Recova: Agent-Guided Failure Recovery for Autonomous Robotic Manipulation

Anonymous authors
Paper under double-blind review

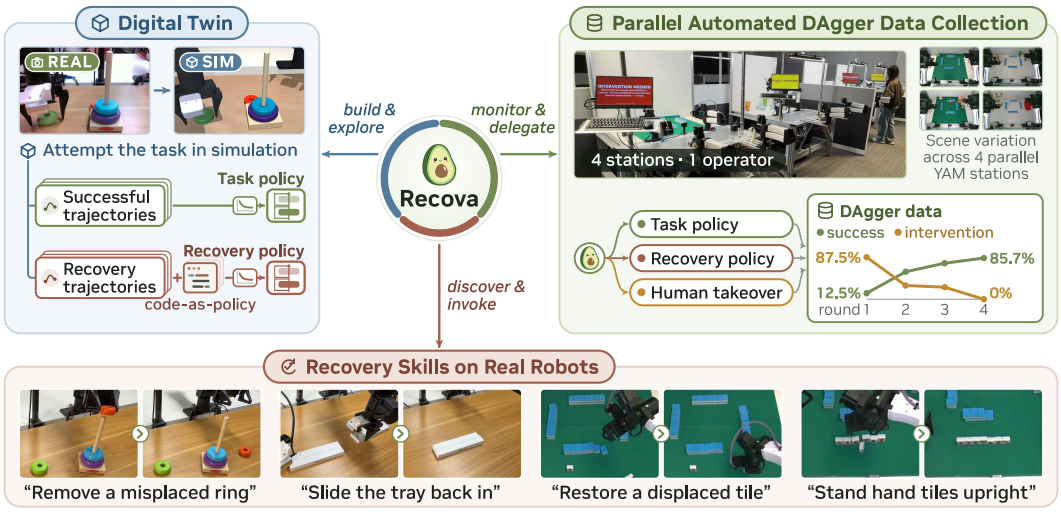


Figure 1: **Overview of Recova.** *Digital twin (top left):* A coding agent explores task execution in a reconstructed workstation and develops recovery skills from failures. Task and recovery trajectories train separate policies, while recovery programs form a reusable skill library. *Parallel DAgger (top right):* Across four real-robot stations, the agent coordinates execution and verified recovery, requests human help when needed, and uses the resulting experience to improve both policies. Success rises and human intervention falls over training rounds (Fig. 4). *Recovery skills (bottom):* Paired frames show learned recoveries executed on real robots. Anonymous project page: https://recova-iclr.github.io/

Abstract

Manipulation failures can leave scenes in states from which a task policy cannot recover. Learning corrective behaviors requires scalable failure exploration and physical grounding. We present Recova, an agent-guided framework that jointly develops task execution and recovery in a reconstructed digital twin, then verifies and refines both through real-world experience. In the twin, the agent diagnoses failures, tests corrective programs, and collects successful task and recovery rollouts for separate policies. During deployment, it monitors progress, invokes a learned or programmatic recovery, verifies scene restoration, and resumes execution. When no suitable recovery is available, a human demonstration resolves the failure and enters the learning loop, allowing the system to expand its recovery capabilities. Physical rollouts and human demonstrations are routed to the corresponding policy for DAgger training. Across six LIBERO-Pro settings and four MolmoSpaces categories, Recova achieves 78.8% and 64.9% mean success, compared with 71.7% and 38.0% for the strongest baselines. With parallel collection across four real-robot workstations, DAgger fine-tuning raises mean success from 23.8% to 77.5%, and recovery skills further raise it to 87.5%. Over four collection rounds on one task, observed human intervention falls from 87.5% to 0%. Together, these results show how agent-guided recovery turns failures into reusable capabilities, improving robustness while progressively reducing human intervention.

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.

1 INTRODUCTION

Vision-language-action (VLA) models enable robots to perform diverse manipulation tasks from language instructions (Brohan et al., 2023a; Kim et al., 2024; NVIDIA et al., 2025), from folding laundry and assembling boxes to cleaning unfamiliar homes (Black et al., 2025b;a). However, even a small mistake can halt an otherwise capable policy. A tipped cup must be set upright before pouring can continue; a pencil caught across a box rim must be moved before the lid can close. In both cases, the robot must restore a workable scene before resuming the task.

Most VLA training data, however, contain successful trajectories from standard initial states and provide little guidance for restoring a failed scene. Failures create a long tail of unusual configurations, and correcting them may require undoing a previous action, repositioning an object, or temporarily moving away from the nominal task goal. Consequently, policies may struggle when objects appear in unfamiliar arrangements (Zhou et al., 2025b) or when asked to continue after a failed attempt (Liu et al., 2026b). Simply retrying the task policy often leaves the underlying obstruction unresolved. Robots therefore need a complementary recovery capability that can expand as new failure modes arise.

Learning such recovery behaviors directly on hardware is costly, as exploring failures requires frequent resets and risks damaging objects (Torne et al., 2024; Li et al., 2026b). Reconstructed digital twins provide a safer and more scalable setting for reproducing failures and testing corrections (Torne et al., 2024), while physical experience remains essential for verifying and refining them under real contacts and observations. Effective recovery therefore requires a loop that discovers behaviors in simulation, validates them on the robot, and feeds physical experience back into learning.

Language-model agents are natural candidates to coordinate this loop. Recent models combine multimodal understanding with the ability to use tools, execute code, and revise solutions from feedback (Anthropic, 2026; OpenAI, 2026). Prior robotics systems show that agents can interpret failures and develop reusable programs (Liu et al., 2023d; Liang et al., 2023; Lu et al., 2026). The central challenge is to connect these abilities into a complete recovery process: diagnose a failure, select or develop a correction, verify that it worked, retain successful solutions, and use the resulting data to improve the appropriate policy. The agent must also decide whether new experience should become a reusable program, a task-policy demonstration, or a recovery-policy demonstration. When no suitable recovery is available, human demonstrations enter the loop to resolve the failure and expand future recovery capabilities (Liu et al., 2023c).

We present Recova (Fig. 1), an agent-guided framework that develops task execution and scene recovery in a reconstructed digital twin, then continually improves both through real-world experience. In the twin, the agent explores task execution, diagnoses failures, and develops corrective behaviors. Successful task and recovery rollouts initialize separate task and recovery policies, while programs that compose perception and motion primitives capture reusable recovery strategies. Keeping these data pathways separate allows the task and recovery policies to specialize in their distinct roles.

During real-world deployment, the agent monitors task progress, invokes an applicable learned or programmatic recovery, verifies that the scene is workable again, and resumes execution. When no suitable recovery succeeds, it requests human assistance. Successful task rollouts and human task demonstrations improve the task policy, while human recovery demonstrations improve the recovery policy. A parallel collection harness runs this loop across four robot workstations, allowing task and recovery experience to accumulate concurrently. Newly encountered failure modes also guide the development of additional skills. Human intervention therefore resolves the current failure while helping the robot recover more autonomously in the future.

Our contributions are:

- An agent-guided real-to-sim-to-real framework for jointly developing task execution and failure recovery. It uses a reconstructed digital twin to safely discover corrective behaviors, then verifies and refines them through physical robot experience.
- A parallel deployment-and-learning loop that coordinates task and recovery policies, reusable recovery programs, outcome verification, and human demonstrations. Across four concurrently operating robot workstations, collected experience improves mean task suc-

cess by 53.7 percentage points to 77.5%. On one task, the observed human-intervention rate falls from 87.5% in the first collection round to 0% in the fourth.

- A comprehensive evaluation on two simulation benchmarks and four real-robot tasks. Recova exceeds the strongest baseline in our comparison by 7.1 percentage points on LIBERO-Pro and 26.9 points on MolmoSpaces. Recovery skills provide a further 10-point gain on the real-robot tasks, reaching 87.5% mean success.

2 Related Work

Robot foundation models and agents. Expressive imitation-learning policies (Chi et al., 2023; Zhao et al., 2023) and large, diverse datasets, including cross-embodiment robot data and egocentric human video (Brohan et al., 2023b; Open X-Embodiment Collaboration et al., 2024; Cai et al., 2025), have produced generalist manipulation policies, including vision-language-action (VLA) models that map images and language instructions to actions (Brohan et al., 2023a; Octo Model Team et al., 2024; Kim et al., 2024; Black et al., 2025b; Pertsch et al., 2025; Black et al., 2025a; NVIDIA et al., 2025; Fang et al., 2026). Language models ground plans in robot affordances (Ahn et al., 2022), replan from closed-loop feedback (Huang et al., 2022), write plans, policy code, value maps, and rewards (Singh et al., 2023; Liang et al., 2023; Huang et al., 2023; Ma et al., 2024), and accumulate reusable skills (Wang et al., 2024a). A task-management vision-language model can also direct a VLA over long horizons by replanning subtasks and visual traces (Liu et al., 2026a). For manipulation, CaP-X benchmarks coding agents (Fu et al., 2026), RATs learns skills through self-directed play (Zhang et al., 2026), and ASPIRE refines robot programs, accumulates validated skills, and transfers simulation experience to real-robot programming (Lu et al., 2026). Recova coordinates learned and programmed recoveries around a task policy, with human demonstrations for refinement.

