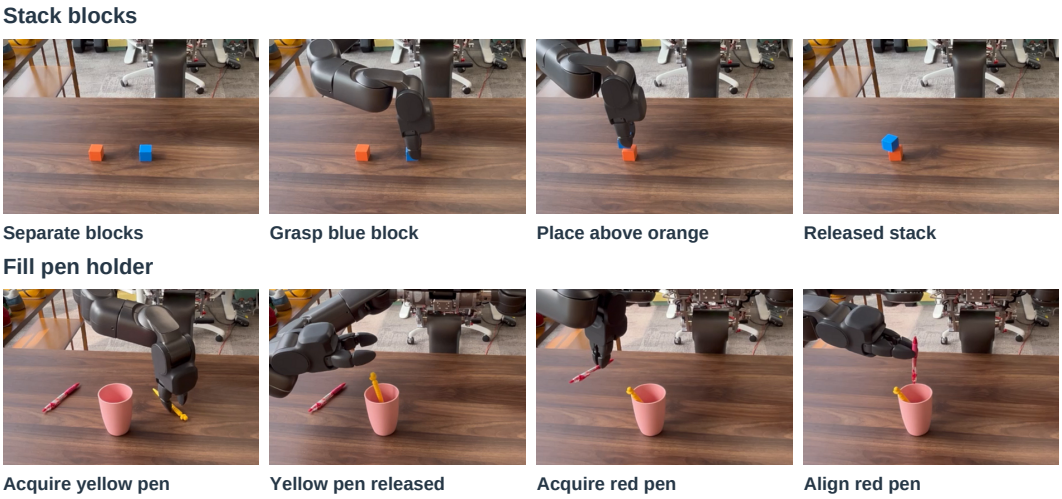

Figure 6: Physical demonstrations of block stacking and pen placement

5 Conclusion

ARS organizes bimanual manipulation around adaptive choices of reasoning and execution scale. Task-level reasoning selects goals, arm roles, and execution procedures, while local reasoning acquires physical views and performs fine corrections within an assigned objective. When new evidence calls for revising that objective or arrangement, control returns to task-level reasoning. Reusable procedures preserve established action sequences, leaving subsequent interpretation and replanning to the agents. Our evaluation shows improved overall task performance over the implemented baseline, while execution traces illustrate the roles of observation, correction, and procedural reuse. Together, these findings support an organization of robot agency in which physical evidence informs both what to execute next and when further reasoning is needed.

Limitations. ARS currently relies on a capable frontier VLM to interpret spatial relations, select useful observation viewpoints, and determine when task goals or arm roles need revision. Errors in these judgments can limit the effectiveness of active observation and local correction. The framework is also constrained by the coverage and reliability of its robot APIs and execution harness. Extending the API repertoire and strengthening execution feedback, failure handling, and recovery would help support a broader range of manipulation tasks.

AI USE STATEMENT

Generative AI tools were used to assist with experimental setup, code development and debugging, routine experiment implementation and execution, organization and presentation of results, literature searches, and manuscript preparation and language polishing. These tools also assisted with editing experimental images. Images from real-robot experiments were edited to improve visual presentation and protect confidential information about the company workspace. The authors take responsibility for the final manuscript, figures, code, experimental results, and scientific claims.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Michel Breyer, Lionel Ott, Roland Siegwart, and Jen Jen Chung. Closed-loop next-best-view planning for target-driven grasping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1411–1416, 2022.

François Chaumette and Seth Hutchinson. Visual servo control, part I: Basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006.

Kaiyuan Chen, Shuangyu Xie, Letian Fu, Justin Yu, William Pacini, Sandeep Bajamahal, Hudson Kim, Jaimyn Drake, Daehwa Kim, Haoru Xue, Jonathan Francis, Christian Juette, Peter Schaldenbrand, Muhammet Yunus Seker, Ruwan Wickramarachchi, Uksang Yoo, Guanzhi Wang, Adithyavairavan Murali, Balakumar Sundaralingam, S. Shankar Sastry, Spencer Huang, Yuke Zhu, Linxi Fan, and Ken Goldberg. GaP: A graph-as-policy multi-agent self-learning harness for variational automation tasks. *arXiv preprint arXiv:2607.05369*, 2026a.

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Weijie Wan, Baijun Chen, Haoran Lu, Haowen Yan, Honghao Su, et al. RoboDojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026b.

