RECURSIVE HARNESS DISTILLATION ACROSS AGENTS FOR ROBOT MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A central goal in robotics is to enable manipulation across changing tasks and environments. Vision-language-action (VLA) models provide broad manipulation capabilities but can struggle when execution requires diagnosing failures and adapting behavior. Strong agents can discover effective interventions through interaction with these policies. We propose **Recursive Harness Distillation** to accumulate this experience as reusable guidance across agents. A strong agent distills its experience into a playbook for a light agent, then recursively refines the playbook using the light agent's execution feedback. The resulting playbook enables agents to reuse accumulated intervention knowledge in new task instances without updating model parameters. In real-world manipulation, the harness improves success from 37.3% to 64.0%. On SimplerEnv Bridge, the light agent with the playbook achieves 66.7% success, compared with 41.7% for the GR00T-only baseline, and outperforms the strong agent without a playbook. The same playbook also benefits the strong agent, which reaches 79.2% success. These results demonstrate the feasibility of harness distillation for robotics: intervention experience can be accumulated, refined through execution, and reused across agents to improve manipulation.

1 INTRODUCTION

A persistent vision in robotics is to develop systems that can apply their manipulation capabilities across diverse tasks and environments. Vision-language-action (VLA) models bring this goal closer by combining pretrained visual and linguistic representations with robot action learning (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024). Using these policies across changing scenes and extended tasks also requires interpreting goals, monitoring execution, and responding when behavior departs from the intended outcome (Huang et al., 2022; Hu et al., 2026). Existing approaches use language models to generate executable robot programs (Liang et al., 2023) and to coordinate learned policies and manipulation tools through planning, monitoring, and recovery (Chen et al., 2026).

Understanding the task does not by itself tell an agent how to elicit the intended behavior from a deployed VLA (Jeong et al., 2026). Even when the agent identifies the correct subgoal, it must determine how the policy will respond to the current scene and how to intervene when execution goes wrong. This requires connecting physical observations, policy behavior, and the effects of different interventions. Experience with the policy helps the agent learn which adjustments work in which situations.

Highly capable models such as Astra can use this understanding to interpret scenes, diagnose failures, and adjust execution. However, relying on these models throughout execution is costly, motivating the use of lighter, lower-cost models for deployment. Transferring expertise to these models also consumes resources: the capable model must explore the task, analyze its experience, and prepare useful guidance. This raises a question: under a limited token budget, how can we distill this expertise to help a lighter model effectively operate the VLA?

In this paper, we propose Recursive Harness Distillation (RHD), a framework that distills a capable agent's experience into a reusable harness for a lighter agent (Figure 1). The harness combines an interface for observing and intervening in VLA execution with a playbook that guides its use. A capable agent first interacts with the VLA and records effective strategies in the playbook. A lighter agent then executes tasks using this harness, and the capable agent reviews the resulting experience

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 revise the guidance. Through this repeated process, the harness captures both what works for the VLA and what the lighter agent needs to apply it.

We evaluate RHD on SimplerEnv Bridge and three real-world manipulation tasks. In real-world experiments, the harness improves success from 37.3% to 64.0%. On SimplerEnv Bridge, adding the light agent without a playbook yields only a modest improvement over the GR00T-only baseline, from 41.7% to 43.8%. With the playbook, the same light agent achieves 66.7% success, outperforming the strong agent operating without a playbook. These benefits extend to the strong agent itself, which achieves 79.2% success when equipped with the same playbook. Together, these results show that experience acquired through interaction with a VLA can be accumulated, refined, and reused across agents to improve robotic manipulation.

2 Related Work

Vision-Language-Action Policies. VLA models have emerged as a promising approach to generalist robot policies, combining pretrained vision-language representations with learning from diverse robot demonstrations (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; Bjorck et al., 2025). Training on diverse manipulation data enables a shared policy to perform multiple tasks conditioned on language instructions (Walke et al., 2023; O'Neill et al., 2024; Team et al., 2024). Their behavior depends on how instructions are grounded in visual observations and translated into motor commands. However, broad task coverage does not guarantee reliable execution under unfamiliar visual conditions, where existing models exhibit generalization limitations (Zhou et al., 2025; Liu et al., 2026). VLA models can struggle to translate learned manipulation capabilities into reliable behavior as scenes and execution states change. Our Recursive Harness Distillation enables an external agent to adapt its guidance to these conditions by accumulating and reusing experience from policy interventions.

Harness. External control structures connect language-model reasoning to robot execution through interfaces for perception, action, and feedback (Ahn et al., 2022; Huang et al., 2022; 2023; Yao et al., 2022). Code as Policies (Liang et al., 2023) uses language models to generate control programs that process perceptual outputs and compose robot APIs. More recent approaches incorporate learned policies as callable tools, enabling agents to decompose tasks, monitor progress, and intervene when execution fails (Chen et al., 2026; Galanti et al., 2026). These approaches provide mechanisms for guiding execution, but do not by themselves explain how effective intervention strategies can be accumulated and transferred across agents. Our RHD develops reusable guidance from policy interactions and refines it based on how subsequent agents apply it, with the goal of reducing repeated reasoning and trial and error.