Failure detection and recovery. Prior work explains manipulation failures from summarized experience (Liu et al., 2023d), detects and analyzes them with fine-tuned vision-language models (Duan et al., 2025; Ye et al., 2025), monitors execution against constraints written by a language or vision-language model (Guo et al., 2024; Zhou et al., 2025a), or predicts them from policy features, uncertainty, or learned dynamics (Gu et al., 2025; Xu et al., 2025; Liu et al., 2024a). Runtime steering filters a policy's proposed actions with a vision-language model before they execute (Wu et al., 2025), and neuro-symbolic methods combine failure detection with recovery planning (Cornelio & Diab, 2024). Separate recovery controllers have substantial precedent. Recovery RL separates task reward from constraint satisfaction (Thananjeyan et al., 2021); RecoveryChaining learns to return to states from which nominal controllers can complete the task and demonstrates sim-to-real transfer (Vats et al., 2025). FARL keeps its offline recovery policy fixed during online task-policy learning (Li et al., 2026b). RACER guides task execution and recovery through a shared visuomotor actor (Dai et al., 2025), while FailSafe generates and verifies corrective-action data for a VLM that assists VLAs (Lin et al., 2026). ReSYNC expands recovery skills and planning abstractions from failures (Li et al., 2026a). Recova connects recovery development in a reconstructed scene with visual restoration checks, learned and programmed executors, and human demonstrations for separate policy updates during deployment. Benchmarks evaluate perturbation robustness (Li et al., 2024; Liu et al., 2023a; Zhou et al., 2025b; Kim et al., 2026) and recovery from failed executions (Liu et al., 2026b); our protocol also tracks human intervention alongside task and recovery outcomes.

Learning from deployment and simulation. DAgger aggregates expert labels on the states a learner visits (Ross et al., 2011). Human-gated variants let an operator take over (Kelly et al., 2019; Mandlekar et al., 2020), and robot-gated variants request help based on uncertainty, novelty, risk, or predicted action discrepancy, including across robot fleets (Menda et al., 2019; Hoque et al., 2021b;a; 2022). Deployed robots also improve from human interventions (Liu et al., 2023c; 2024b), language corrections (Shi et al., 2024; Liu et al., 2023b), human-in-the-loop reinforcement learning (Luo et al., 2025), and autonomously collected experience (Bousmalis et al., 2024; Zhou et al., 2024; Ghasemipour et al., 2025). ENPIRE lets agents manage scene resets, rollouts, verification, and policy improvement on real robots (Xiao et al., 2026). Before deployment, real-to-sim pipelines build digital twins or cousins of a scene (Torne et al., 2024; Dai et al., 2024) and reconstruct articulated assets (Chen et al., 2024; Mandi et al., 2025), while generative pipelines propose new simulation tasks with demonstrations (Wang et al., 2024b;c) or synthesize demonstrations from a few human examples (Mandlekar et al., 2023). Recova combines both: simulation initializes task and recovery behaviors, and deployment routes task successes, task demonstrations, and recovery demonstrations

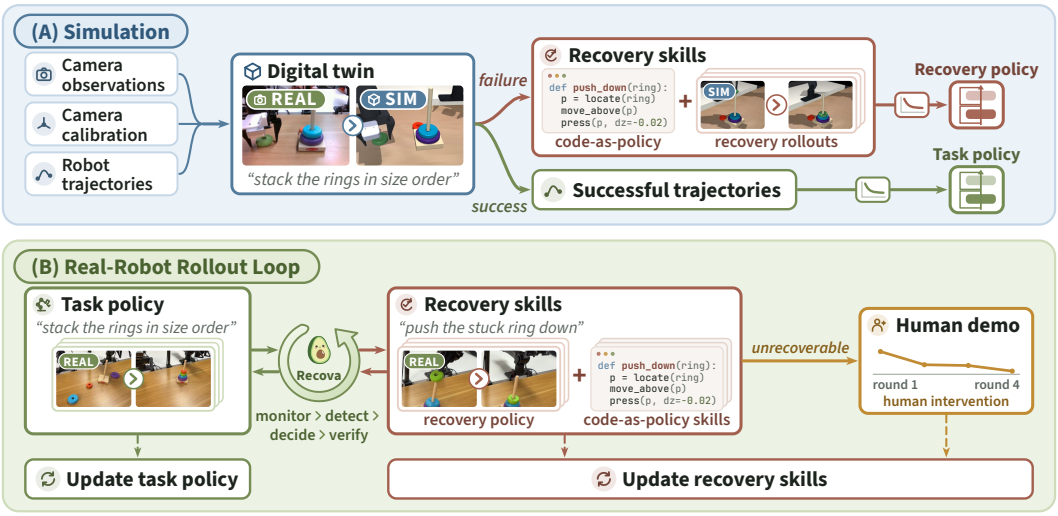


Figure 2: **(A) Simulation:** Recova builds a digital twin from camera observations, calibration, and robot trajectories. Successful task and recovery rollouts train separate policies, while recovery programs populate a code-as-policy skill library. **(B) Real-robot rollout:** Recova monitors execution, invokes recovery policies or programs, verifies restoration, and requests human demonstrations when needed. Task successes, verified recoveries, and human demonstrations update the corresponding policies, while new failures guide skill-library expansion. Paired frames show rollout endpoints; the inset tracks human intervention across DAgger rounds.

to the policy each one teaches, requesting human input only at failures rather than expert labels at every visited state.

3 METHOD

Recova decouples task execution from scene recovery and connects the two through an agent that coordinates simulation, deployment, and learning (Fig. 2). This separation trains each capability on the data suited to its role and lets recovery skills accumulate independently of any single task. Sec. 3.1 formalizes task execution and recovery. Sec. 3.2 describes how a coding agent builds a digital twin of the workstation and turns simulated successes and failures into training data and reusable recovery programs. Sec. 3.3 presents the real-robot loop, in which a monitor tracks the task policy, triggers and verifies recoveries, and involves a human operator only when autonomous execution falls short. Sec. 3.4 describes how each deployment round refines the task and recovery policies using their respective experience, while newly encountered failures guide the development of additional recovery skills.

3.1 TASK EXECUTION AND SCENE RECOVERY

Let g be a language instruction for a manipulation task and o_t the camera observations and robot state at time t. A task policy $\pi_\theta(a_t \mid o_t, g)$ acts until the scene satisfies a completion condition C_g. The task can often still be completed after a failure, but the task policy may be unable to continue until the scene is corrected. For example, when stacking rings on a peg in size order, the robot may release a ring at an angle, causing it to catch on the peg's tip rather than slide into place. Recova treats recovery as a distinct objective that targets the failure blocking task progress. A recovery may restore a workable scene for the task policy to resume, or resolve the remaining task through a corrective action.

A recovery skill is specified by a natural-language recovery instruction z, such as "push the stuck ring down," and is executed either by a language-conditioned recovery policy $\rho_\phi(a_t \mid o_t, z)$ or by a code-as-policy program from a skill library that composes scene queries and motion primitives (Liang et al., 2023). The recovery skill set $\mathcal{K}$ contains the recovery instructions known for a task. Conditioning recovery on z rather than on g decouples recovery skills from individual tasks, so a

skill can be reused wherever the same failure recurs. When neither the task policy nor a recovery skill can make progress, a human operator provides a demonstration, which becomes training data (Sec. 3.4).

Recovery skills serve three complementary roles, and a single skill may serve several. *Restorative* skills return the scene to a workable state, such as uprighting a fallen object. *Enabling* skills prepare the next step, such as moving an object within reach. *Adaptive* skills change the approach rather than repeating a failed attempt, such as re-grasping with a new grasp pose or opening an oven from another direction (Fig. 3).

3.2 Simulation in a Digital Twin

Agent-built digital twin. Exploring failures on hardware requires frequent human resets and risks damaging objects. Recova instead explores each task and its failure modes in a digital twin built by a coding agent, a language model that writes, executes, and revises code from execution feedback. The agent reconstructs the workstation from real-robot recordings: camera observations, robot trajectories, and camera calibration when available. In our implementation, the twin is a MuJoCo scene (Todorov et al., 2012). Starting from models of the robot and the task objects, the agent fits object poses and contact parameters so that replaying a recorded trajectory in the twin aligns with the real camera views. This grounds the twin in the real workstation's scale and layout; Fig. 2(A) pairs a real frame with the twin's render of the same moment.

Discovering task and recovery behaviors. Given g, the agent attempts the task in the twin, and successful attempts form the task dataset. When an attempt fails, the agent diagnoses the failure, names a recovery instruction z, and develops the corresponding skill in two complementary forms: a program for the skill library, and recovery rollouts that pair the disturbed scene and z with corrective actions, which form the recovery dataset. These datasets provide initial training data for π_θ and ρ_ϕ, and the new instructions seed $\mathcal{K}$, so the robot enters deployment with recovery skills already in place. Recova queries a VLM agent to assess the current scene and verify whether the recovery policy has restored it to a state from which the task policy can resume (Sec. 3.3).

3.3 Real-Robot Rollout Loop

Simulation gives both policies a strong starting point, but the real workstation still presents contact dynamics, sensing conditions, and failure modes that the twin cannot fully anticipate. The real-robot rollout loop closes this gap: the task policy runs autonomously, Recova intervenes only when needed, and the resulting experience becomes training data that further improves both policies (Sec. 3.4). A vision-language monitor implements the four steps of Fig. 2(B), and Alg. 1 in Sec. A gives the complete control flow.

- **Monitor.** Before the first attempt, a requirement query converts g and the initial images into an explicit completion condition C_g. While the task policy acts, a completion query periodically checks whether C_g holds.
- **Detect.** At a finer interval, an intervention query compares recent frames with the initial scene to decide whether to intervene. It also reports whether the scene is still intact and names a recovery instruction z, reusing an existing instruction from $\mathcal{K}$ whenever one applies so that demonstrations of the same behavior share a label.
- **Decide.** If the scene is intact, the task policy has stalled without disturbing it, and the operator provides a task demonstration. Otherwise, the recovery policy executes z within a fixed time budget if z is in its registry of trained instructions. If no learned skill applies or the attempt fails, the operator demonstrates the recovery, which is recorded under z. After two consecutive interventions on a task, the next attempt is a full human demonstration, which avoids cycles of repeated failure.
- **Verify.** After an autonomous recovery, a restoration query uses the initial images as a reference to check whether the scene is ready for another task attempt. Objects do not need to return to their exact original positions and orientations. Once recovery is verified, the robot returns to its home pose and the task policy starts a new attempt.