William Chen, Jagdeep Singh Bhatia, Catherine Glossop, Nikhil Mathihalli, Ria Doshi, Andy Tang, Danny Driess, Karl Pertsch, and Sergey Levine. Steerable vision-language-action policies for embodied reasoning and hierarchical control. *arXiv preprint arXiv:2602.13193*, 2026c.

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

Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Dantong Niu, Fei-Fei Li, 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. *arXiv preprint arXiv:2603.22435*, 2026.

Yuxin He, Ruihao Zhang, Tianao Shen, Cheng Liu, and Qiang Nie. Towards exploratory and focused manipulation with bimanual active perception: A new problem, benchmark and strategy. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

Hengyuan Hu, Priya Sundaresan, Jensen Gao, and Dorsa Sadigh. VIA: Visual interface agent for robot control. *arXiv preprint arXiv:2607.11119*, 2026.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. ReKep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

Fanqi Lin, Ruiqian Nai, Yingdong Hu, Jiacheng You, Junming Zhao, and Yang Gao. OneTwoVLA: A unified vision-language-action model with adaptive reasoning. In *International Conference on Learning Representations (ICLR)*, 2026.

Zhenyang Liu, Yongchong Gu, Yikai Wang, Xiangyang Xue, and Yanwei Fu. ActiveVLA: Injecting active perception into vision-language-action models for precise 3d robotic manipulation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8141–8151, 2026.

Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, Mosharaf Chowdhury, Yuke Zhu, Linxi Fan, and Guanzhi Wang. ASPIRE: Agentic /Skills discovery for robotics. *arXiv preprint arXiv:2607.00272*, 2026.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL `https://arxiv.org/abs/2504.16054`.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Guokang Wang, Hang Li, Shuyuan Zhang, Di Guo, Yanhong Liu, and Huaping Liu. Observe then act: Asynchronous active vision-action model for robotic manipulation. *IEEE Robotics and Automation Letters*, 10(4):3422–3429, 2025.

Kuancheng Wang, Vaibhav Saxena, Shuo Cheng, Yotto Koga, and Danfei Xu. Learning to see while learning to act: Diffusion models for active perception in robot imitation. *arXiv preprint arXiv:2606.23625*, 2026a.

Yilong Wang, Cheng Qian, Ruomeng Fan, and Edward Johns. Observer-actor: Active vision imitation learning with sparse-view gaussian splatting. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026b.

Haoyu Xiong, Xiaomeng Xu, Jimmy Wu, Yifan Hou, Jeannette Bohg, and Shuran Song. Vision in action: Learning active perception from human demonstrations. In *Conference on Robot Learning (CoRL)*, volume 305 of *Proceedings of Machine Learning Research*, pp. 5450–5463, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Junyi Zhang, Jiaxin Ge, Hanjun Yoo, Letian Fu, Zihan Yang, Yaowei Liu, Raj Saravanan, Shaofeng Yin, Justin Yu, Dantong Niu, Zirui Wang, Roei Herzig, Ken Goldberg, Yutong Bai, David M. Chan, Ion Stoica, Angjoo Kanazawa, Jiahui Lei, Haiwen Feng, and Trevor Darrell. Playful agentic robot learning. *arXiv preprint arXiv:2606.19419*, 2026a.

Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, Zhengru Fang, Yi Nie, Changxu Wei, Yu Wang, Wenbo Ding, and Chao Yu. Harness VLA: Steering frozen VLAs into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*, 2026b.

A AGENT TOOL INTERFACES AND SHARED CONTEXT

The current ARS interface separates robot APIs, reusable skills, and agent-control tools. A robot API provides a measurement, a specified motion, or bounded control of a single geometric objective. A skill composes several operations into a procedure with a defined endpoint. The public robot interface contains 16 APIs and 9 skills.

A.1 ROBOT APIS

Table 2 lists the robot APIs available to each agent. Task has access to 15 APIs; Local has access to five, subject to the operations, arm roles, and limits specified in its delegation contract.