Knowledge Distillation. Knowledge distillation transfers a teacher's predictive behavior to a student through training targets (Hinton et al., 2015), including teacher-generated rationales that supervise the student's reasoning (Hsieh et al., 2023). Student feedback can also guide the transfer itself, informing teacher updates and revisions to the knowledge provided in subsequent rounds (Liu et al., 2021; Zhou et al., 2022; Chen et al., 2025). These methods motivate adapting what is taught to what the student can use. A complementary line of work on LLM agents retains textual reflections and extracts reusable insights from task experience (Shinn et al., 2023; Zhao et al., 2024), while further approaches organize and refine this knowledge through generation, reflection, and curation (Zhang et al., 2026). RHD brings these ideas to VLA manipulation through a recursive teacher–playbook–student process. A strong agent initializes the playbook through policy interaction, then revises it using the light agent's subsequent executions. The object of refinement is external guidance for recognizing when and how to intervene on the VLA, grounded in the physical outcomes of the recipient agent's decisions.

3 Method

3.1 Problem Formulation

We consider manipulation tasks drawn from a distribution $\mathcal{D}$, each specifying a goal instruction I and an initial scene. A pretrained VLA π_θ generates robot actions from observations and instructions. Its

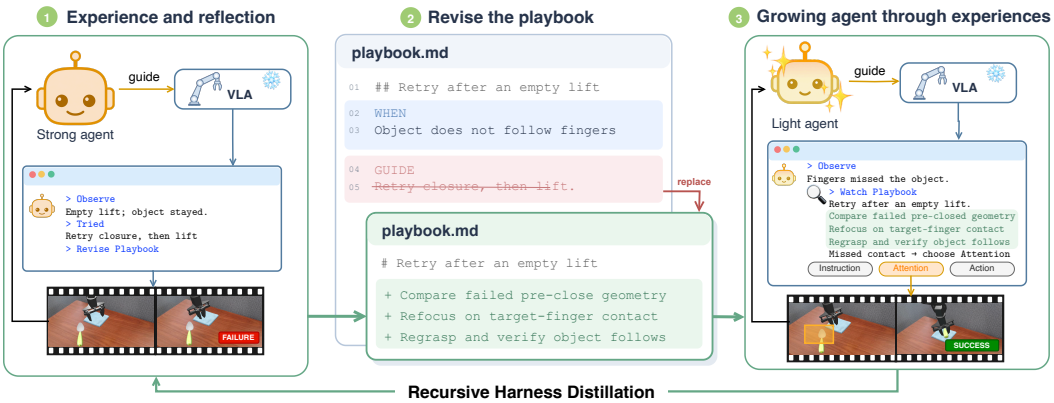


Figure 1: **Recursive Harness Distillation.** (1) A strong agent interacts with a frozen VLA and reflects on execution outcomes to identify useful intervention guidance. (2) This experience is distilled into a playbook that specifies when to intervene, how to adjust execution, and what to check afterward. The illustrated revision replaces a simple retry instruction with guidance to inspect the failed grasp, refocus on target–finger contact, and verify that the object follows the gripper. (3) A light agent consults the playbook to guide VLA execution through instruction, attention, and action interventions. Its execution experience returns to the strong agent for further playbook refinement, closing the recursive loop.

parameters θ remain fixed. An external agent observes execution and intervenes through an interface that exposes the instruction, internal feature attention, and action output. We denote the strong agent by T and the lighter deployment agent by S. Their model parameters also remain fixed; knowledge acquired through execution is retained in a playbook P.

An episode has a finite horizon H. At step t, h_t contains the goal, observations, previous interventions, and executed actions available up to that step. The underlying physical state need not be fully observed. The interface presents the relevant information from h_t and makes a prescribed set of intervention operations available to the agent. Each operation includes an identity choice, so the agent can leave the corresponding part of the VLA unchanged. The playbook guides the agent's choice of operations and their arguments. Let b_t denote the complete decision record at step t, including the chosen operations, the proposed VLA action, and the executed action. The agent and VLA together induce an execution policy $\Pi_{S,P}(b_t \mid h_t)$. Retaining this record makes the feedback available to subsequent decisions explicit. We seek a playbook that improves the task success of this composed policy.

3.2 Intervening on VLA Execution

We expose three intervention sites along the VLA's computation. An instruction operation transforms the language input, an attention operation modifies internal feature attention during inference, and an output operation transforms the generated action. At decision point t, let o_t denote the current observation provided to the frozen VLA π_θ, and let I denote the original task instruction. The agent's instruction, attention, and output interventions are denoted by u_t^I, u_t^F, and u_t^A, respectively. For a candidate action sequence, these interventions are represented as

$$\widetilde{I}_t = f_I(I, u_t^I), \qquad \widehat{a}_t = \pi_\theta(o_t, \widetilde{I}_t; u_t^F), \qquad a_t = f_A(\widehat{a}_t, u_t^A). \tag{1}$$

Here, f_I maps the original instruction and the instruction intervention to the modified instruction $\widetilde{I}_t$. The frozen VLA produces the proposed action sequence $\widehat{a}_t$, and f_A maps this proposal and the output intervention to the candidate action sequence a_t. The admissible transformations of f_A are specified by the interface. The semicolon denotes an intervention on the VLA's forward computation: u_t^F modifies selected attention computations over features while preserving the learned parameters θ. A decision need not advance the environment: the agent may request and inspect alternative candidates at the same environment step. Upon approval, the executor applies only the selected prefix of the approved candidate, within the interface's execution limits, and returns a new observation. Thus, an agent decision does not necessarily correspond to executing an entire action chunk.