Because human assistance is needed only at these handoffs, one operator can supervise multiple stations with different scene configurations in parallel (Fig. 4). This scales data collection without

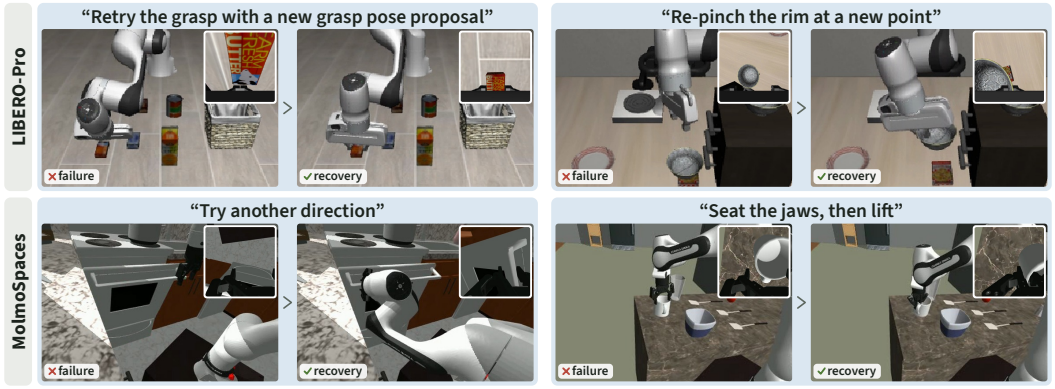


Figure 3: **Recovery skills in simulation.** The failure (left) and recovery (right) moments of one LIBERO-Pro (top) or MolmoSpaces (bottom) episode: the main camera view, with the wrist camera inset at the top right, under the recovery instruction.

Table 1: **MolmoSpaces task success (%).** P&P denotes pick-and-place. Avg. is the unweighted mean across the four simulation task categories. Bold marks the best reported value in each column. Baseline rows reproduce published results.

	MolmoSpaces ↑				
Methods	Pick	P&P	Open	Close	**Avg.**
Agentic code-as-policy methods					
CaP-Agent0 (Fu et al., 2026)	23.0	11.0	14.0	36.0	21.0
RATs (Zhang et al., 2026)	37.0	22.0	20.0	73.0	38.0
Vision-language-action policies					
π_0 (Black et al., 2025b)	14.0	7.0	10.0	47.0	19.5
π_0-FAST (Pertsch et al., 2025)	20.0	10.0	9.0	35.0	18.5
$\pi_{0.5}$ (Black et al., 2025a)	28.0	10.0	18.0	62.0	29.5
MolmoAct2-DROID (Fang et al., 2026)	41.0	32.0	3.8	67.0	36.0
Recova (ours)	**78.7**	**40.5**	**56.5**	**83.9**	**64.9**

requiring additional operators. On real robots, the learned recovery policy performs autonomous recoveries. Meanwhile, the agent develops and tests new recovery programs in the digital twin, expanding the skill library (Sec. 3.2).

3.4 Learning Across DAgger Rounds

Recova refines the task and recovery policies through repeated rounds of DAgger-style data collection (Ross et al., 2011). During round k, both policies remain fixed while the agent collects trajectories from robot execution and human demonstrations. These trajectories are divided into four groups. $\mathcal{S}_k$ (`Success`) contains task rollouts completed without human takeover, and $\mathcal{E}_k$ (`Expand`) contains human task demonstrations. $\mathcal{R}_k$ (`Recover`) contains human recovery demonstrations, each labeled with its recovery instruction z, while $\mathcal{A}_k$ contains autonomous recoveries that pass the scene-restoration check. Failed policy segments are saved for analysis but excluded from training.

At the end of each round, Recova adds successful task rollouts and human task demonstrations to the task dataset, and human recovery demonstrations and verified autonomous recoveries to the recovery dataset:

$$\mathcal{D}_{\text{task}}^{k+1} = \mathcal{D}_{\text{task}}^{k} \cup \mathcal{S}_k \cup \mathcal{E}_k, \qquad \mathcal{D}_{\text{rec}}^{k+1} = \mathcal{D}_{\text{rec}}^{k} \cup \mathcal{R}_k \cup \mathcal{A}_k. \tag{1}$$

Each policy is then fine-tuned on its updated dataset using its standard supervised objective, with the task policy conditioned on the task instruction g and the recovery policy on the recovery instruction z. Keeping these datasets separate allows each policy to learn the behavior it is responsible for: completing the task or restoring a scene from which task execution can resume.

Table 2: **LIBERO-Pro task success (%).** Object, Goal, and Spatial each include initial-position swaps (Pos.) and task perturbations (Task). Avg. is the unweighted mean of all six columns. Bold marks the best reported value in each column. Baseline rows reproduce published results.

	Object ↑		Goal ↑		Spatial ↑		
Methods	Pos.	Task	Pos.	Task	Pos.	Task	**Avg.** ↑
Agentic code-as-policy methods							
CaP-Agent0 (Fu et al., 2026)	27.0	31.0	29.0	16.0	13.0	23.0	23.2
RATs (Zhang et al., 2026)	61.0	63.0	43.0	36.0	29.0	31.0	43.8
ASPIRE (Lu et al., 2026)	**98.0**	**95.0**	81.0	45.0	51.0	60.0	71.7
Vision-language-action policies							
π_0 (Black et al., 2025b)	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$\pi_{0.5}$ (Black et al., 2025a)	17.0	1.0	38.0	0.0	20.0	1.0	12.8
MolmoAct2-LIBERO (Fang et al., 2026)	52.0	24.0	20.0	21.0	27.0	57.0	33.5
Recova (ours)	91.2	87.2	**82.2**	**67.0**	**62.2**	**83.0**	**78.8**

The recovery skill set also expands as the robot encounters new failures. When no known recovery instruction applies, the intervention query proposes a new instruction, and a human demonstrates the corresponding recovery. This demonstration provides the first training example for the new skill. The instruction is added to $\mathcal{K}$ and guides the development of recovery programs in the digital twin (Sec. 3.2). After the recovery policy is trained on the new demonstrations, the instruction is added to the recovery policy's registry, making the learned skill available for autonomous execution. In this way, human interventions provide training examples that can help the robot handle similar failures autonomously in later rounds.

4 Experiments

We evaluate Recova in simulation and on real robots around three questions. (1) Does Recova raise task success under perturbations over code-as-policy agents and VLA policies? (2) Which recovery skills does it develop and execute? (3) Does deployment with Recova reduce human assistance over successive rounds? Sec. 4.1 addresses the first question on LIBERO-Pro and MolmoSpaces. Sec. 4.2 evaluates task and recovery behavior on four real-robot tasks and examines human intervention during data collection.

4.1 Simulation Benchmarks and Comparisons

Benchmarks and baselines. We evaluate on two simulation benchmarks that test manipulation beyond nominal configurations. Our LIBERO-Pro evaluation (Zhou et al., 2025b) covers the Object, Goal, and Spatial suites under initial-position swaps and task perturbations, yielding six settings. Position swaps exchange object locations, while task perturbations change the instruction to target another object. These settings test adaptation to changes in both scene layout and task requirements. Following prior work (Lu et al., 2026), we evaluate each task on 50 initial states. MolmoSpaces (Kim et al., 2026) spans four manipulation categories across diverse simulated scenes: pick, pick-and-place, open, and close. These categories cover rigid-object grasping and placement as well as articulated-object manipulation, such as opening and closing cabinets or drawers.

We compare against *code-as-policy (CaP) methods*, which use language models to generate executable robot-control programs, and *vision-language-action (VLA) policies*, which map visual observations and language instructions to robot actions. CaP baselines include CaP-Agent0 (Fu et al., 2026), RATs (Zhang et al., 2026), and, on LIBERO-Pro, ASPIRE (Lu et al., 2026). CaP-Agent0 uses multi-turn execution feedback to refine control programs, RATs acquires reusable code skills through self-directed play, and ASPIRE expands a skill library through iterative program refinement and exploration. VLA baselines include π_0 (Black et al., 2025b), $\pi_{0.5}$ (Black et al., 2025a), the MolmoAct2-LIBERO and MolmoAct2-DROID checkpoints (Fang et al., 2026), and, on MolmoSpaces, π_0-FAST (Pertsch et al., 2025). Baseline results are taken from prior publications, while Recova results are obtained from our own evaluation.

Benchmark implementation. In the simulation benchmarks, Recova pairs a MolmoAct2 task policy (Fang et al., 2026) with programmatic recovery skills. A recovery program takes over from

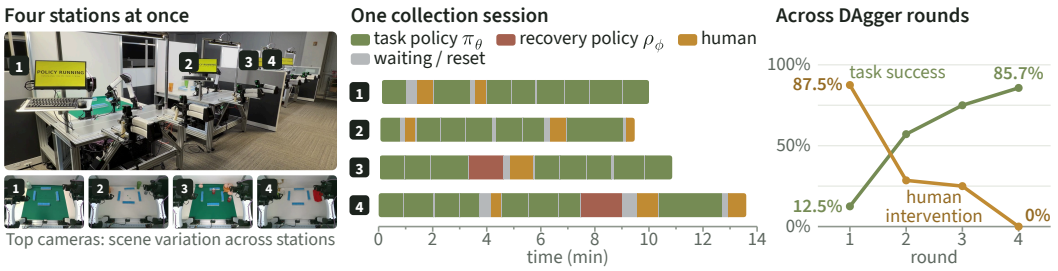


Figure 4: **Parallel automated DAgger data collection. Left:** Four robot stations operate in parallel with varied scene configurations. **Middle:** Station activity during one collection session, distinguishing task-policy execution, recovery-policy execution, human control, and waiting or reset periods. **Right:** Observed task success and human intervention rates across four collection rounds. As policies are fine-tuned on accumulated data, task success increases while human intervention decreases.

