Training Reasoning Models in Simulation to Natively Use Real Robots

Jiaming Tang [*1 5 †] **Zhuoyang Zhang** [*1] **Junxian Guo** [*1 ‡] **Fangzhou Zhao** [2 5] **Yilong Zhao** [2] **Zekai Li** [3]
Xingyang Li [1] **Yuyang Zhao** [4] **Song Bian** [4] **Zhijian Liu** [3 5] **Song Han** [1 4]

https://github.com/mit-han-lab/pear

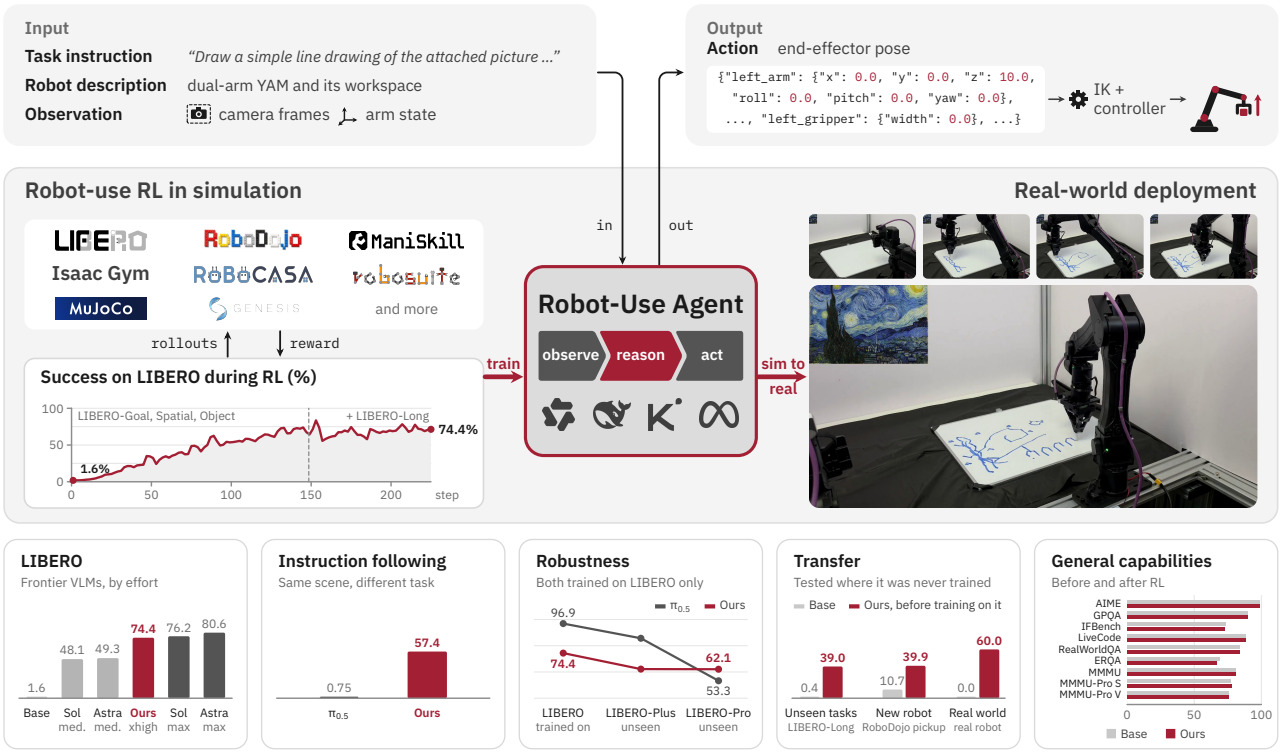


Figure 1. **Overview of robot-use reinforcement learning (RURL).** Like a computer-use agent, the robot-use agent (RUA) observes the scene, reasons in a chain of thought, and generates the next end-effector pose (top). It is trained with reinforcement learning entirely in simulation (left) and deployed on a real robot (right). The logos show examples of simulators and open-weight models; we train Qwen3.8-27B in LIBERO and RoboDojo. Bottom: results on LIBERO, instruction following, robustness, transfer to unseen tasks, a new robot and the real world, and general capabilities, all without a harness.

Abstract

Robot-use agents (RUAs) have emerged as a promising approach to general-purpose robot control. However, strong performance has so far been achieved only by frontier proprietary models such as GPT-6 Astra and Claude Fable 5.2, whereas open-weight models remain substantially less capable. We show that an open-weight 27B reasoning model can be trained to use real robots natively through reinforcement learning, entirely in simulation and without demonstrations. Like a computer-use agent (CUA), the model observes the environment, reasons in a chain of thought, and generates the next end-effector pose, without any harness. We build a Post-training Engine for Agentic Robotics (PEAR) and use it to train the model on simulated robot tasks at scale with robot-use reinforcement learning (RURL), mirroring how frontier models acquire computer use. We find that the skills acquired in simulation

[*]Equal contribution [†]Part of this work was done during an internship at Inco AI. [‡]Part of this work was done during an internship at NVIDIA. [1]MIT [2]UC Berkeley [3]UCSD [4]NVIDIA [5]Inco AI. Correspondence to: Song Han <songhan@mit.edu>.