API	Task	Local	Function
Perception and geometric queries			
`locate`	✓	–	Ground an object or surface in an observation and return a reference.
`inspect`	✓	–	Measure object geometry, an axis and endpoint, a support plane, or an opening.
`grasp_candidates`	✓	–	Generate grasp candidates for a referenced object and a specified arm.
`check_relation`	✓	–	Measure a specified geometric relation.
`reach_query`	✓	–	Query reachability or a candidate motion chain without moving the robot.
Motion and view acquisition			
`move_to`	✓	–	Move an arm's tool center point (TCP) to a specified pose.
`shift`	–	✓	Apply a position correction in the reference camera's image plane.
`rotate`	✓	✓	Rotate about a specified axis in a reference frame.
`advance`	✓	✓	Move a bounded distance along a specified direction.[a]
`gripper`	✓	✓	Open or close a gripper.
`observe`	✓	✓	Move a physical wrist camera to acquire a view.
`wait`	✓	–	Advance simulation time while maintaining arm and gripper commands.
Bounded geometric controllers			
`orient_held`	✓	–	Align a measured held-object axis with a target direction.
`carry_tip_to`	✓	–	Translate a held working point to a target while preserving orientation.
`center_over`	✓	–	Reduce lateral image error using the acting arm's wrist camera.
`advance_until`	✓	–	Advance along an axis until a stopping condition or travel bound is reached.

Table 2: Robot API access in the two-agent configuration. A checkmark indicates that an API can be exposed to the agent; Local access additionally requires contract authorization. Bounded controllers may execute several corrections while retaining a single geometric objective.

[a] Task specifies the direction; Local inherits it from the contract.

The `inspect` API provides four measurement modes: `shape`, `axis_tip`, `support_plane`, and `opening`. These measurements are derived from RGB-D observations. The `check_relation` API reports geometric evidence, including an unknown status when the relation cannot be measured; it does not expose the benchmark evaluator's task-success flag.

A.2 REUSABLE SKILLS

Task invokes the skills in Table 3 through `run_skill(skill_id, bindings)`. Each skill executes its prescribed stages and returns an execution receipt. Skill execution does not invoke the Task or Local policy model, although perception services may be called within a procedure. The returned evidence supports the agent's next decision.

Skill	Composed operations
`pick_and_hold`	Select and execute a grasp, then lift while maintaining wrist orientation.
`stage_near`	Approach a target, measure the reached pose, and create a contact reference.
`lift_upright`	Lift a held object to provide clearance, then reorient it.
`transport_held`	Lift, translate laterally, and lower while maintaining orientation and the grasp.
`place_for_regrasp`	Reach a placement pose, open the gripper subject to release checks, and retreat.
`seat_probe`	Align a held working point and perform bounded axial probing while maintaining the grasp.
`press_to_contact`	Approach from a hovering pose, press toward a surface, and retract.
`knob_cycle`	Close the gripper, rotate, open, and reset; optionally withdraw before resetting.
`tool_stroke`	Position a held tool's working point, execute a bounded stroke, and retract.

Table 3: The nine public reusable skills. Completion of the listed operations reports execution status; task success is assessed separately.

Local access to a compound action. Local can additionally call a dedicated `knob_cycle` tool when its contract authorizes rotation and gripper operations. This tool invokes the same skill with one cycle and zero withdrawal distance, subject to the contract's release and rotation checks. It does not give Local access to the general `run_skill` interface.

A.3 AGENT CONTROL AND SHARED CONTEXT

Task uses `remember` to write shared notes, `run_skill` to invoke a procedure, `interact` to delegate a bounded objective, and `finish` to end an episode with its outcome report. When skill descriptions are loaded on demand, `skill_info` provides their input contracts. Local uses `return_to_task` to end a delegation and report its outcome and supporting evidence. Together with the five robot APIs above, this gives Local six base tools and the optional `knob_cycle` tool.

Both agents receive current images and robot measurements and can read shared notes and geometric references. Task writes the notes; recent tool histories remain specific to each agent session. Geometric references retain their observation provenance and do not automatically track object motion. Measurements must be refreshed when scene changes invalidate their supporting evidence.

B Evaluation Protocol and Task Coverage

B.1 Task Set and Adapter Assumptions

Table 4 lists the 18 tasks in our study and their adapter assumptions or evaluation boundaries. This is a selected subset supported by our adapters, rather than a random sample of RoboDojo. Matched comparisons require the same task set and eligibility criteria across conditions; low performance alone does not justify removing a task from an aggregate.