Table 3: **Real-robot task success (%)** over 20 trials per task and configuration. Rows add DAgger fine-tuning of the task policy, then recovery skills at deployment. Bold marks the best result for each task.

Policy	Pencil box	Stack rings	Draw tile	Discard tile	**Avg.** ↑
Base task policy	20.0	35.0	25.0	15.0	23.8
+ DAgger	70.0	75.0	85.0	80.0	77.5
+ Recovery skills	**85.0**	**90.0**	**90.0**	**85.0**	**87.5**

the state left by the policy when its execution budget is exhausted or, in LIBERO-Pro, progress stalls. Program development and evaluation use disjoint random seeds, with held-out initial object configurations at evaluation. Task success is the fraction of evaluation trials satisfying the task's completion criterion, including cases where the recovery program completes the task.

LIBERO-Pro results. Recova achieves the highest reported mean success rate of 78.8% across six settings, exceeding ASPIRE, the strongest baseline on average, by 7.1 percentage points (Tab. 2). The largest gains occur under task perturbations in the Goal and Spatial suites, where Recova outperforms ASPIRE by 22.0 and 23.0 percentage points, respectively. Although ASPIRE remains stronger on the Object suite, Recova leads in all four Goal and Spatial settings, suggesting particular strength in handling the task variations evaluated in these suites.

MolmoSpaces results. Recova achieves the highest reported success rate in all four MolmoSpaces categories, with a mean of 64.9%, exceeding RATs, the strongest baseline on average, by 26.9 percentage points (Tab. 1). Consistent gains across picking, pick-and-place, opening, and closing suggest that the framework's benefits extend across diverse manipulation behaviors. Pick-and-place remains the most challenging category, but Recova reaches 40.5% success, compared with 32.0% for MolmoAct2-DROID, the strongest baseline in this category. Qualitative examples in Fig. 3 illustrate how recovery behaviors address execution failures across both benchmarks.

4.2 Real-Robot Experiments

Setup. We evaluate four tasks on bimanual YAM workstations: packing a pencil box, stacking rings, and drawing and discarding a mahjong tile. Each workstation has two I2RT YAM arms with parallel grippers, four Intel RealSense cameras (overhead, front, and one on each wrist), and a 30 Hz control loop. For each task, the task and recovery policies are fine-tuned from the $\pi_{0.5}$ base checkpoint (Black et al., 2025a); the task policy observes the overhead and wrist cameras, and the recovery policy is conditioned on the recovery instruction. Gemini 3.8 Flash serves as the monitor and sees all four views. In each task's digital twin, the agent developed six or seven recovery skills, 25 in total (Sec. D). Each task is evaluated over 20 trials per configuration: the base task policy, after DAgger fine-tuning, and with recovery skills enabled at deployment.

Task success. DAgger fine-tuning more than triples the base task policy's mean success, from 23.8% to 77.5% (Tab. 3), and enabling recovery skills at deployment raises it further to 87.5%, the best result on every task. Recovery contributes the largest gains on ring stacking and pencil-box packing, 15.0 percentage points each, consistent with tasks in which a single misplaced ring

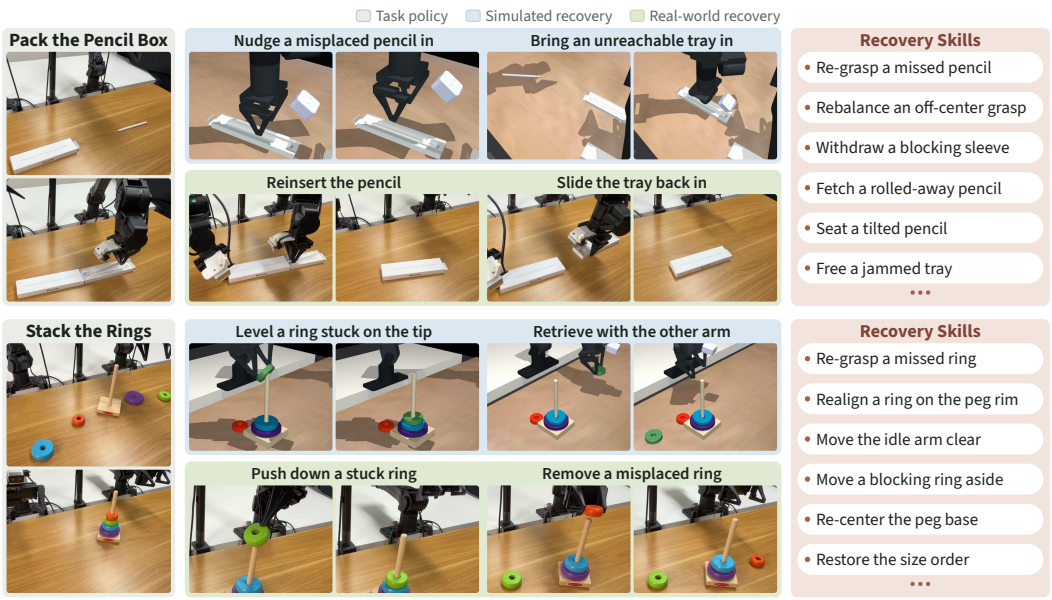


Figure 5: **Real-robot tasks and their recovery skills.** For each task, the gray panel shows the start and result of a task-policy rollout. Each recovery skill appears under its name as two frames of one rollout, the action and then the result, developed in the digital twin (blue) or executed on the robot by a learned recovery policy (green). The list on the right names further recovery skills.

or pencil blocks the remaining steps. On the two mahjong tasks, whose DAgger policies already succeed in at least 80% of trials, recovery still adds 5.0 points each.

DAgger rounds. Beyond task success, we track the human intervention rate, the fraction of collection episodes involving human takeover. Collection episodes include policy attempts and human demonstrations, excluding operator-discarded episodes. A verified recovery ends its episode, and resumed task execution begins a new attempt. The right panel of Fig. 4 follows four DAgger rounds on drawing a mahjong tile (seven or eight episodes each), with outcomes measured before the collected data are used for the next policy update. The fraction of collection episodes completed by the task policy rose from 12.5% to 85.7%. Human takeovers occurred in seven of eight episodes in the first round (87.5%) and none of seven in the fourth (0%). The middle panel shows a later 14-minute session on four stations at once: the task policy completed 22 of 31 episodes, and human control accounted for 11% of the summed station time.

Recovery skills. Fig. 5 shows two of the four tasks, each with two skills developed in the digital twin and two learned recovery policies executed on the robot. In *ring stacking*, the twin skills address failures such as a missed pickup or a ring caught on the peg tip, and on the robot, learned recovery policies push down a stuck ring and remove a ring stacked out of order. In *pencil-box packing*, they handle missed grasps, a pencil jammed at the tray, and objects out of reach, and the recovery policies reinsert the pencil and use both arms to slide a detached tray back into its sleeve. For the two mahjong tasks, the recovery policies stand a fallen tile back upright, return a drawn tile to the wall, and realign a displaced hand tile. Fig. 1 shows one real-robot recovery per task.

5 Conclusion

We presented Recova, an agent-guided framework that treats task execution and scene recovery as distinct, coordinated capabilities. A coding agent develops both in a reconstructed digital twin, and a deployment loop monitors the task policy, invokes and verifies recovery skills, and turns unresolved failures into demonstrations for the policy they can teach. Recova outperforms code-as-policy agents and VLA policies on two simulation benchmarks, its recovery skills improve real-robot success beyond DAgger fine-tuning alone, and the collection study shows declining human intervention over four rounds on one task. These results establish scene recovery as a learnable capability that complements task execution, and show that agents can turn failures into reusable skills. Automating the registration of newly learned skills and sharing them across tasks will further expand the set of recovery skills.

AI USE STATEMENT

In this work, we used generative AI tools for help generate synthetic data sets. We have not used generative AI tools for propose or refine hypotheses, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for draft parts of a research paper. We have reviewed all AI-assisted work. We checked the LLM-generated text. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

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. In *Conference on Robot Learning*, 2022. 3

Anthropic. System Card: Claude Opus 5.5. System card, Anthropic, September 2026. URL https://www.anthropic.com/claude-opus-5-5-system-card. 2

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. In *Conference on Robot Learning*, 2025a. 2, 3, 6, 7, 8, 18

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Robotics: Science and Systems*, 2025b. 2, 3, 6, 7

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X. Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. RoboCat: A Self-Improving Generalist Agent for Robotic Manipulation. *Transactions on Machine Learning Research*, 2024. 3

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. In *Conference on Robot Learning*, 2023a. 2, 3

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. RT-1: Robotics Transformer for Real-World Control at Scale. In *Robotics: Science and Systems*, 2023b. 3

Xiongyi Cai, Ri-Zhao Qiu, Geng Chen, Lai Wei, Isabella Liu, Tianshu Huang, Xuxin Cheng, and Xiaolong Wang. In-N-On: Scaling Egocentric Manipulation with in-the-wild and on-task Data. *arXiv preprint arXiv:2511.15704*, 2025. 3

Zoey Chen, Aaron Walsman, Marius Memmel, Kaichun Mo, Alex Fang, Dieter Fox, and Abhishek Gupta. URDFormer: A Pipeline for Constructing Articulated Simulation Environments from Real-World Images. In *Robotics: Science and Systems*, 2024. 3

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Robotics: Science and Systems*, 2023. 3

Cristina Cornelio and Mohammed Diab. Recover: A Neuro-Symbolic Framework for Failure Detection and Recovery. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2024. 3

Tianyuan Dai, Josiah Wong, Yunfan Jiang, Chen Wang, Cem Gokmen, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Automated Creation of Digital Cousins for Robust Policy Learning. In *Conference on Robot Learning*, 2024. 3