Preprint. November 12, 2026.

generalize well beyond the training tasks: they transfer to an unseen robot, to the real world, and to novel instructions. After training on LIBERO and RoboDojo in simulation, the model succeeds in 60% of our trials on a real dual-arm robot, whereas the base model succeeds in none. On LIBERO, training raises the success rate from 1.6% to 74.4%, approaching GPT-6 Astra and GPT-6.1 Sol at maximum reasoning effort. Compared with $\pi_{0.5}$, a vision-language-action model fine-tuned on demonstrations, our model is substantially more robust to changes in the task itself. On LIBERO-Pro, $\pi_{0.5}$ succeeds in only 0.75% of episodes when assigned a new task in a familiar scene whereas our model succeeds in 57.4%. Finally, the model retains its general capabilities, with no measurable degradation on nine reasoning, coding and vision benchmarks. These results highlight robot-use reinforcement learning as a promising component of foundation-model post-training.

1. Introduction

Robot-use agents (RUAs) have recently become a promising direction for general-purpose robot control (Isola, 2026; Yang et al., 2026c; Zhang et al., 2026c; Lu et al., 2026). Similar to a computer-use agent (CUA) that operates a computer from screenshots (OpenAI, 2025a; Anthropic, 2024), an RUA uses a vision-language model (VLM) that observes the robot's camera images, reasons about the task, and outputs the next end-effector pose. Because the VLM operates the robot directly, RUAs take advantage of the broad world knowledge and the strong reasoning ability of a capable foundation model, which allows them to generalize and to complete complex tasks.

RUAs built on frontier models such as GPT-6 Astra (OpenAI, 2026b; Pikus et al., 2026; Liu et al., 2026a) and Claude Fable 5.2 (Anthropic, 2026) have shown exceptional generalization and language following. On RoboDojo (Chen et al., 2026; Zhang et al., 2026b) and RoboLab (Yang et al., 2026b;a; SenseNova et al., 2026), agents built on GPT-6 Astra have outperformed the best previous policies. In contrast, vision-language-action models (VLAs) (Zitkovich et al., 2023; Kim et al., 2025; Black et al., 2025b;a) fine-tuned on task-specific demonstrations often struggle to generalize to new goals and follow novel instructions (Liu et al., 2025a; Zhou et al., 2025; Fei et al., 2026; Fang et al., 2026; Kim et al., 2026; Hou & Zhao, 2026; Zhang et al., 2026a). Two factors may contribute to the performance of frontier RUAs: the models have been trained for computer use, and they have strong spatial reasoning abilities. We hypothesize that the ability to reason in a chain of thought is particularly important for generalization.

However, such performance has so far been obtained only with the strongest proprietary models. Open-weight models have low success rates on the same benchmarks and fail on some simple tasks as well. Several works build a harness that adds skills and components such as the segmentation model SAM 3 (Carion et al., 2026) and turns some motions into a small set of options (Yang et al., 2026c; Lu et al., 2026; Zhang et al., 2026c; Chu et al., 2026; Huang et al., 2026; Si et al., 2026; RoboDojo Team, 2026). Such a harness allows open-weight models to complete some of the tasks (Yang et al., 2026c), but it may restrict their action space and thereby make other tasks infeasible. As models become more capable, some components of the harness may also be absorbed into the models themselves.

Computer-use agents such as OpenAI's are trained with reinforcement learning (RL) on computer tasks (OpenAI, 2025a;b; Wang et al., 2025), and the close resemblance between the two settings suggests that robot use could be learned in the same way. We therefore train an open-weight reasoning model with robot-use reinforcement learning (RURL), entirely in simulation and without demonstrations. To support this long-horizon, multi-turn RURL at scale, we build PEAR (a Post-training Engine for Agentic Robotics), an infrastructure designed specifically for training RUAs (Section 3.2). In each round, the model observes camera images and the robot's state, reasons about the task, and outputs the next end-effector pose (Figure 1). Inverse kinematics converts this pose into joint targets, and the robot's controller executes the movement before the model receives the next observation. Earlier rounds remain in the context, and each episode receives a binary reward indicating whether the task was completed (Figure 2). We provide no perception tools, planners, or skill libraries, so the model itself must localize objects, determine how to manipulate them, and recognize when an action has failed. Because the model specifies poses rather than joint commands and each robot's own inverse kinematics and controller execute them, the same interface applies in simulation and on a real robot.

Although the model is trained entirely in simulation, the skills it learns transfer well to the real world. We train the model first on LIBERO (Liu et al., 2023) and then on two tasks in RoboDojo (Chen et al., 2026). On a real dual-arm robot that it never encountered during training, the model then completes 60% of our trials, whereas the base model completes none (Section 4.4). The checkpoint trained only on the single-arm LIBERO tasks also succeeds in 23% of the trials. These results suggest that simulation-based RL can equip general-purpose reasoning models with robot-use skills that transfer to unseen real-world embodiments.

Our main findings are as follows:

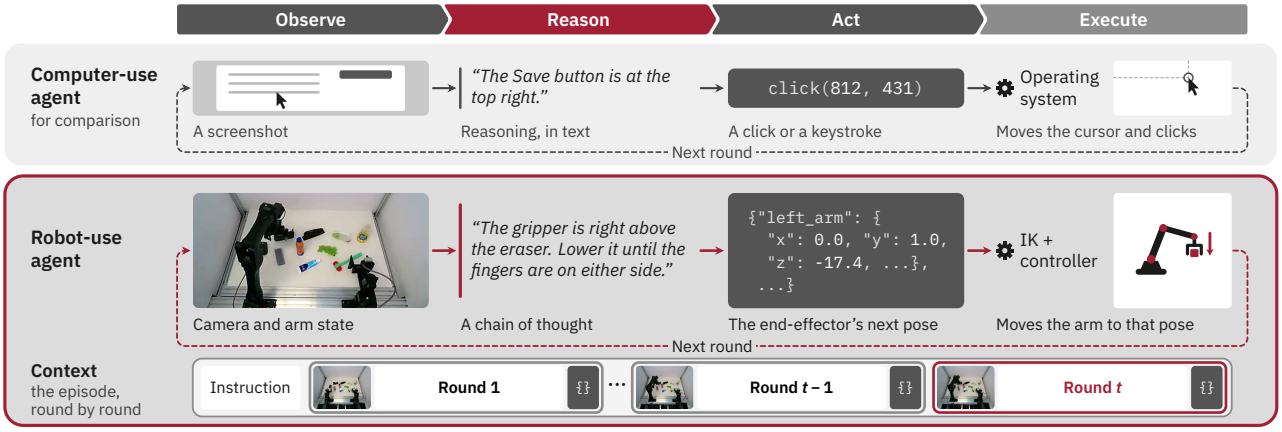


Figure 2. **One round of robot use.** A computer-use agent (top) observes a screenshot, reasons about the task and outputs a click or a keystroke, which the operating system executes. Our model (bottom) operates a robot in a similar way: it observes the camera views and the arm's state, reasons in a chain of thought and outputs the end-effector's next pose, and inverse kinematics and the arm's controller then move the arm. Each round is added to the context, which also holds the instruction and all earlier rounds. The same steps are used in simulation and on the real robot.

- **Approaching frontier models on LIBERO.** The model improves from 1.6% to 74.4% on LIBERO, outperforms GPT-6.1 Sol (OpenAI, 2026a) and GPT-6 Astra at medium effort (48.1% and 49.3%), and closes most of the gap to them at maximum effort (76.2% and 80.6%; Section 4.1).
- **Robustness and instruction following.** The model remains robust when its training tasks are perturbed and follows instructions it has never seen. LIBERO-Pro (Zhou et al., 2025) changes LIBERO's goals and layouts, and on this benchmark the model outperforms the VLA $\pi_{0.5}$ (Black et al., 2025a) (62.1% vs. 53.3%; Section 4.2).
- **Zero-shot transfer to unseen tasks and robots.** Without any training on LIBERO-Long, the model reaches 39.0% on this suite, on which the base model succeeds in 0.4% of episodes. On two RoboDojo tasks with a dual-arm robot, the model trained only on LIBERO succeeds in 39.9% and 17.0% of episodes, compared with 10.7% and 13.0% for the base model. On RoboLab-120, training only on LIBERO raises the success rate from 5.4% to 22.1%, and the model trained on LIBERO and RoboDojo outperforms the $\pi_{0.5}$ entry on the RoboLab leaderboard (37.3% vs. 28.0%; Section 4.3).
- **Sim-to-real transfer.** The skills also transfer zero-shot from simulation to the real world: without any real-robot data, the model trained on LIBERO and RoboDojo succeeds in 60% of our trials on a real dual-arm robot and the model trained only on LIBERO in 23%, whereas the base model succeeds in none (Section 4.4). Beyond these trials, the model also performs reasoning tasks unlike any seen during training: it draws a simple line version of a painting on a whiteboard and selects a suitable reagent to test whether a solution is alkaline (Figure 7c).
- **Retained general capabilities.** Robot-use training causes no measurable loss in performance on nine reasoning, coding and vision benchmarks (Section 4.5). This result suggests that RURL can be included as a component of foundation-model post-training.

2. Related Work

Robot-use agents. Frontier VLMs such as GPT-6 Astra (OpenAI, 2026b) have been evaluated as RUAs in simulation and on real robots (Zhang et al., 2026b; Yang et al., 2026a; SenseNova et al., 2026; Leet et al., 2026; Liu et al., 2026a), and RoboDojo (Chen et al., 2026) ranks robot agents in a separate track (RoboDojo Team, 2026). Many recent works build RUA harnesses that provide perception tools, skill libraries, motion primitives, memory or code execution, including ORA (Yang et al., 2026c), ASPIRE (Lu et al., 2026), Harness VLA (Zhang et al., 2026c), OpenRUA (Chu et al., 2026), RAX (Huang et al., 2026), unirobot-agent (Yuan et al., 2026) and PyRUA-Lean (Si et al., 2026). Concurrent work also trains open-weight models inside such harnesses, with RL in PyRUA-Lean and with supervised fine-tuning followed by multi-turn RL in CaPtain (Fan et al., 2026), AgenticRoboRL (Liu et al., 2026b) and ZAP (Pikus et al., 2026). In contrast, our model uses no harness and no demonstrations. It is trained with RL alone and transfers from simulation to a real robot.

RL for computer-use agents. Computer-use agents operate a computer from screenshots (OpenAI, 2025a; Anthropic, 2024; Qin et al., 2025) and are evaluated on benchmarks such as OSWorld (Xie et al., 2024) and WebArena (Zhou et al., 2024). OpenAI reports that its agent is trained with supervised learning followed by RL (OpenAI, 2025b). Open-weight agents have been trained with online multi-turn RL