Task	Regime	Adapter assumption or evaluation boundary
Stack blocks	Gen.	Shared placement execution and verification.
Stack bowls	Gen.	Rim grasps; transfer between arms is available.
Pack objects into box	Gen.	Placement slots based on object footprints; transfer for transport across the workspace.
Arrange largest number	Gen.	Visual digit recognition and shared placement execution.
Sort nesting dolls	Gen.	Size ordering estimated from observed geometry.
Cover blocks	Mem.	Remembered object identities and inspection after uncovering.
Match and pick from conveyor	Mem.	Sequential observations; waiting consumes actions.
Swap blocks	Mem.	Remembered target poses and renewed perception.
Swap T	Mem.	Initial object outlines retained as pose references.
Press by number	Mem.	Ordered presses; reset-drive modification treated as an explicit environment setting.
Insert key	Prec.	Acquiring the key is distinct from completing insertion.
Insert tubes	Prec.	Inserting one tube is distinct from full-task success.
Fasten screws	Prec.	Included in overall coverage; seating remains a limitation.
Fill pen holder	Long	Tip-down placement through the shared executor.
Classify objects	Long	Visual category assignment; transfer via the table is available.
Put bottles into dustbin	Long	Any container-location prior is specified and shared across matched conditions.
Play tic-tac-toe	Long	Opponent moves observed through sequential images.
General pickup	Open	Grasp attempts and structured execution refusals.

Table 4: Task scope and adapter assumptions. Gen., Mem., Prec., Long, and Open denote generalization, memory, precision, long-horizon, and open-task regimes, respectively. These entries describe task conditions and evaluation boundaries. Fasten screws is included in overall coverage but excluded from the matched scale-selection comparison.

Task profiles, demonstrations, geometry priors, and authored procedures are specified resources and are held constant across matched variants. Model decisions are evaluated within these shared conditions. The press-by-number reset-drive modification, any container-location prior, grasp attachment, and motion configuration are explicit experiment settings. Grasp attachment is held constant within a comparison, and its simulator state is never exposed to the policy. Results from patched and unpatched environments are reported separately.

C Additional Experiments and Execution Examples

C.1 Observer Views Reveal Fastener Misalignment

Figure 7 shows a scripted fastener-alignment example. The left arm holds a red nut while the right wrist camera views the nut and bolt from the side. The holding wrist camera looks nearly along the approach direction, whereas the side view reveals a lateral offset before descent. Using geometric measurements from the observer and head views, the alignment routine reduces its estimated lateral offset from 11.9 to 0.2 mm during the staging correction. The sequence illustrates how an observer view can reveal geometry relevant to a correction while the holding arm maintains its grasp.

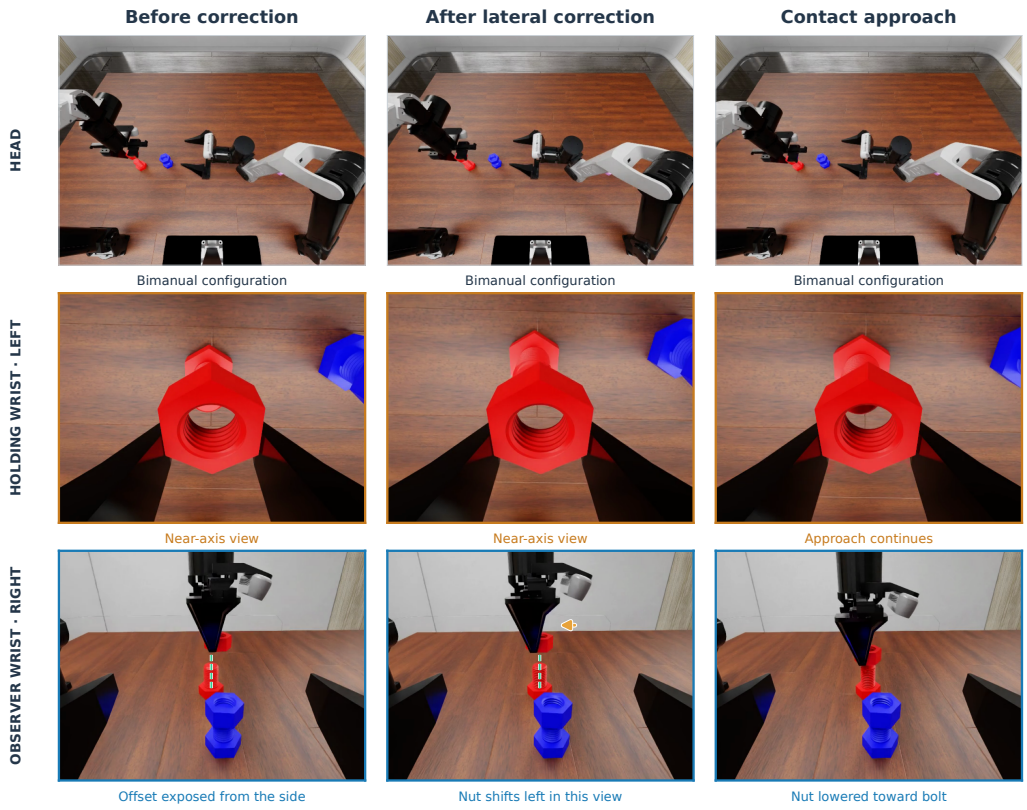

Figure 7: Observer-view evidence during scripted fastener alignment. Rows show the head, holding-wrist, and observer-wrist views; columns show correction and approach. The dashed bolt reference and leftward arrow are author-added visual guides, not measured displacement vectors.