These intervention sites do not define a mandatory sequence of agent operations. The agent may select any admissible subset without first attempting the others. Instruction and attention interventions affect candidate generation, whereas output interventions modify candidate actions; this computational dependency does not impose an order in which the agent must try the intervention sites. Available operations can be combined within a decision; their identity settings recover the unmodified VLA. The choice of intervention site is part of the agent's decision. The playbook specifies the situations in which an operation is useful, how to apply it, and which subsequent observations indicate that it had the intended effect.

3.3 Policy Optimization View: Improving the Recipient's Decisions

A playbook changes robot behavior through the agent that interprets it. We therefore view Recursive Harness Distillation as policy optimization over the policies induced by the deployment agent. Let $Y(\tau) \in \{0, 1\}$ indicate whether trajectory τ achieves the original task goal within the horizon. The objective is

$$P^\star \in \arg\max_{P \in \mathcal{P}} J_S(P), \qquad J_S(P) = \mathbb{E}_{\tau \sim p_{\mathcal{D}}(\tau; \Pi_{S,P})}[Y(\tau)], \tag{2}$$

where $\mathcal{P}$ denotes the set of admissible playbooks. Here, $p_{\mathcal{D}}(\tau; \Pi_{S,P})$ denotes the trajectory distribution induced by $\Pi_{S,P}$ on tasks and initial scenes drawn from $\mathcal{D}$. The task distribution, VLA, intervention interface, and execution horizon are held fixed. The objective evaluates how the light agent uses the guidance, while the strong agent supplies experience and proposes revisions to it.

A playbook distilled from the strong agent's execution experience provides an initialization. Subsequent revisions use recipient rollouts, including failures, to refine when and how the guidance should be applied. The light agent can choose different interventions under the same guidance, and those decisions change the situations encountered later in the episode. The relevant experience distribution is thus induced by the recipient. This is closely related to the role of learner-induced distributions in sequential imitation learning (Ross et al., 2011).

The dependence on the recipient's execution can be made explicit using the finite-horizon performance-difference identity (Kakade & Langford, 2002; Schulman et al., 2015). Let k index playbook revisions. For the following identity, t indexes agent–interface decisions, and $N(\tau)$ denotes their number before episode termination under the robot-step horizon H. A decision may generate a candidate without advancing the robot or execute a variable-length action prefix. For the current playbook P_k, let $V_k(h_t)$ be the probability of success when continuing with Π_{S,P_k}, and let $Q_k(h_t, b)$ be the success probability after the decision recorded in b and then continuing with that policy. Define $\mathcal{A}_k(h_t, b) = Q_k(h_t, b) - V_k(h_t)$. For a candidate playbook P', the change in success is

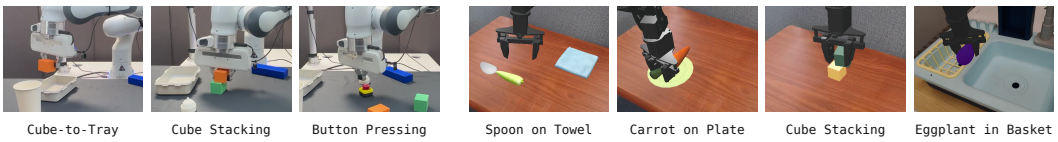
$$J_S(P') - J_S(P_k) = \mathbb{E}_{\tau \sim p_{\mathcal{D}}(\tau; \Pi_{S,P'})} \left[\sum_{t=0}^{N(\tau)-1} \mathcal{A}_k(h_t, b_t) \right]. \tag{3}$$

The identity follows by telescoping the continuation values along the candidate policy's trajectory; a derivation is provided in Appendix A. It identifies the improvement target: a revision should induce better decisions at the histories the recipient actually encounters. Reproducing the teacher's decisions on its own histories alone does not evaluate this quantity. The values in Eq. 3 characterize the objective; we evaluate proposed revisions through complete recipient rollouts and their task outcomes.

3.4 Recursive Harness Distillation (RHD)

The strong agent first executes tasks through the intervention interface, producing experience $\mathcal{E}_T$. It extracts an initial playbook P_0 that relates observable situations to intervention choices and their observed consequences. The light agent then executes with P_k, producing experience $\mathcal{E}_{S,k}$. The strong agent reviews how the guidance was applied, which interventions were selected, and how the VLA and environment responded. It uses this evidence to propose revisions that address ambiguities, incorrect applicability conditions, and ineffective intervention choices.

We formulate the update as a search over playbooks proposed by the teacher. Let $\mathcal{G}_T(P_k, \mathcal{E}_T, \mathcal{E}_{S,0:k})$ denote a finite set of proposed revisions based on the available teacher experience $\mathcal{E}_T$ and accumulated



(a) Real-world tasks (b) Simulator tasks

Figure 2: **Experimental setups.** (a) Three real-world tasks: Cube-to-Tray, Cube Stacking, and Button Pressing. (b) Four SimplerEnv Bridge tasks: Spoon-on-Towel, Carrot-on-Plate, Cube Stacking, and Eggplant-in-Basket.

recipient experience $\mathcal{E}_{S,0:k}$, where $\mathcal{E}_{S,0:k}$ denotes experience collected from revisions 0 through k. Proposed revisions may be generated and evaluated sequentially. Let $\mathcal{C}_k$ collect the current playbook and the revisions proposed during the search. The set is constructed incrementally as proposals are generated:

$$\mathcal{C}_k = \{P_k\} \cup \mathcal{G}_T(P_k, \mathcal{E}_T, \mathcal{E}_{S,0:k}), \qquad \mathcal{C}_k \subseteq \mathcal{P}. \tag{4}$$