Yinpei Dai, Jayjun Lee, Nima Fazeli, and Joyce Chai. RACER: Rich Language-Guided Failure Recovery Policies for Imitation Learning. In *IEEE International Conference on Robotics and Automation*, 2025. 3

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. AHA: A Vision-Language-Model for Detecting and Reasoning Over Failures in Robotic Manipulation. In *International Conference on Learning Representations*, 2025. 3

Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, Shanli Xing, Jaemin Cho, Jae Sung Park, Ainaz Eftekhar, Peter Sushko, Karen Farley, Angad Wadhwa, Cole Harrison, Winson Han, Ying-Chun Lee, Eli VanderBilt, Rose Hendrix, Suveen Ellawela, Lucas Ngoo, Joyce Chai, Zhongzheng Ren, Ali Farhadi, Dieter Fox, and Ranjay Krishna. MolmoAct2: Action Reasoning Models for Real-world Deployment. *arXiv preprint arXiv:2605.02881*, 2026. 3, 6, 7

Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Fei-Fei Li, Guanya Shi, Jiajun Wu, Shankar Sastry, Yuke Zhu, Ken Goldberg, and Linxi Fan. CaP-X: A Framework for Benchmarking and Improving Coding Agents for Robot Manipulation. In *International Conference on Machine Learning*, 2026. 3, 6, 7

Seyed Kamyar Seyed Ghasemipour, Ayzaan Wahid, Jonathan Tompson, Pannag Sanketi, and Igor Mordatch. Self-Improving Embodied Foundation Models. In *Advances in Neural Information Processing Systems*, 2025. 3

Qiao Gu, Yuanliang Ju, Shengxiang Sun, Igor Gilitschenski, Haruki Nishimura, Masha Itkina, and Florian Shkurti. SAFE: Multitask Failure Detection for Vision-Language-Action Models. In *Advances in Neural Information Processing Systems*, 2025. 3

Yanjiang Guo, Yen-Jen Wang, Lihan Zha, and Jianyu Chen. DoReMi: Grounding Language Model by Detecting and Recovering from Plan-Execution Misalignment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2024. 3

Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S. Brown, and Ken Goldberg. ThriftyDAgger: Budget-Aware Novelty and Risk Gating for Interactive Imitation Learning. In *Conference on Robot Learning*, 2021a. 3

Ryan Hoque, Ashwin Balakrishna, Carl Putterman, Michael Luo, Daniel S. Brown, Daniel Seita, Brijen Thananjeyan, Ellen Novoseller, and Ken Goldberg. LazyDAgger: Reducing Context Switching in Interactive Imitation Learning. In *IEEE International Conference on Automation Science and Engineering*, 2021b. 3

Ryan Hoque, Lawrence Yunliang Chen, Satvik Sharma, Karthik Dharmarajan, Brijen Thananjeyan, Pieter Abbeel, and Ken Goldberg. Fleet-DAgger: Interactive Robot Fleet Learning with Scalable Human Supervision. In *Conference on Robot Learning*, 2022. 3

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. In *Conference on Robot Learning*, 2022. 3

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. In *Conference on Robot Learning*, 2023. 3

Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J. Kochenderfer. HG-DAgger: Interactive Imitation Learning with Human Experts. In *IEEE International Conference on Robotics and Automation*, 2019. 3

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Pannag Sanketi, Quan Vuong, et al. OpenVLA: An Open-Source Vision-Language-Action Model. In *Conference on Robot Learning*, 2024. 2, 3

Yejin Kim, Wilbert Pumacay, Omar Rayyan, Max Argus, Winson Han, Eli VanderBilt, Jordi Salvador, Abhay Deshpande, Rose Hendrix, Snehal Jauhri, Shuo Liu, Nur Muhammad Mahi Shafiullah, Maya Guru, Ainaz Eftekhar, Karen Farley, Donovan Clay, Jiafei Duan, Arjun Guru, Piper Wolters, Alvaro Herrasti, Ying-Chun Lee, Georgia Chalvatzaki, Yuchen Cui, Ali Farhadi, Dieter Fox, and Ranjay Krishna. MolmoSpaces: A Large-Scale Open Ecosystem for Robot Navigation and Manipulation. *arXiv preprint arXiv:2602.11337*, 2026. 3, 7

Bowen Li, Mayank Mishra, Y. Isabel Liu, Stone Tao, Nishanth Kumar, Alexander G. Gray, Ruwan Wickramarachchi, Jonathan Francis, Sebastian Scherer, and Tom Silver. Recover, Discover, Plan: Learning Skills and Concepts from Robot Failures. In *Conference on Robot Learning*, 2026a. 3

Huanyu Li, Kun Lei, Sheng Zang, Kaizhe Hu, Yongyuan Liang, Bo An, Xiaoli Li, and Huazhe Xu. Failure-Aware RL: Reliable Offline-to-Online Reinforcement Learning with Self-Recovery for Real-World Manipulation. *arXiv preprint arXiv:2601.07821*, 2026b. 2, 3

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. In *Conference on Robot Learning*, 2024. 3

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*, 2023. 2, 3, 4

Zijun Lin, Jiafei Duan, Haoquan Fang, Dieter Fox, Ranjay Krishna, Cheston Tan, and Bihan Wen. FailSafe: Reasoning and Recovery from Failures in Vision-Language-Action Models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2026. 3

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In *Advances in Neural Information Processing Systems*, 2023a. 3

Huihan Liu, Alice Chen, Yuke Zhu, Adith Swaminathan, Andrey Kolobov, and Ching-An Cheng. Interactive Robot Learning from Verbal Correction. *arXiv preprint arXiv:2310.17555*, 2023b. 3

Huihan Liu, Soroush Nasiriany, Lance Zhang, Zhiyao Bao, and Yuke Zhu. Robot Learning on the Job: Human-in-the-Loop Autonomy and Learning During Deployment. In *Robotics: Science and Systems*, 2023c. 2, 3

Huihan Liu, Shivin Dass, Roberto Martín-Martín, and Yuke Zhu. Model-Based Runtime Monitoring with Interactive Imitation Learning. In *IEEE International Conference on Robotics and Automation*, 2024a. 3

Huihan Liu, Yu Zhang, Vaarij Betala, Evan Zhang, James Liu, Crystal Ding, and Yuke Zhu. Multi-Task Interactive Robot Fleet Learning with Visual World Models. In *Conference on Robot Learning*, 2024b. 3

Isabella Liu, An-Chieh Cheng, Rui Yan, Geng Chen, Ri-Zhao Qiu, Xueyan Zou, Sha Yi, Hongxu Yin, Xiaolong Wang, and Sifei Liu. Long-Horizon Manipulation via Trace-Conditioned VLA Planning. *arXiv preprint arXiv:2604.21924*, 2026a. 3

Lin Liu, Zhicheng Bao, Lu Zhang, Ziying Song, Wu Yang, Yuzheng Zhuang, Shuai Tao, Wulong Liu, Caiyan Jia, and Huchuan Lu. LIBERO-RECOVER: Beyond Task Success Towards Failure Recovery in Robotic Manipulation Models. *arXiv preprint arXiv:2609.05178*, 2026b. 2, 3

Zeyi Liu, Arpit Bahety, and Shuran Song. REFLECT: Summarizing Robot Experiences for Failure Explanation and Correction. In *Conference on Robot Learning*, 2023d. 2, 3

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. 2, 3, 7

Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and Dexterous Robotic Manipulation via Human-in-the-Loop Reinforcement Learning. *Science Robotics*, 10(105):eads5033, 2025. 3

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-Level Reward Design via Coding Large Language Models. In *International Conference on Learning Representations*, 2024. 3

Zhao Mandi, Yijia Weng, Dominik Bauer, and Shuran Song. Real2Code: Reconstruct Articulated Objects via Code Generation. In *International Conference on Learning Representations*, 2025. 3

Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. Human-in-the-Loop Imitation Learning using Remote Teleoperation. *arXiv preprint arXiv:2012.06733*, 2020. 3

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. MimicGen: A Data Generation System for Scalable Robot Learning using Human Demonstrations. In *Conference on Robot Learning*, 2023. 3

Kunal Menda, Katherine Driggs-Campbell, and Mykel J. Kochenderfer. EnsembleDAgger: A Bayesian Approach to Safe Imitation Learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019. 3

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, et al. GR00T N1: An Open Foundation Model for Generalist Humanoid Robots. *arXiv preprint arXiv:2503.14734*, 2025. 2, 3

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. In *Robotics: Science and Systems*, 2024. 3

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, et al. Open X-Embodiment: Robotic Learning Datasets and RT-X Models. In *IEEE International Conference on Robotics and Automation*, 2024. 3

OpenAI. GPT-6 Astra System Card. System card, OpenAI, September 2026. URL https://deploymentsafety.openai.com/gpt-6-astra. 2

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient Action Tokenization for Vision-Language-Action Models. In *Robotics: Science and Systems*, 2025. 3, 6, 7

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *International Conference on Artificial Intelligence and Statistics*, 2011. 3, 6

Lucy Xiaoyang Shi, Zheyuan Hu, Tony Z. Zhao, Archit Sharma, Karl Pertsch, Jianlan Luo, Sergey Levine, and Chelsea Finn. Yell At Your Robot: Improving On-the-Fly from Language Corrections. In *Robotics: Science and Systems*, 2024. 3

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. ProgPrompt: Generating Situated Robot Task Plans using Large Language Models. In *IEEE International Conference on Robotics and Automation*, 2023. 3

Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E. Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery RL: Safe Reinforcement Learning with Learned Recovery Zones. *IEEE Robotics and Automation Letters*, 6(3):4915–4922, 2021. 3

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A Physics Engine for Model-Based Control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012. 5

Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling Reality through Simulation: A Real-to-Sim-to-Real Approach for Robust Manipulation. In *Robotics: Science and Systems*, 2024. 2, 3