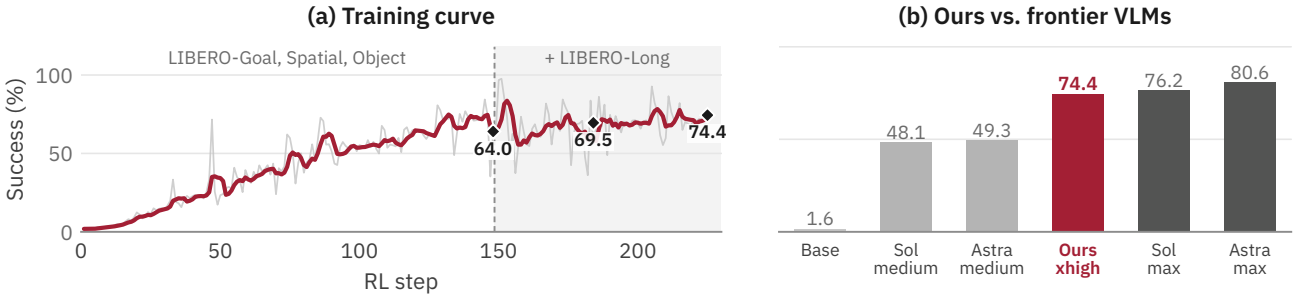


Figure 3. **Training curve on LIBERO and comparison with frontier VLMs.** (a) The success rate of training rollouts per RL step (thin line) and smoothed (thick line); LIBERO-Long is added at step 149, and the diamonds are the same run evaluated on LIBERO's 2,000 episodes. (b) The base model, our model at xhigh, the effort used in training, and GPT-6.1 Sol (OpenAI, 2026a) and GPT-6 Astra (OpenAI, 2026b) at medium and max effort, on LIBERO's 2,000 evaluation episodes; all models use the same protocol, and effort is each model's own setting for how long it reasons.

on Android devices in DigiRL (Bai et al., 2024), on the web in WebRL (Qi et al., 2025) and on desktop operating systems in UI-TARS-2 (Wang et al., 2025). RL with verifiable rewards is also widely used to train long chains of thought (Shao et al., 2024; Guo et al., 2025; Yu et al., 2025). RL on a new task erodes a model's other skills less than supervised fine-tuning does (Shenfeld et al., 2026), and it generalizes better (Chu et al., 2025). RURL applies this recipe to simulated robot tasks and uses the reasoning model itself as the policy.

3. Robot-Use Reinforcement Learning

Our goal is to train a reasoning model in simulation to use a real robot. We first describe how a robot-use agent operates a robot and the protocol that we use in simulation and on the real robot (Section 3.1). We then present our training infrastructure PEAR and the RL algorithm that it runs (Section 3.2). Because the base model rarely succeeds at first, training follows a curriculum (Section 3.3).

3.1. Robot-use agent

A computer-use agent (OpenAI, 2025a; Anthropic, 2024) observes a screenshot, reasons about the task, and outputs a click or a keystroke, which the operating system then executes (Figure 2, top). Our model operates a robot (Isola, 2026) in the same way (Figure 2, bottom). In each round, it observes the camera frames and the arm's state, reasons in a chain of thought, and outputs the end-effector's next pose. Inverse kinematics then converts this pose into joint angles, and the arm's controller executes the move.

We use no perception tools, grasp or motion planners (Yang et al., 2026c), or skill libraries (Zhang et al., 2026c; Lu et al., 2026), so the model itself has to localize the objects in the camera frames, determine how to reach and manipulate them, and detect when a move has failed. Earlier rounds and the instruction remain in the context, so each decision can take the entire episode into account. The model acquires all of these abilities through RL on robot tasks in simulation (Section 3.2).

The system prompt states the protocol of this interaction, describes the robot's coordinate frame and cameras, and gives the task instruction (Section H). The protocol describes a robot as a set of named parts with named quantities: the position and orientation of each arm's end effector. The state lists the current value of every quantity as a JSON object, and an action is a JSON object of the same structure that gives the change of every quantity, in centimeters and radians. The controller moves the robot until every target is reached and the robot is at rest, or until the robot is blocked short of a target. An episode ends when the simulator reports success, when the benchmark's step limit or the token budget of the context is reached, or when a response contains no valid action.

Because an action is a pose-level target and the controller stays outside the model, the protocol is identical for every robot: only the list of parts and the description of the robot change between LIBERO's single-arm Franka (Liu et al., 2023), RoboDojo's dual-arm ARX (Chen et al., 2026) and the real dual-arm YAM. Deployment on a real robot therefore requires no further training: the model acts through the same protocol, the prompt describes the real robot's frame and cameras, and the robot's own inverse kinematics and controller execute the actions.

3.2. PEAR: a Post-training Engine for Agentic Robotics

We build PEAR, an infrastructure for training RUAs, on top of veRL (Sheng et al., 2025) and LMCache (Liu et al., 2025b). We train the model with reinforcement learning on robot tasks in simulation, following the recipe used for computer use (OpenAI, 2025a;b; Wang et al., 2025). Let $\tau = (o_1, y_1, \ldots, o_T, y_T)$ denote an episode, where o_t is the observation of round t and y_t the model's response, which contains its reasoning and its action. The policy π_θ generates each response token by token given the context