Each proposed revision is fixed before evaluation with the light agent on a development batch of n task instances. Within a development round, the same instances are reused across evaluated revisions. Denoting these rollouts by $\tau_i^{S,P}$, we measure empirical success. When a candidate meets the target, the accepted playbook satisfies:

$$P_{k+1} \in \left\{ P \in \mathcal{C}_k : \widehat{J}_{S,k}(P) \geq \eta \right\}, \qquad \widehat{J}_{S,k}(P) = \frac{1}{n} \sum_{i=1}^{n} Y(\tau_i^{S,P}). \tag{5}$$

Here, η is the teacher's empirical success rate on the same development instances. Candidates are proposed and evaluated sequentially, and the first candidate satisfying Eq. (5) is accepted. If no candidate meets the target, execution feedback guides further proposals. The acceptance rule does not guarantee termination; until a candidate meets the target, no update is accepted under this rule. Subsequent rounds collect new experience under the revised playbook. The resulting recursion couples knowledge transfer to its actual use: teacher experience starts the process, and recipient experience continually informs what the teacher should convey next.

4 Experiments

We evaluate whether experience distilled into a playbook improves the use of a frozen VLA, and whether these gains extend from a light agent to a more capable agent and to physical robot execution.

4.1 Experimental Setup

Simulation Setup. We evaluate on four WidowX manipulation tasks in the Bridge setting of SimplerEnv (Li et al., 2024): spoon-on-towel, carrot-on-plate, cube stacking, and eggplant-in-basket (Figure 2b). We use the GR00T-N1.7-SimplerEnv-Bridge checkpoint and keep its parameters fixed throughout playbook development and evaluation. The policy receives a 256×256 RGB image, robot proprioception, and a language instruction. We reserve 12 initial configurations for development and 12 for evaluation, yielding 48 instances per split.

Real-World Setup. We use a 7-DoF Franka Panda with wrist, front, and right-side RGB cameras for three tasks (Figure 2a): Cube-to-Tray, which places an orange cube in a tray; Cube Stacking, which places the orange cube on a green cube; and Button Pressing, which presses a red emergency stop button. We collect 50 human-teleoperated demonstrations per task, randomizing object positions across demonstrations, and fine-tune $\pi_{0.5}$ (Intelligence et al., 2025) from its base checkpoint. The fine-tuned policy is then fixed, and the harness uses the same underlying policy execution settings as the policy-only baseline. At evaluation time, object positions are randomized before every trial within a $50\text{cm} \times 50\text{cm}$ workspace, while remaining visible from the wrist camera. This requires the policy

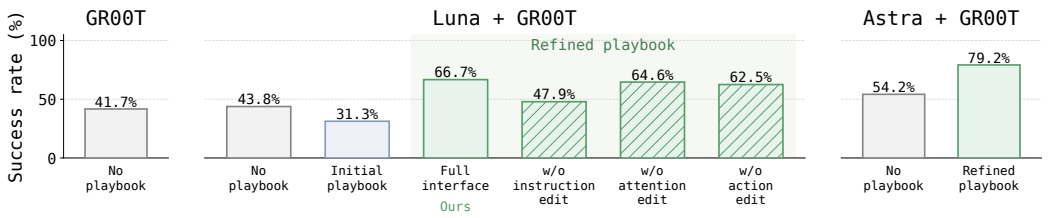


Figure 3: **Overall execution success and intervention ablations.** We compare agents without a playbook and with initial or refined playbooks, and examine the contribution of each intervention site for Luna + GR00T. Each ablation disables one site while retaining the other two and keeping the refined playbook fixed. The results suggest that the gains extend beyond direct action correction, with playbook-guided instruction editing making the largest contribution among the tested sites.

to handle variation in object placement across trials. We report success over 25 trials per task, for 75 trials in total. For real-world evaluation, Astra adapts the GR00T-derived playbook to the physical tasks and fine-tuned $\pi_{0.5}$ policy, and Luna uses the adapted guidance.

Agents. We use GPT-6 Astra as the strong agent and GPT-5.6 Luna as the light agent. Astra collects interaction experience with GR00T and revises the playbook using recorded observations, interventions, and outcomes from Luna's execution. Agents can replace the policy instruction with text of up to 512 characters, apply an attention-logit bias of strength 0–4 to visual tokens within a selected image region, and modify end-effector motion and gripper commands for up to five control steps per intervention. The intervention parameters are selected by the agent from the current observations. Later development rounds combine eight new teacher cases, two per task, with eight previously encountered cases for recipient evaluation. Revisions compare teacher and recipient execution on the same cases to clarify intervention conditions and outcome checks. Each revised playbook is saved as a separate version before subsequent execution. The initial checkpoint is distilled from Astra's first eight development rollouts with GR00T. Luna then executes development tasks using this playbook, and Astra revises the guidance using Luna's execution feedback. Subsequent checkpoints incorporate additional interaction experience and recipient feedback.

Evaluation Protocol. Playbook construction and refinement use development instances, while evaluation uses initial configurations excluded from that process. Each evaluated playbook is frozen across all evaluation episodes. Maximum episode lengths are 120 steps for spoon-on-towel, carrot-on-plate, and cube stacking, and 240 steps for eggplant-in-basket. Comparisons use the same evaluation configurations, reset seeds, and policy checkpoint. Agents without a playbook receive interface documentation. Simulator object states and success signals are excluded from agent inputs; success is determined by the environment against the original task goal, even when the agent modifies the policy instruction.