C.2 Remembering Swap Targets and Correcting Placement

Swapping two objects requires retaining their target poses while tracking changes to their current poses. Figure 8 illustrates this distinction in a `swap_T` episode (seed 0, layout 1). Task first calls `remember` to record the objects' initial positions and orientations, which define the swap targets. After placing both blocks, it compares the red block with its stored target and updates the current-state note: the red block has been released with a position and orientation error and should be regrasped, while the blue block should remain undisturbed. Task then selects a right-arm grasp, rotates and repositions the red block, and releases it again. This sequence shows how shared memory connects post-placement evidence to a revised plan for one object while retaining the original task targets.

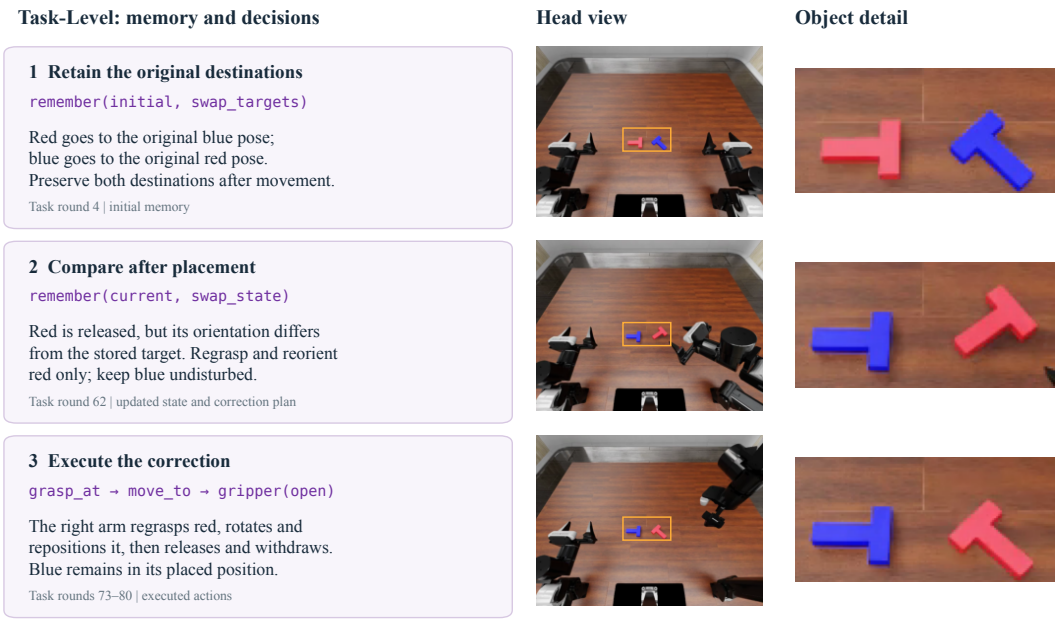

Figure 8: Memory-guided placement correction. Left: summaries of recorded Task memory writes and subsequent actions. Right: video frames before manipulation, after the first placement, and after corrective actions; enlarged details correspond to the orange boxes. The initial target is retained while the current-state note is updated. Text summarizes recorded tool arguments and actions.

C.3 Observation Actions in Demonstration Learning

A successful demonstration may depend on visual access or prior knowledge that is absent from the learner's recorded observations. Vision in Action addresses this issue by learning active perception from human demonstrations in an observation space shared by the operator and robot (Xiong et al., 2025). Here, we study whether including observation actions in scripted demonstrations improves subsequent policy learning. This data-design study is separate from the main evaluation of online decision-scale coordination.