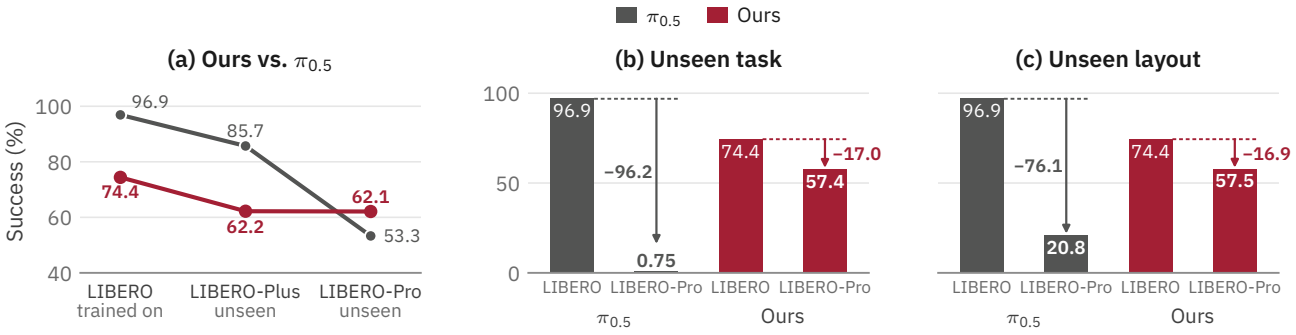


Figure 4. **Comparison with $\pi_{0.5}$ on LIBERO, LIBERO-Plus and LIBERO-Pro.** Both policies are trained only on LIBERO: $\pi_{0.5}$ with supervised fine-tuning on demonstrations and our model with RURL. (a) Success rate on LIBERO and on the two perturbed benchmarks. (b, c) Two of LIBERO-Pro's perturbations, an unseen task in a training scene and swapped object positions; the arrow gives the drop from LIBERO. The $\pi_{0.5}$ results are from prior work (Physical Intelligence, 2025; Yan et al., 2026; Zhou et al., 2025).

$h_t = (p, o_1, y_1, \ldots, o_t)$, where p is the system prompt. The reward $R(\tau)$ is 1 if the simulator reports success by the end of the episode and 0 otherwise, and training maximizes its expectation over the training tasks.

We use the group-normalized advantages of GRPO (Shao et al., 2024): we sample groups of G episodes $\tau_1, \ldots, \tau_G$ that share a task and an initial state and set

$$\hat{A}_i = \frac{R(\tau_i) - \operatorname{mean}_j R(\tau_j)}{\operatorname{std}_j R(\tau_j) + \epsilon}, \tag{1}$$

where ϵ is a small constant. Every generated token of an episode receives the advantage of that episode, whether it belongs to the reasoning or to an action and whichever round it is in, and the gradient estimate averages over the N generated tokens of a batch:

$$g = \frac{1}{N} \sum_i \hat{A}_i \sum_{t,k} \Big[\nabla_\theta \log \pi_\theta\big(y_{i,t}^k \mid c_{i,t}^k\big) - \mathbb{E}_{v \sim \mu(\cdot \mid c_{i,t}^k)} \nabla_\theta \log \pi_\theta\big(v \mid c_{i,t}^k\big) \Big], \tag{2}$$

where $y_{i,t}^k$ is the k-th token of the response in round t of episode i, $c_{i,t}^k = (h_{i,t}, y_{i,t}^{<k})$ is its context, and μ is the policy that generated the episode. The first term is the REINFORCE estimator (Williams, 1992) with the group mean as a baseline, used without PPO clipping (Schulman et al., 2017). Rollouts and updates run fully asynchronously, so μ lags one or two versions behind the policy being trained. The second term is the score-centering correction (Marek & Ryabinin, 2026) for this mismatch. We use this correction alone, without truncated importance sampling. As in DAPO (Yu et al., 2025), groups in which every episode succeeds or every episode fails are discarded and replaced because their advantages are zero.

Training is long-horizon, multi-round agentic RL, so a high prefix-cache hit rate is important for efficient rollouts. We therefore use LMCache to offload the prefix KV cache to CPU memory and limit the number of concurrent episodes to keep the prefix cache hit rate high. For the policy update, a whole episode forms one training sequence, which we fit in memory by combining FSDP (Zhao et al., 2023) with sequence parallelism (Jacobs et al., 2024) and gradient checkpointing (Chen et al., 2016). Section A gives the configuration for each type of GPU.

3.3. Training curriculum

With a binary reward, the advantage in Equation (1) is nonzero only in groups that contain both successful and failed episodes, so a task on which the model never succeeds provides no learning signal. At the start of training, almost every episode fails: the base model succeeds in about 2% of episodes on LIBERO's Spatial, Object and Goal suites, in 0.4% on its long-horizon suite LIBERO-Long, and almost never on most RoboDojo tasks.

We therefore train first on the tasks that already provide a signal and rely on transfer to extend it to the others. The skills learned on the first tasks can raise the success rate on tasks that the model has not been trained on, and we add such tasks to training once the model begins to succeed on them. In our experiments, training proceeds in three stages: LIBERO's Goal, Spatial and Object suites, then all four LIBERO suites, and then the two RoboDojo tasks that provide the strongest signal after training on LIBERO (`general_pickup` and `align_blocks`; Section 4). Excluding LIBERO-Long from the first stage also provides a clean test of transfer (Section 4.3).

4. Experiments

Training. We train the open-weight reasoning model Qwen3.8-27B (Qwen Team, 2026) with the curriculum of Section 3.3: 148 steps on LIBERO's Goal, Spatial and Object suites, steps 149 to 225 on all four suites, and 115 further steps on the two RoboDojo tasks. We refer to the checkpoint at step 225 as the LIBERO-only model and la-