4.2 Main Results

SimplerEnv Bridge. The refined playbook substantially improves execution with the same agent, policy, and interface (Figure 3). GR00T alone achieves 41.7% success, and adding Luna without a playbook raises this to 43.8%. With the refined playbook, Luna reaches 66.7%, a gain of 22.9 percentage points over its unguided execution. The same playbook also benefits Astra, which reaches 79.2%. Experience accumulated through policy interaction therefore produces guidance that is useful across agents with different capabilities.

Real-World Manipulation. We evaluate the harness on a Franka Panda using $\pi_{0.5}$ for Cube-to-Tray, Cube Stacking, and Button Pressing (Table 1). Across the three tasks, the playbook improves Luna's overall success to 64.0%, compared with 37.3% for $\pi_{0.5}$ alone and 0% for Luna with the policy but no playbook. These results show that access to intervention tools alone does not ensure effective control of the physical robot. Without a playbook, Luna struggled to compensate for recurring spatial offsets between the intended interaction location and the robot's executed motion. Changes to instructions or attention did not reliably resolve these errors in the task-specific fine-tuned policy. The real-world

Table 1: **Real-world manipulation success rates.** We compare the frozen policy with Luna-guided execution, with and without the refined playbook, over 25 trials per task. Task entries report successful trials; Overall reports the aggregate success rate (%).

Method	Playbook	Cube-to-Tray	Cube Stacking	Button Pressing	Overall (%)
$\pi_{0.5}$	None	15/25	3/25	10/25	37.3
Luna + $\pi_{0.5}$	None	0/25	0/25	0/25	0
Luna + $\pi_{0.5}$	**Refined (Ours)**	18/25	13/25	17/25	**64.0**

Table 2: **Contribution of the teacher model.** Success on 48 held-out SimplerEnv Bridge instances using refined playbooks. All recipients operate frozen GR00T; teacher and recipient use separate contexts.

Teacher	Recipient	Overall (%)
Luna	Luna	22.9
Astra	**Luna**	**66.7**

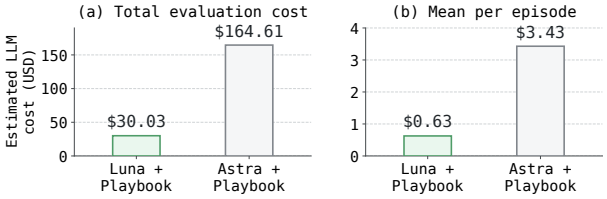


Figure 4: **Agent inference cost.** Total and per-episode API cost estimates for Luna and Astra operating GR00T with the refined playbook on the same test set.

playbook was adapted by Astra from the GR00T-derived playbook to the current physical tasks, providing Luna with task-specific guidance for intervening during execution.

4.3 Recursive Harness Distillation

Effect of Iterative Refinement. The initial playbook reduces Luna's success from 43.8% without a playbook to 31.3%, whereas the refined playbook raises it to 66.7%. Figure 5 tracks success rates and mean tokens per episode across the initial, intermediate, and refined playbooks, showing that improvements emerge at different stages across tasks. Refinement improves success across all four tasks while reducing mean tokens per episode in most tasks relative to the initial playbook. Overall, playbook refinement improves success without increasing average token usage per episode.

Role of a More Capable Teacher. To examine whether a Luna teacher can obtain the same gains through recursive refinement, we compare Luna and Astra as teachers while keeping Luna as the recipient operating the same frozen GR00T on SimplerEnv Bridge tasks (Table 2). In the Luna-to-Luna setting, the teacher and recipient operate in separate contexts. Both settings use the same development split, interface, playbook length limit, and limits on development rollouts and revision attempts. The Luna-derived refined playbook achieves only 22.9% success, compared with 66.7% for the Astra-derived counterpart, and falls below Luna without a playbook (43.8%). These results show that having the light agent construct and recursively refine its own playbook is insufficient to obtain the gains achieved with a strong teacher. This comparison supports the teacher–recipient hierarchy in RHD: a more capable teacher develops and refines guidance that the lighter recipient can reuse.

4.4 Contribution of Intervention Sites

We evaluate the contribution of instruction editing, attention guidance, and action correction by disabling one intervention site at a time while retaining the other two and keeping the refined playbook fixed (Figure 3). Removing instruction editing causes the largest performance drop among the tested sites. The results show that playbook-guided use of the policy instruction contributes to the gains alongside direct action correction.

5 Analysis

We examine what the playbook captures about the policy, how this guidance changes execution, and the inference cost of deploying the recipient.