Shivam Vats, Devesh K. Jha, Maxim Likhachev, Oliver Kroemer, and Diego Romeres. RecoveryChaining: Learning Local Recovery Policies for Robust Manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2025. 3

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. *Transactions on Machine Learning Research*, 2024a. 3

Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. GenSim: Generating Robotic Simulation Tasks via Large Language Models. In *International Conference on Learning Representations*, 2024b. 3

Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. RoboGen: Towards Unleashing Infinite Data for Automated Robot Learning via Generative Simulation. In *International Conference on Machine Learning*, 2024c. 3

Yilin Wu, Ran Tian, Gokul Swamy, and Andrea Bajcsy. From Foresight to Forethought: VLM-In-the-Loop Policy Steering via Latent Alignment. In *Robotics: Science and Systems*, 2025. 3

Wenli Xiao, Jia Xie, Tonghe Zhang, Haotian Lin, Letian Fu, Haoru Xue, Jalen Lu, Yi Yang, Cunxi Dai, Zi Wang, Jimmy Wu, Guanzhi Wang, S. Shankar Sastry, Ken Goldberg, Linxi Fan, Yuke Zhu, and Guanya Shi. ENPIRE: Agentic Robot Policy Self-Improvement in the Real World. In *Conference on Robot Learning*, 2026. 3

Chen Xu, Tony Khuong Nguyen, Emma Dixon, Christopher Rodriguez, Patrick Miller, Robert Lee, Paarth Shah, Rares Ambrus, Haruki Nishimura, and Masha Itkina. Can We Detect Failures Without Failure Data? Uncertainty-Aware Runtime Failure Detection for Imitation Learning Policies. In *Robotics: Science and Systems*, 2025. 3

Zewei Ye, Weifeng Lu, Minghao Ye, Tao Lin, Shuo Yang, Junchi Yan, and Bo Zhao. RoboFAC: A Comprehensive Framework for Robotic Failure Analysis and Correction. *arXiv preprint arXiv:2505.12224*, 2025. 3

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. In *Conference on Robot Learning*, 2026. 3, 6, 7

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Robotics: Science and Systems*, 2023. 3

Enshen Zhou, Qi Su, Cheng Chi, Zhizheng Zhang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, and He Wang. Code-as-Monitor: Constraint-aware Visual Programming for Reactive and Proactive Robotic Failure Detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025a. 3

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*, 2025b. 2, 3, 7

Zhiyuan Zhou, Pranav Atreya, Abraham Lee, Homer Walke, Oier Mees, and Sergey Levine. Autonomous Improvement of Instruction Following Skills via Foundation Models. In *Conference on Robot Learning*, 2024. 3

Appendix Table of Contents

A Real-Robot Orchestration **15**

B Monitor Prompts and Example Verdicts **15**

C Implementation and Evaluation Details **16**

C.1 Hardware and Tasks 16

C.2 Real-Robot Monitor and Data Interface 17

C.3 Policy Training 18

C.4 Evaluation Protocol 19

C.5 Trajectory Provenance 19

C.6 Qualitative Video Examples 19

D Simulated Recovery Skills **19**

E Recovery Examples on the Simulation Benchmarks **21**

A Real-Robot Orchestration

Alg. 1 gives the control flow that the real-robot harness runs in deployment round k (Sec. 3.3). The monitor M is realized as four vision-language-model queries: completion (M_{done}), intervention (M_{int}, which also reports whether the scene is intact and gives a recovery instruction z), restoration (M_{res}), and requirement generation (M_{req}). I_0 and I_t are the initial and current images from the monitor's four cameras, H_t holds the two most recent snapshots, taken one second apart, and V_t holds the round's last three intervention verdicts. $\mathcal{K}$ is the set of known recovery instructions, which the intervention query reuses verbatim when one applies; $\mathcal{K}_\rho \subseteq \mathcal{K}$ holds those in the learned-skill registry, for which the recovery policy ρ_ϕ is deployed. Step counts are control steps at 30 Hz, with the harness defaults shown.

The harness also handles details that Alg. 1 omits. Every verdict carries its round and submission step, and verdicts from an ended round or from before a manual scene adjustment are discarded. When an intervention verdict arrives while a completion query is still running, the harness holds the robot until the completion verdict lands. Intervention queries run asynchronously, so the task policy keeps acting while they are evaluated. If no recovery instruction is available, the human segment is labeled "Restore the scene to its pre-task state." The operator can also pause a rollout to adjust the scene by hand, which adds no demonstration; take over at any time, which records a task demonstration in $\mathcal{E}_k$; request a task or recovery demonstration directly; flip the pending label before it is saved; and stop or discard the round. A discarded prefix is excluded from the training data but remains in the session log with the monitor's prompts and verdicts.

B Monitor Prompts and Example Verdicts

The monitor sends every query with the same system prompt and one of four templates (Fig. 6). Each image is a 448×336 composite of the four cameras at one moment: top and front views in the upper row and the two wrist views in the lower row (the prompts call the front camera the bottom camera). The requirement query sees the four initial views; the completion and restoration queries see the initial and current composites; and the intervention query sees the initial composite and the two most recent snapshots, together with its three previous verdicts and the known recovery skills for the task. All queries run at temperature 0 and return JSON. Across the four-station collection sessions, the median latency is 4.5 s for intervention, 5.0 s for completion, and 5.5 s for restoration queries. Fig. 7 shows recorded verdicts from these sessions.

Algorithm 1 Recova's orchestration in real-robot deployment round k.

Input: task g with cached completion requirement C_g; policies π_θ and ρ_ϕ; instructions $\mathcal{K}$ and registry $\mathcal{K}_\rho \subseteq \mathcal{K}$; check intervals $N_c = 600$ and $N_i = 300$; step limits $T_{\max} = 5400$ and $T_{\text{rec}} = 1500$; demonstration threshold $c_{\max} = 2$
Output: trajectory collections $\mathcal{S}_k$, $\mathcal{E}_k$, $\mathcal{R}_k$, and $\mathcal{A}_k$

$c \leftarrow 0$ ▷ consecutive interventions on g
while continuous operation is enabled **do** ▷ one task attempt per iteration
  return the robot to its home pose
  if $c \geq c_{\max}$ **then**
    a human demonstrates g; add the demonstration to $\mathcal{E}_k$; $c \leftarrow 0$
    continue
  $I_0 \leftarrow$ images of the current scene
  if C_g is empty **then**
    $C_g \leftarrow M_{\text{req}}(g, I_0)$ with the robot stationary; cache C_g
  start recording trajectory τ; $t \leftarrow 0$; $d \leftarrow 0$
  repeat
    execute $a_t \sim \pi_\theta(\cdot \mid o_t, g)$; $t \leftarrow t + 1$
    if $t \bmod N_c = 0$ **then**
      $d \leftarrow M_{\text{done}}(g, C_g, I_0, I_t)$ with the robot held
    if $t \bmod N_i = 0$ **then**
      submit $M_{\text{int}}(g, C_g, I_0, H_t, V_t, \mathcal{K})$; π_θ keeps running
    if $d = 1$ **then** ▷ completion takes precedence
      add τ to $\mathcal{S}_k$; $c \leftarrow 0$
    else if an intervention verdict $(intact, z)$ has arrived **then**
      $c \leftarrow c + 1$; discard τ; return the robot to its home pose
      if $intact$ **then** ▷ stalled progress, no disturbance
        a human finishes g; add the demonstration to $\mathcal{E}_k$; $c \leftarrow 0$
      else
        RECOVER(z) ▷ the next attempt resumes π_θ
  until the attempt has ended or $t = T_{\max}$ ▷ τ is discarded at $T_{\max}$

procedure RECOVER(z)
  if $z \in \mathcal{K}_\rho$ **then** ▷ one bounded autonomous attempt
    start recording trajectory τ'
    for $s = 1, \ldots, T_{\text{rec}}$ **do**
      execute $a_s \sim \rho_\phi(\cdot \mid o_s, z)$
      if $s \bmod N_i = 0$ or $s = T_{\text{rec}}$ **then**
        if $M_{\text{res}}(z, I_0, I_s) = 1$ (robot held) **then**
          add τ' to $\mathcal{A}_k$ ▷ verified restoration
          return
    return the robot to its home pose ▷ not restored in time
  a human performs only the recovery; add the segment, labeled z, to $\mathcal{R}_k$

C Implementation and Evaluation Details

C.1 Hardware and Tasks

Each workstation has two YAM arms, each with six joints and a parallel gripper, four Intel RealSense cameras recording at 640×480 (top, front, and one on each wrist), and a pair of leader arms through which the operator takes over. A status display at each station turns green while a policy acts and red when the operator is needed. Tab. 4 lists the four tasks and their success criteria; each episode is limited to 3 minutes (5,400 control steps). During parallel collection, one operator supervises four stations, each arranged with a different background and set of distractor objects (Fig. 4). Rollouts start automatically, and drawing sessions continue until the hand holds eight tiles.

```
SYSTEM  You are a robot-task monitor. You watch a dual-arm robot manipulation rollout
        through four cameras (top, bottom, and one on each wrist) and judge its status.
        Respond ONLY with a JSON object, no markdown fences.

REQUIREMENT  Task instruction: "{instruction}"   [four initial views]
  Write the completion requirement for this task: a concise, objectively checkable
  description of what the scene must look like once the task has been fully completed
  (object positions/arrangement, gripper state, anything that must NOT have changed).
  1-3 sentences, only about the final scene state, verifiable purely from camera images.
  -> {"requirement": "<the requirement>"}

COMPLETION  Task instruction, completion requirement   [initial and current composites]
  Has the task been fully completed? Compare against the initial scene to judge what has
  changed. Be strict: answer true only when the requirement is clearly and completely
  satisfied; partial progress or an ambiguous view is false. Check EVERY moved object:
  if any object left its initial position but sits at no valid goal position, the task
  is NOT complete.
  -> {"complete": true/false, "reason": "<one short sentence>"}

INTERVENTION  Task instruction, completion requirement, operator notes
              [initial composite, snapshots 1 s ago and now, previous verdicts]
  Decide whether a HUMAN needs to take over NOW. Set intervene = true only when the
  snapshots clearly show one of:
  - an object accidentally dropped, fallen, or knocked over;
  - the robot stuck on ONE atomic action for more than ~30 s (use your previous verdicts);
  - an EXTRA object moved by mistake lies loose (neither at its start nor at a goal);
  - for hand-tile tasks: a hand tile standing tilted or not facing forward;
  - the scene disturbed so badly the task cannot proceed.
  A deliberate intermediate placement is NOT a failure. When intervene is true, set
  scene_intact = true if nothing is displaced (the robot merely failed to make progress;
  recovery is empty), or false with ONE short imperative recovery instruction.
  Known recovery skills: <list>. When a listed skill matches, return it VERBATIM;
  otherwise write a new one.
  -> {"intervene": true/false, "reason": "...", "scene_intact": true/false,
      "recovery": "<one imperative clause or empty>"}

RESTORATION  Recovery instruction: "{recovery}"   [initial and current composites]
  Has the recovery been completed? The INITIAL images define what a valid pre-task state
  looks like. If the current arrangement matches it or is an equivalently workable
  starting state, the scene IS restored. Exact object positions need not match.
  -> {"restored": true/false, "reason": "<one short sentence>"}
```