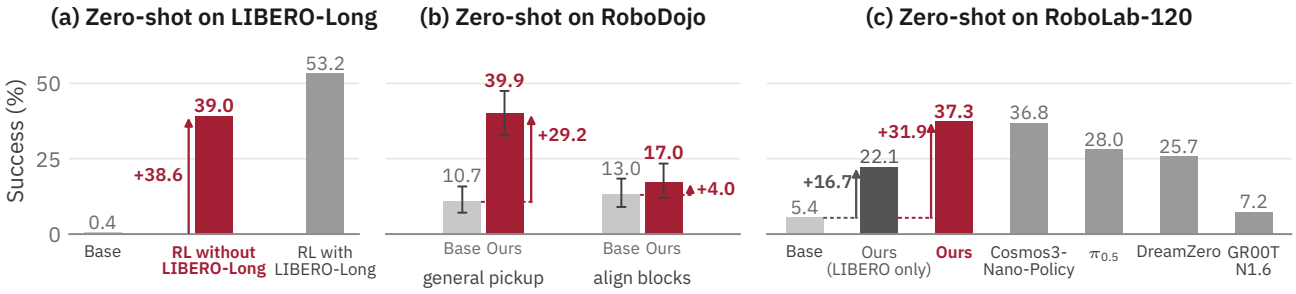


Figure 5. **Zero-shot success rate on unseen tasks, a new simulator and robot, and a new benchmark.** (a) RL without LIBERO-Long is trained on the other three LIBERO suites; RL with LIBERO-Long, the same run after LIBERO-Long is added, is shown for reference. (b) RoboDojo, a different simulator with a dual-arm ARX, after RL on LIBERO only; error bars are 95% Wilson intervals. (c) RoboLab-120, a third benchmark, against four generalist robot policies from the RoboLab leaderboard (NVIDIA, 2026c), all DROID (Khazatsky et al., 2024) checkpoints.

bel it Ours (LIBERO only) in figures and tables. The last checkpoint is the final model, Qwen3.8-27B-PEAR, which we deploy on the real robot and label Ours.

Evaluation. On LIBERO (Liu et al., 2023), we report the success rate over its 2,000 evaluation episodes (50 for each of its 40 tasks). LIBERO-Plus (Fei et al., 2026) perturbs LIBERO along seven dimensions: camera, lighting, background, sensor noise, instruction wording, layout and the robot's initial state (10,030 episodes for our model). From LIBERO-Pro (Zhou et al., 2025), we use four perturbation types (2,000 episodes each for our model): a different task in the same scene, swapped object positions, a reworded instruction and restyled objects. In RoboDojo (Chen et al., 2026), we evaluate `general_pickup` and `align_blocks` with 163 to 200 episodes per task and report 95% Wilson intervals. RoboLab-120 (Yang et al., 2026b) comprises 120 tabletop tasks at three levels of difficulty, performed by a single-arm Franka with a Robotiq gripper. We run ten episodes per task (1,200 in total) and report the mean of two evaluation runs for the final model. On the real dual-arm YAM, we run 30 trials for each model: two in each of five scenes with the instruction "Use the set square to push the three blocks into a straight, aligned row, then release the set square and reset the robot arm" and two for each of ten objects with the instruction "Pick up the ⟨object⟩ by 10 cm." A trial fails if it takes more than 25 rounds or if the model reasons for more than 32k tokens in a single round. When the model outputs an invalid action that inverse kinematics cannot solve, it retries, and the retry also counts as a round. Unless stated otherwise, our models run at xhigh effort.

Baselines. Frontier vision-language models use the same protocol as our model (Section 3.1): GPT-6.1 Sol (OpenAI, 2026a) and GPT-6 Astra (OpenAI, 2026b) at the medium and max effort settings. Effort is each model's own setting for how long it reasons. $\pi_{0.5}$ (Black et al., 2025a) is a VLA supervised fine-tuned on demonstrations of every LIBERO task. We do not evaluate $\pi_{0.5}$ ourselves and use results reported for its released checkpoint. Its LIBERO results are from the openpi repository (Physical Intelligence, 2025), its LIBERO-Plus result is from Yan et al. (2026), and its LIBERO-Pro results are the per-suite results of Zhou et al. (2025) averaged over the four suites. On RoboLab-120, we compare with four generalist robot policies from the RoboLab leaderboard (NVIDIA, 2026c): Cosmos3-Nano-Policy (NVIDIA, 2026a), $\pi_{0.5}$, DreamZero (Ye et al., 2026) and GR00T N1.6 (NVIDIA GEAR Team, 2025). All four are checkpoints trained on the real-robot dataset DROID (Khazatsky et al., 2024) and evaluated by the RoboLab organizers, so this $\pi_{0.5}$ is a different checkpoint from the LIBERO one above.

4.1. Learning robot use on LIBERO

RL improves the policy from a near-zero start. On Goal, Spatial and Object, the base model succeeds in about 2% of rollouts, so almost every rollout ends with no reward. The training success rate on these three suites still rises to about 70% within 148 steps (Figure 3a). The skills learned on these three suites also transfer well to tasks that the model has never been trained on. At step 148, before it is trained on LIBERO-Long, the model solves 39.0% of it, up from 0.4% for the base model (Figure 5a). Its success rate on LIBERO as a whole is 64.0% at this point, and we return to this transfer in Section 4.3. Training on all four suites from step 149 then raises the success rate to 69.5% at step 184 and to 74.4% at step 225, each about 5 points above the last. At 74.4%, our model approaches GPT-6 Astra (80.6%; OpenAI, 2026b) and GPT-6.1 Sol (76.2%; OpenAI, 2026a) at max effort and exceeds both at medium effort (49.3% and 48.1%; Figure 3b).