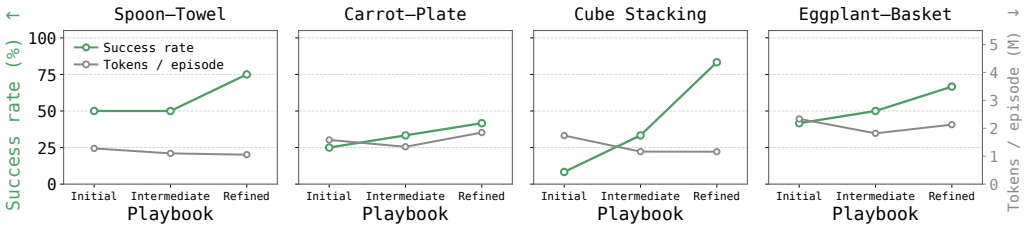


Figure 5: **Task success and deployment token use across playbook revisions.** Success rate (left axis) and mean agent token usage per episode (right axis) are evaluated on the same 12 held-out instances per task. Token usage is averaged over all episodes, including failures.

playbook.md

```
# 1. Re-identify the object after occlusion
After occlusion, re-identify the source before claiming grasp, transport or placement.
# 2. Distinguish a weak lift from an empty grasp
Do not reopen solely because an upward command failed to rise.
If geometry remains plausible, inspect one fresh coherent CLOSED lift prefix that can produce visible separation,
accounting for lag.
# 3. Check the policy response
Inspect the returned executable prefix: an OPEN instruction can still return CLOSE, and an alignment instruction can
still return empty transport.
# 4. Redirect attention toward the object
When learned recovery keeps withdrawing toward an empty destination, a simple source-directed grasp instruction and a
current-source attention region can produce a more useful OPEN approach than repeated complex directional prose.
# 5. Preserve an achieved correction
after a useful OPEN lateral/depth correction, inspect every following native prefix for erasing that correction.
# 6. Verify support before and after release
projected overlap alone does not establish lateral/depth support.
only later source/pad separation and stable support while the hand withdraws can establish independent placement.
```

Figure 6: **Operational insights from the playbook.** Selected excerpts address object identity after occlusion, weak versus empty lifts, instruction–action mismatches, attention-based recovery, preservation of corrections, and evidence for placement.

5.1 Understanding Policy Behavior through the Playbook

Figure 6 presents six examples from the playbook developed through interactions with GR00T on SimplerEnv Bridge tasks, illustrating how execution experience becomes practical intervention guidance.

Interpreting Observations. During development, Luna mistook a robot component for the spoon after occlusion, motivating explicit object re-identification using appearance and scene landmarks (entry 1). In eggplant manipulation, a weak initial lift was followed by a closed-gripper lift that confirmed possession. The playbook therefore recommends an additional bounded lift when the grasp remains plausible, rather than reopening solely because upward motion is small (entry 2).

Selecting Interventions. An OPEN instruction can still produce closing actions, so the playbook requires inspecting policy outputs and applying direct action corrections when needed (entry 3). When the policy moves toward the destination without the object, it recommends a simple grasp instruction with attention on the object's current location. In one cube-recovery sequence, this combination changed an upward proposal into an open-gripper descent; the proposed motion and gripper state still require inspection (entry 4).

Verifying Physical Outcomes. In a cube-stacking episode, executing the native prefix, the initial portion of a VLA-predicted action sequence, reversed a useful position correction. The playbook therefore requires checking that the corrected grasp geometry is preserved before closing the gripper (entry 5). Later, releasing the cube without confirmed support caused it to fall in front of the target. The playbook therefore requires checking object–target geometry before release and, when execution continues, stable support as the gripper withdraws (entry 6). These checks turn interaction experience into explicit conditions for selecting and verifying interventions.

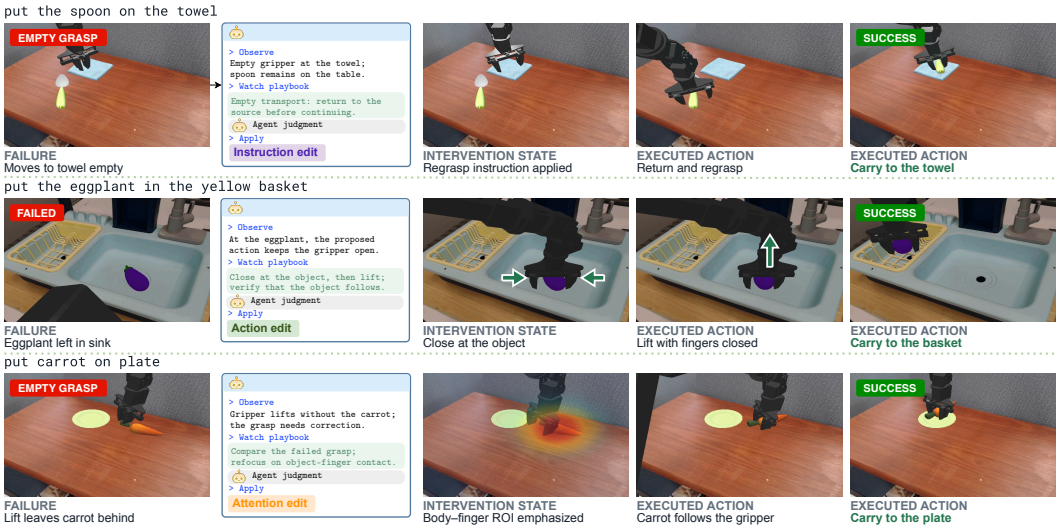


Figure 7: **Playbook-guided recovery in three Bridge tasks.** Each row connects an observed failure, a relevant playbook rule, an intervention, and subsequent execution. Multiple interventions can contribute to a recovery.

5.2 How Guidance Changes Execution

The examples show how an agent can return a stalled execution to a state from which GR00T can continue the task (Figure 7). For the spoon, the instruction edit directs the robot back to the source to regrasp before returning to the towel. For the eggplant, correcting the gripper action allows the robot to lift and carry the object. For the carrot, the spatial intervention focuses on object–finger contact before transport resumes. These cases illustrate how guidance about a local failure can support completion of the original task through the existing policy.