Figure 6: **Monitor prompts**, abridged. Bracketed items describe the images and context attached to each query; −> gives the required JSON reply.

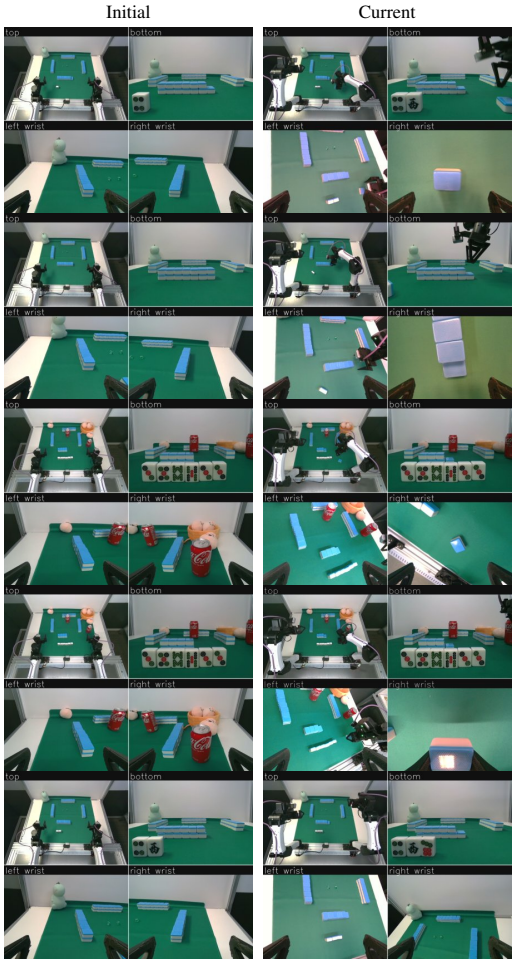
Table 4: **Real-robot tasks and success criteria.**

Task	Success criterion
Pencil box	The pencil lies in its tray, and the tray is slid back into the box sleeve.
Stack rings	The four rings sit on the peg in size order.
Draw tile	Exactly one more tile stands upright at the end of the hand row, facing the front camera; no tile has fallen, and the tile wall is otherwise unchanged.
Discard tile	The specified hand tile lies face-up in the central discard area, the other hand tiles are undisturbed, and both grippers are empty.

C.2 Real-Robot Monitor and Data Interface

The real-robot harness runs a 30 Hz control loop. The monitor receives all four camera views (top, front, and both wrists), and the task policy uses the top and wrist cameras. The task description, completion requirement, and behavior notes are cached across collection sessions; when no completion requirement exists yet, the monitor generates it with the robot stationary before recording starts.

Completion and intervention checks run every 600 and 300 control steps (20 s and 10 s at 30 Hz), and each autonomous recovery attempt is limited to 1,500 steps (50 s). The robot holds still during completion and restoration checks, while intervention checks run asynchronously. The intervention query uses the initial scene, strided recent frames, recent verdicts, and the known recovery instructions, and distinguishes a stalled task in an intact scene from a scene that needs correction. The restoration query accepts any arrangement equivalent to the initial one rather than requiring exact



(a) Intervention: {"intervene": true, "scene_intact": false, "reason": "The hand row is complete, but an extra tile has been moved loose onto the right side of the table.", "recovery": "Return the loose extra tile to the tile wall."}

(b) Intervention: {"intervene": true, "scene_intact": true, "reason": "The robot has made no progress on re-grasping the deliberately staged tile for over 30 seconds.", "recovery": ""}

(c) Intervention: {"intervene": true, "scene_intact": false, "reason": "The drawn tile has fallen flat after the robot's attempt to stand it upright.", "recovery": "Stand the fallen tile upright and insert it to the row."}

(c, after recovery) Restoration: {"restored": true, "reason": "The tiles are upright in the row, and the current arrangement matches the initial workable state."}

(d) Completion: {"complete": true, "reason": "Exactly one upright, forward-facing tile has been added to the right end of the hand, with no fallen tiles and the remaining wall unchanged."}

Figure 7: **Recorded monitor verdicts** while drawing a mahjong tile. Each image is the four-camera composite sent to the monitor. (a) The monitor reuses a known recovery skill verbatim. (b) A stalled task in an intact scene leads to a task demonstration. (c) The recovery policy executes the proposed instruction, and the restoration query confirms the scene. (d) The completion query accepts a finished task.

object poses. Before a segment is saved, the operator can relabel it, which keeps task and recovery data distinct.

A learned-skill registry determines which recovery instructions the recovery policy handles; after training, newly learned instructions are added to it, and the online harness reads it. A registered skill receives one bounded attempt; otherwise, or if restoration fails, the operator takes control. After two consecutive interventions on a task, the harness requests a new task demonstration; the count resets after a successful task or a task demonstration. Successful autonomous recoveries need no human demonstration.

C.3 Policy Training

All task and recovery policies are fine-tuned from the $\pi_{0.5}$ base checkpoint (Black et al., 2025a) with the same recipe (Tab. 5). Actions are 14-dimensional joint-position targets for both arms, six joints and one gripper each, predicted in chunks of 50 steps (1.67 s at 30 Hz); joint targets are learned as offsets from the current state, and gripper commands as absolute values. The policies observe the robot state together with the top and two wrist images at 224×224, and the language prompt is

the task instruction for the task policy and the recovery instruction for the recovery policy. Training proceeds in stages, each starting from the previous checkpoint as more data arrive.

Table 5: **Policy fine-tuning hyperparameters**, shared by all task and recovery policies.

Setting	Value
Initialization	$\pi_{0.5}$ base checkpoint
Actions	14-D joint positions (joint offsets, absolute grippers)
Action chunk	50 steps (1.67 s at 30 Hz)
Image inputs	top, left wrist, right wrist; 224×224
Optimizer	AdamW, gradient clipping 1.0
Learning rate	cosine decay from 2.5×10^{-5} to 2.5×10^{-6}
Warmup	300–1,000 steps
Batch size	64
EMA decay	0.99
Steps per stage	2,000–5,000
Compute	8 NVIDIA L40 GPUs, 1–3 h per stage

C.4 Evaluation Protocol

Tab. 3 reports 20 trials for each task and configuration. An operator judges each trial against the task's success criterion in Tab. 4.

C.5 Trajectory Provenance

Each saved trajectory is stored with its instruction and type: a `Recover` trajectory is labeled with its recovery instruction, and `Success` and `Expand` trajectories with the original task. The session log records the acting controller, episode boundaries, and outcomes, together with the monitor's images, prompts, responses, and timestamps. Failed policy prefixes are excluded from the demonstrations but kept in this log, so every accepted recovery can be audited without using failed segments as supervision.

C.6 Qualitative Video Examples

Fig. 5 takes two frames from each of ten recordings: two real task-policy rollouts, four real recovery-policy rollouts, and four simulated recovery skills rendered from the digital twin's inspection camera. Frames use a fixed crop and uniform resizing, without retouching.

D Simulated Recovery Skills

Figs. 8 and 9 show 24 recovery skills that the agent developed in the digital twins of the four real-robot tasks, including the simulated skills pictured in Fig. 5. Each skill appears as four frames of one simulated rollout, from the failure to the result, taken from the twin's inspection camera; for moving the idle arm clear, the top camera shows the motion.