We fine-tune $\pi_{0.5}$ (Physical Intelligence et al., 2025) on paired tube-insertion demonstration sets with $N \in \{50, 100, 200\}$ demonstrations per condition. The initial layouts and manipulation logic are shared; active demonstrations additionally move the other arm's wrist camera to view the insertion region. The main comparison uses 5,000 training updates for both conditions. Evaluation uses 20 held-out layouts and the official process score, which gives partial credit for insertion and release stages. For each training seed, we include only layouts with completed evaluations under both conditions, yielding 17-20 paired results. We then average the layout scores within each seed and weight the seed means equally.

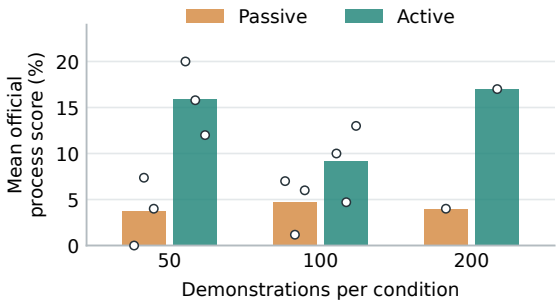

Figure 9: Insertion progress after training on demonstrations with and without observation actions. Bars show averages of training-seed means; points show the individual seed means. There are three training seeds at each of $N = 50$ and $N = 100$, and one at $N = 200$. Each seed contributes 17-20 complete evaluation pairs from 20 held-out layouts; missing pairs are excluded. Values are process scores expressed as percentages, not success rates.

At 5,000 updates, active demonstrations yield higher mean process scores at each data budget (Figure 9): 15.9% versus 3.8% at $N = 50$, 9.2% versus 4.7% at $N = 100$, and 17.0% versus 4.0% at $N = 200$. Active scores are higher for all three training seeds at both $N = 50$ and $N = 100$. These results support including observation actions in demonstrations under the tested training budget. An additional run at $N = 100$ with training seed 42 and 10,000 updates yields 5.0% for active demonstrations and 8.0% for passive demonstrations, indicating sensitivity to training duration.

C.4 Active View Acquisition in the GPT-6 Zero-Shot Condition

Figure 10 illustrates active view acquisition during tube insertion in the GPT-6 zero-shot condition. As the right arm holds the tube near the rack, the model reports that the gripper obscures the tube tip in the head view. It then moves the unoccupied left arm to reposition its wrist camera near the rack. After receiving the new wrist images, the model reports seeing openings beside the held tube and adjusts the right-arm position, lowering the tube tip further. The four columns show synchronized head and left-wrist views. The visual assessments are taken from the model's recorded action explanations.

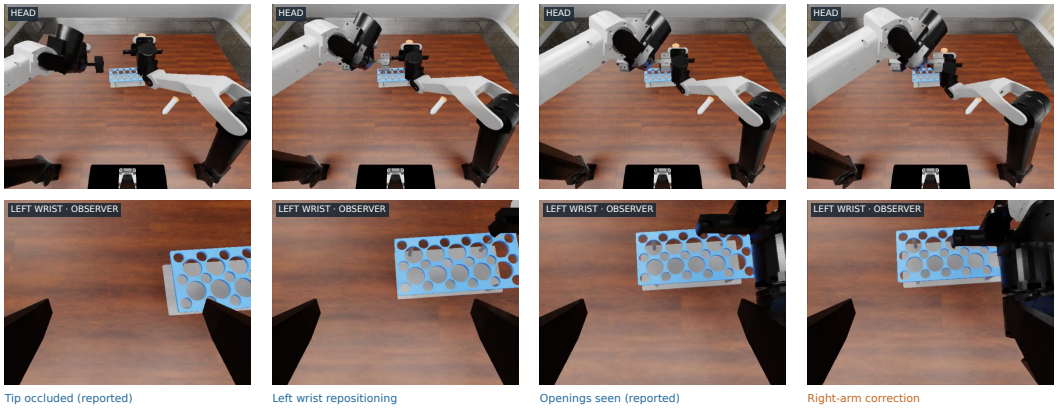

Figure 10: Active view acquisition during tube insertion in the GPT-6 zero-shot condition. Synchronized head and left-wrist views show the reported occlusion, observer-arm movement, and subsequent right-arm correction.

The sequence connects a reported observation need to a physical viewpoint change and a subsequent manipulation correction. It illustrates the model's use of arm motion to acquire evidence for its next action. An observation pose must also leave room for subsequent manipulation, so view selection and action planning share physical constraints. ARS organizes these choices through its two agents: Task specifies the goal and arm roles, while Local acquires views and makes authorized corrections. If the assigned arrangement needs revision, Local returns its evidence to Task for further planning.