5.3 Deployment Cost

Agent inference cost is an important consideration for repeated robot deployment. We compare Luna and Astra operating frozen GR00T on the same held-out Bridge test set using the refined playbook. Costs are calculated from recorded token usage and model-specific API rates (Figure 4). Per million tokens, Luna is priced at $0.20 for standard input, $0.25 for cache writes, $0.02 for cache reads, and $1.20 for output; Astra is priced at $10 for uncached input, $1 for cached input, and $50 for output. We normalize the total cost by the number of evaluation episodes to obtain the mean cost per episode. Over the full test set, Luna incurs an estimated inference cost of $30.03, compared with $164.61 for Astra. The corresponding mean costs are $0.63 and $3.43 per episode, respectively, making Astra approximately 5.5 times more expensive. This cost gap motivates deploying a light agent that reuses guidance distilled from a strong agent through the playbook.

6 Conclusion

We presented Recursive Harness Distillation, a framework for accumulating a strong agent's experience with a VLA as reusable guidance for a light agent. The strong agent distills its interaction experience into a playbook and recursively refines it using the light agent's execution feedback, adapting the guidance to how the recipient applies it without updating model parameters. Experiments on SimplerEnv Bridge and three real-world manipulation tasks show that the resulting playbook improves the light agent's ability to operate the deployed policy in both simulation and physical environments. On Bridge, the light agent with the playbook outperforms the strong agent without one, while the same playbook also improves the strong agent's performance. These results demonstrate the feasibility of harness distillation for robotics: experience with a deployed policy can be accumulated, refined through execution, and reused across agents to improve manipulation.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing, assistance with experimental execution, and discussion of experimental results. GPT models also served as teacher and recipient agents within our experimental framework. We did not use generative AI tools for related-literature discovery or synthetic dataset generation. Additionally, we used these tools to revise manuscript passages. The authors reviewed the AI-assisted text, code, and figures for consistency with the experimental procedures and results. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 describes the intervention interface and the procedure for constructing and recursively refining the playbook. Section 4 specifies the policy checkpoints, agent models, simulation and physical robot setups, development and evaluation splits, and evaluation protocol. The simulation comparisons use the same held-out initial configurations, reset seeds, and policy checkpoint, with playbooks fixed before evaluation. Appendix B provides development examples illustrating playbook revision and application, and Appendix C summarizes the agent inputs, available interventions, and execution feedback. The supplementary material includes the complete refined playbook used by Luna with frozen GR00T in the main SimplerEnv Bridge evaluation. Section 5.3 details the calculation of deployment costs from recorded token usage and API rates.

REFERENCES

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

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

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.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Siyi Chen, Hugo Hadfield, Alex Zook, Mikaela Angelina Uy, Chan Hee Song, Erwin Coumans, Xuning Yang, Faisal Ladhak, Qing Qu, Stan Birchfield, et al. Volo: A physical orchestrator for open-vocabulary long-horizon manipulation. *arXiv preprint arXiv:2606.07723*, 2026.

Yujia Chen, Yang Ye, Zhongqi Li, Yuchi Ma, and Cuiyun Gao. Smaller but better: Self-paced knowledge distillation for lightweight yet effective lcms. *Proceedings of the ACM on Software Engineering*, 2(FSE):3057–3080, 2025.

Liane Galanti, Dhruv Shah, and Tri Dao. Addressing the orchestration gap in generalist robots via physical agency. *arXiv preprint arXiv:2607.21725*, 2026.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the association for computational linguistics: ACL 2023*, pp. 8003–8017, 2023.

Jiaheng Hu, Mohit Shridhar, Caden Lu, Dhruv Shah, Hao-Tien Lewis Chiang, Jie Tan, and Annie Xie. What matters in orchestrating robot policies: A systematic study of hierarchical vla agents. *arXiv preprint arXiv:2606.10267*, 2026.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *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.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert 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 Rich Walke, Anna Walling, Hao-Huan Wang, Li-Li Yu, and Ury Zhilinsky. π0.5: a vision-language-action model with open-world generalization. *ArXiv*, abs/2504.16054, 2025. URL `https://api.semanticscholar.org/CorpusID:277993634`.

Hyun Joe Jeong, Gokul Swamy, and Andrea Bajcsy. Learning what to say to your vla: Mostly harmless vision language action model steering. *arXiv preprint arXiv:2606.12299*, 2026.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the nineteenth international conference on machine learning*, pp. 267–274, 2002.

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. *arXiv preprint arXiv:2406.09246*, 2024.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 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 *2023 IEEE International conference on robotics and automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

Shuo Liu, Ishneet Sukhvinder Singh, Yiqing Xu, Jiafei Duan, and Ranjay Krishna. Vls: Steering pretrained robot policies via vision-language models. *arXiv preprint arXiv:2602.03973*, 2026.

Yitao Liu, Tianxiang Sun, Xipeng Qiu, and Xuanjing Huang. Learning to teach with student feedback. *arXiv preprint arXiv:2109.04641*, 2021.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. Pmlr, 2015.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on robot learning*, pp. 1723–1736. PMLR, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Wangchunshu Zhou, Canwen Xu, and Julian McAuley. Bert learns to teach: Knowledge distillation with meta learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7037–7049, 2022.

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 Performance Difference under Playbook Revision

Fix the task distribution and robot-step horizon H. As in Section 3, t indexes agent–interface decisions, and $N(\tau)$ denotes the number of decisions before episode termination. Assume that $N(\tau)$ has finite expectation under the policies considered.