These results show that a base model with a low initial success rate can be trained to a high success rate with RL alone, from 1.6% to 74.4% in this case. The capability it acquires is also general rather than specific to LIBERO's training tasks: the following sections show that the model remains robust when LIBERO's scenes and tasks are perturbed, follows instructions it was never trained on (Section 4.2), and

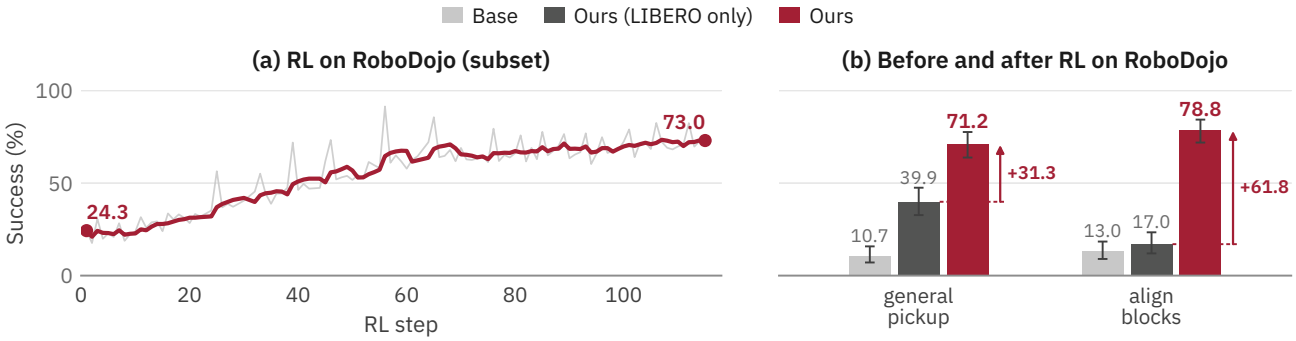


Figure 6. **Training curve on RoboDojo subset and success rate before and after RL.** The final model continues from the LIBERO-only model with RURL on two RoboDojo tasks, entirely in simulation. (a) Success rate of training rollouts per RL step (thin line) and smoothed (thick line). (b) Success rate on the two tasks before and after this training; error bars are 95% Wilson intervals.

transfers to a new benchmark, a new robot and the real world (Sections 4.3 and 4.4).

4.2. Robustness and instruction following

Trained only on LIBERO, our model generalizes beyond its training tasks: it remains robust when they are perturbed and follows instructions that it has never seen (Figure 4). $\pi_{0.5}$ (Black et al., 2025a) outperforms our model on LIBERO, where it is supervised fine-tuned on demonstrations of every task, but it loses this lead when a task's goal or object layout changes. We test both policies on LIBERO-Plus (Fei et al., 2026) and LIBERO-Pro (Zhou et al., 2025), which perturb the training tasks in ways that neither policy has seen. LIBERO-Plus mainly perturbs the observation, such as the camera, the lighting, the background and sensor noise, and the two policies lose a similar amount on it: 11.2 points for $\pi_{0.5}$ and 12.2 for our model. LIBERO-Pro also changes the goals and the object layouts. Our model again loses about 12 points there, but $\pi_{0.5}$ loses 43.6, so its 22.5-point lead on LIBERO becomes an 8.8-point deficit (Figure 4a).

Our model's lead on LIBERO-Pro comes from the perturbations that change what the robot has to do. The task perturbation keeps a training scene and its initial state but gives an instruction for a different task. $\pi_{0.5}$ falls from 96.9% on LIBERO to 0.75%, a drop of 96.2 points, while our model falls from 74.4% to 57.4%, a drop of 17.0 points (Figure 4b). $\pi_{0.5}$ tends to carry out the task it was trained on in that scene, while our model does what the new instruction asks. The position perturbation tests the same ability from the other side: it keeps the instruction but swaps the objects' positions, so following the instruction now takes a different motion. $\pi_{0.5}$ drops by 76.1 points to 20.8%, and our model by 16.9 points to 57.5% (Figure 4c).

$\pi_{0.5}$ still leads on the other two perturbations, which reword the instruction or restyle the objects but leave the original task in place (Figure 10b). Thus, $\pi_{0.5}$ remains robust when the motion it was shown is still correct and fails when the motion has to change, whereas our model loses between 1 and 17 points on each perturbation but keeps following the instruction. This difference may come from imitating demonstrations versus reasoning about the task: our model keeps the base model's ability to read an instruction and a scene, and RL teaches it to act on what it reads.

4.3. Zero-shot transfer to unseen tasks and robots

Without training on LIBERO-Long episodes, our model reaches 39.0% on LIBERO-Long, where the base model succeeds in 0.4% of episodes (Figure 5a). RL could hardly have learned LIBERO-Long from the base model, because a model that almost never succeeds receives almost no reward and has little to learn from. Transfer from the other three suites changes this: when LIBERO-Long is added to training at step 149, the model's successes on it give RL a signal to train on, and its success rate on LIBERO-Long reaches 53.2%.

The skills also transfer when both the simulator and the robot change. RoboDojo (Chen et al., 2026) is a different environment, and its tasks use a dual-arm ARX instead of LIBERO's single-arm Franka. Trained only on LIBERO, our model succeeds in 39.9% of RoboDojo's `general_pickup` episodes and 17.0% of its `align_blocks` episodes, compared with 10.7% and 13.0% for the base model (Figure 5b). What the model learns on LIBERO is therefore not tied to LIBERO's simulator or to its robot.

We also evaluate the final model zero-shot on RoboLab-120 (Yang et al., 2026b), a third simulation benchmark. It was trained with RL only on LIBERO and RoboDojo (Section 3.3), and RoboLab's tasks and scenes are new to it. It succeeds in 37.3% of RoboLab-120's 1,200 episodes, while the base model succeeds in 5.4%. This is on par with Cosmos3-Nano-Policy (NVIDIA, 2026a) at 36.8% and above $\pi_{0.5}$ (Black et al., 2025a) at 28.0%, DreamZero (Ye et al., 2026) at 25.7% and GR00T N1.6 (NVIDIA GEAR Team, 2025) at 7.2%, four generalist robot policies on the