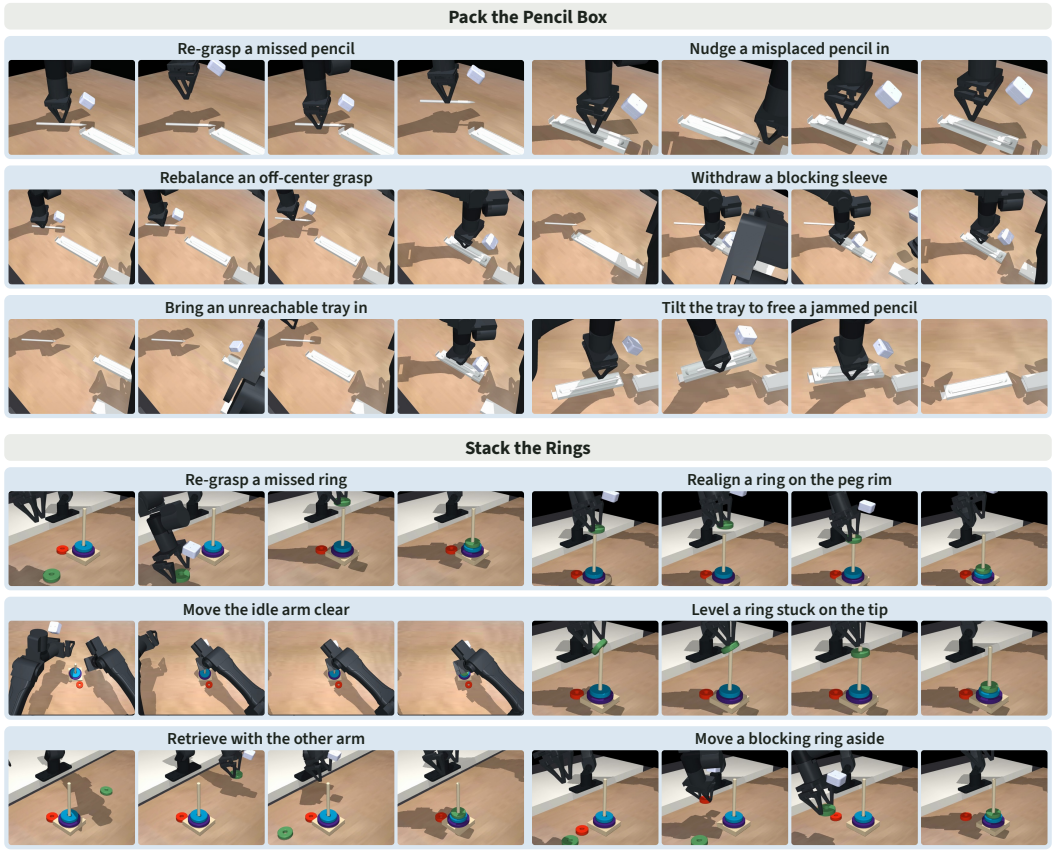


Figure 8: **Simulated recovery skills for pencil-box packing and ring stacking.** Each blue panel shows one recovery skill developed in the task's digital twin, as four frames of one rollout, from the failure to the result. Frames come from the twin's inspection camera, except *Move the idle arm clear* (top camera).

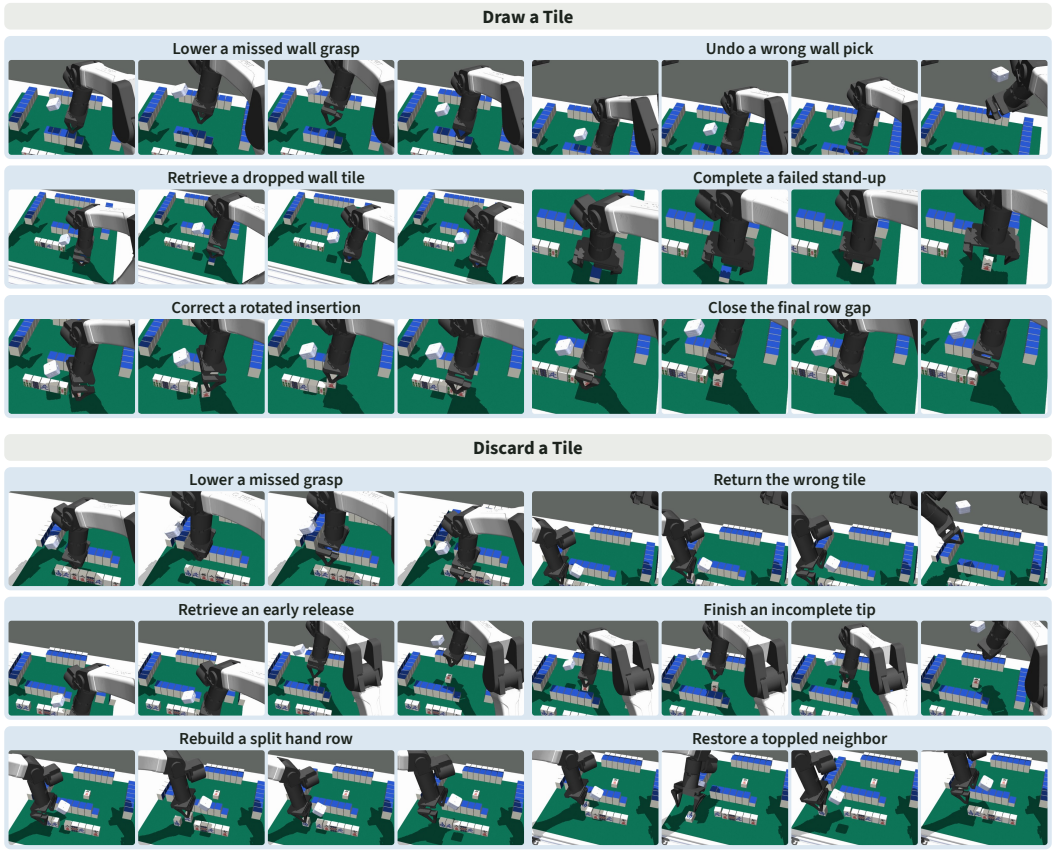


Figure 9: **Simulated recovery skills for drawing and discarding a mahjong tile.** Panels follow the conventions of Fig. 8.

E Recovery Examples on the Simulation Benchmarks

Fig. 10 extends Fig. 3 with 16 further episodes, eight from each simulation benchmark. Each episode fails with recovery disabled and succeeds, from the same initial state, with recovery enabled. Both frames of an episode come from the run with recovery: the failure frame just after the setback that triggers the recovery, and the recovery frame just before the simulator first reports success. The eight LIBERO-Pro episodes show seven distinct recoveries episodes, and the eight MolmoSpaces episodes show four recovery skills credited, including "try another direction" on four articulated objects. Episodes were chosen to cover distinct recoveries and scenes. Frames are cropped from the recordings' camera views and resized uniformly, without retouching.

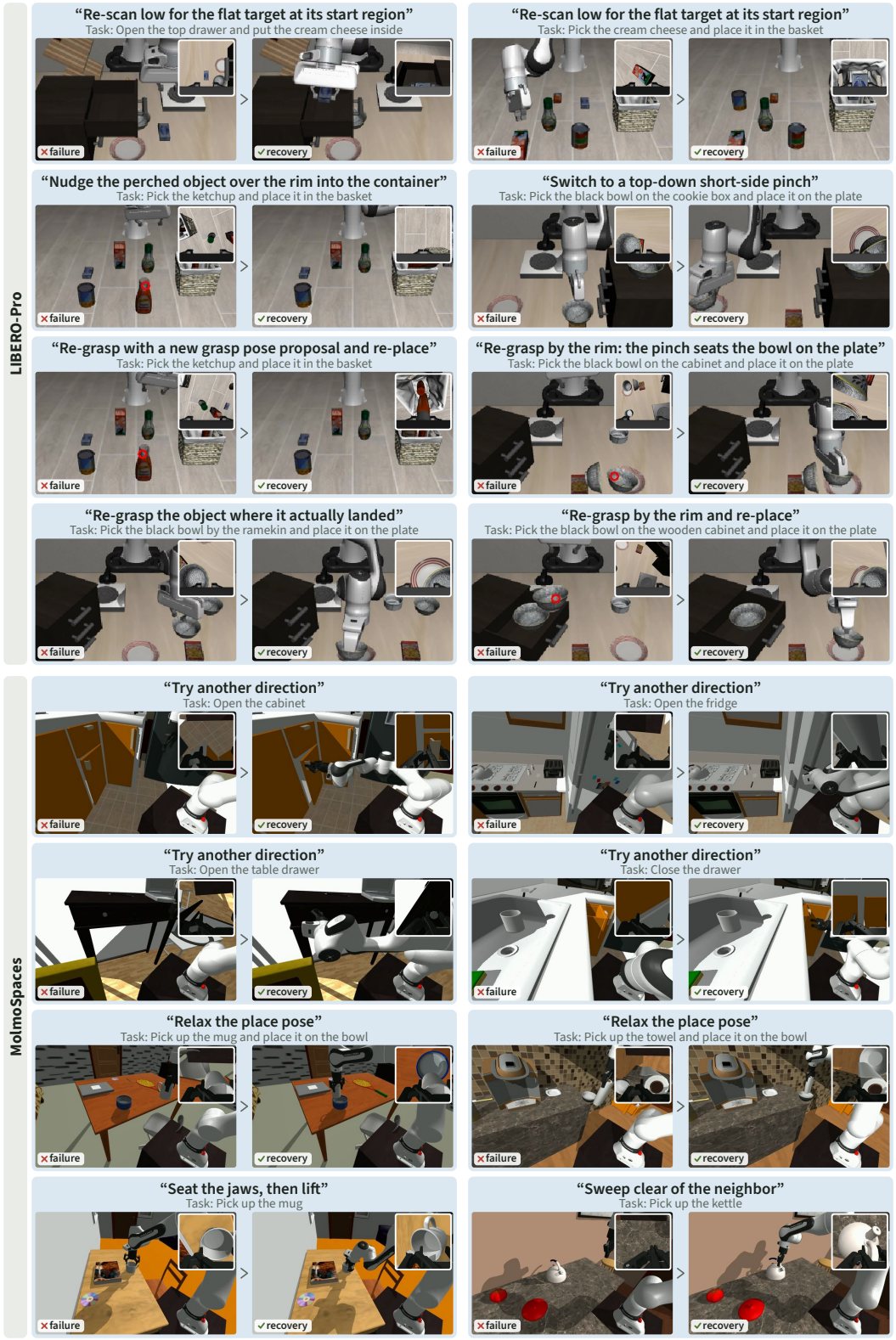


Figure 10: **Recovery examples on LIBERO-Pro (top) and MolmoSpaces (bottom).** Each panel shows one episode's failure and recovery moments under its recovery instruction and task, as in Fig. 3. Red rings in some LIBERO-Pro frames are markers from the source recordings.