Let h_t be the full observable history, including past decision records and executed actions, and include the terminal task outcome in $h_{N(\tau)}$. Conditional on a history and decision record, the next-history distribution is the same for all playbooks. All policies share the same initial task distribution.

For the incumbent policy Π_{S,P_k}, define $V_k(h_t)$ at nonterminal histories as the probability of eventual success when continuing with that policy, and set $V_k(h_{N(\tau)}) = Y(\tau)$ at termination. At histories not reached by the incumbent, the value is defined by continuing with the incumbent from that history under the same environment model. Then

$$Q_k(h_t, b_t) = \mathbb{E}[V_k(h_{t+1}) \mid h_t, b_t]\,. \tag{6}$$

Let $\mathbb{E}_{P'}$ denote expectation over trajectories induced by the candidate policy $\Pi_{S,P'}$. Using $A_k(h_t, b_t) = Q_k(h_t, b_t) - V_k(h_t)$, conditional expectation and telescoping give

$$\begin{aligned}
&\mathbb{E}_{P'}\left[\sum_{t=0}^{N(\tau)-1} A_k(h_t, b_t)\right] \\
&= \mathbb{E}_{P'}\left[\sum_{t=0}^{N(\tau)-1} \big(V_k(h_{t+1}) - V_k(h_t)\big)\right] \\
&= \mathbb{E}_{P'}\big[V_k(h_{N(\tau)}) - V_k(h_0)\big] \\
&= \mathbb{E}_{P'}[Y(\tau)] - \mathbb{E}_{h_0}[V_k(h_0)] \\
&= J_S(P') - J_S(P_k).
\end{aligned} \tag{7}$$

This is the episodic history-based form of the standard performance-difference identity (Schulman et al., 2015). It applies to the executed robot policy induced by the entire interface, including instruction, attention, and output interventions.

B Playbook Refinement Examples

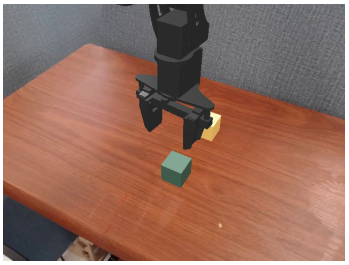
We examine two development episodes with Luna operating frozen GR00T on SimplerEnv Bridge. The first connects an execution error to a documented playbook revision; the second shows how recorded guidance is applied through the action interface. The examples come from different development checkpoints.

B.1 Preserving a position correction

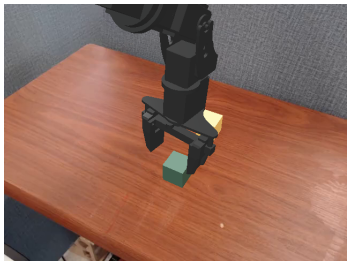
In cube stacking, a useful position correction was undone by subsequent policy motion. Between steps 34 and 36, the measured end-effector displacement along X was +8.7 mm; between steps 36 and 40, it was -11.7 mm. The teacher review identified this reversal. The revised playbook therefore requires checking that later policy actions retain the achieved correction, rather than judging progress from the latest command alone.



Step 34: before correction



Step 36: after correction



Step 40: subsequent policy motion

> PRESERVE THE CORRECTION: after a useful OPEN lateral/depth correction, inspect every following native prefix for erasing that correction.

Here, a native prefix is the initial portion of the policy-predicted action sequence selected for execution. This example shows how recipient feedback becomes a concrete condition for committing later policy actions.

B.2 Separating closure from lifting

In an eggplant episode, Luna replaced an OPEN descent proposal with one stationary CLOSE action at step 15. It then requested two CLOSED upward actions of 12 mm each. The measured rise over that segment was only 6.3 mm, so the command itself did not establish a grasp. After a further CLOSED lift, the object moved with the fingers and vacated its support location. The rollout reached environment success at step 24.



Step 15: close at the source



Step 20: verify source departure



Step 24: carry toward the basket

> If the aperture is still changing and the source remains supported, first use a short low-motion CLOSED settling prefix, or a bounded stationary CLOSED edit when the native rows immediately lift/sweep away. Then observe the source/pads before lifting.

The guidance separates closure, attempted lifting, and observed object motion. It supports a bounded correction followed by an outcome check. The final environment success does not establish independent object support after release.

C Intervention Interface

Table 3 summarizes the inputs and interventions available to the agent operating GR00T on SimplerEnv Bridge.

Table 3: **Intervention interface.**

Component	Specification
Agent inputs	Current RGB observation, robot proprioception, task instruction, playbook, recent execution history, candidate actions, and runtime feedback.
Instruction editing	Replace the policy instruction with text of up to 512 characters for the requested inference call. The original task remains the evaluation objective.
Attention editing	Specify a normalized image rectangle and an additive attention-logit bias of strength 0–4 for the corresponding visual tokens in the action head's visual cross-attention layers.
Action editing	Replace executable actions with up to five control steps of end-effector translation, rotation, and gripper commands. Translation is expressed in metres and rotation in radians.
Candidate review and execution	Preview candidate actions without advancing the simulator, then execute a selected prefix of 1–4 policy actions or 1–5 edited actions.
Execution feedback	Return updated observations and execution records after the selected actions. Commanded actions and measured robot motion are recorded separately.
Information restrictions	Simulator object poses, rewards, and success labels are excluded from agent inputs. The policy parameters remain frozen.