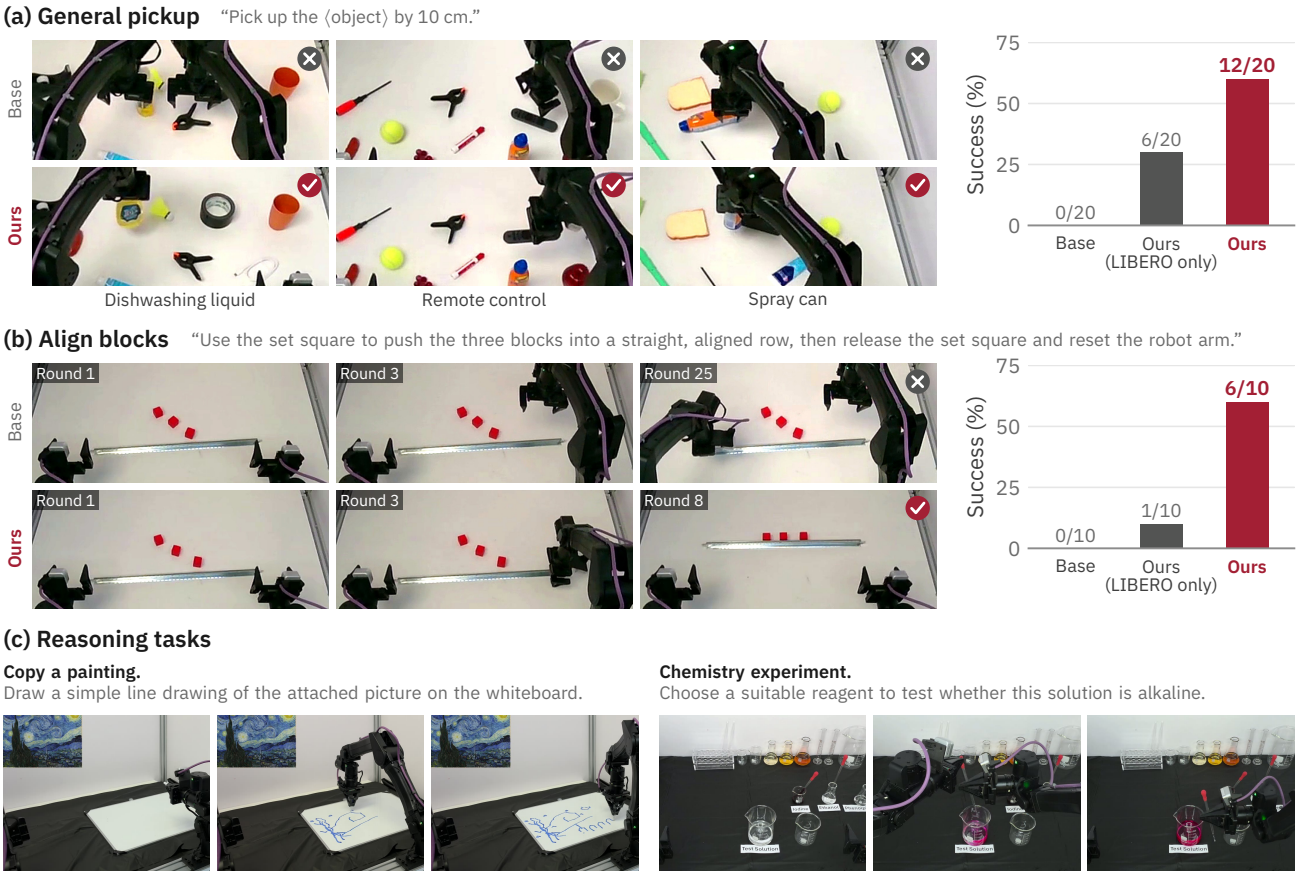


Figure 7. **Real-robot evaluation on a dual-arm YAM.** Our training does not include this robot and uses no real-robot data. (a) General pickup: three trials in which the final model picks up the object and the base model does not; the bar chart gives each model's successful trials out of 20, two for each of ten objects. (b) Align blocks: three rounds of one trial in which the final model succeeds and the base model fails; the bar chart gives each model's successful trials out of ten. (c) Reasoning tasks: the final model draws a simple line version of a painting and selects a reagent that shows whether a solution is alkaline.

RoboLab leaderboard (NVIDIA, 2026c) (Figure 5c). The LIBERO-only model succeeds in 22.1%. Section D reports the results by task difficulty.

4.4. Sim-to-real transfer

Starting from the LIBERO-only model, we continue training with RL on two RoboDojo tasks, entirely in simulation. RL works here as it does on LIBERO, on a different simulator and a dual-arm robot: in 115 steps, the training success rate on the two tasks rises from about 25% to about 70% (Figure 6a). Evaluated in RoboDojo, the final model succeeds in 71.2% of `general_pickup` episodes and 78.8% of `align_blocks` episodes, up from 39.9% and 17.0% before this stage (Figure 6b).

We then evaluate the models zero-shot on a real dual-arm YAM, a robot that no model was trained on. Without any real-robot data, the final model succeeds in 18 of 30 trials, whereas the base model succeeds in none. The LIBERO-only model succeeds in 7 of the 30 trials, although it has seen neither these tasks nor this robot (Figure 7a and b). The successful trials of the final model take between 7 and 24 rounds and use between 23k and 132k tokens of context; Section G lists every trial and the cause of each failure.

Reasoning tasks. The trials above repeat the two tasks on which the final model was trained in RoboDojo. We additionally give it two reasoning tasks, which we define as tasks that have no counterpart in its training and that depend on reasoning about what to do rather than on a trained motion (Figure 7c). For such tasks, the prompt describes the robot and the task in detail (Section 5). Given a picture of *The Starry Night*, the final model draws a simple line version of the painting on a whiteboard. To test whether a solution is alkaline, it selects phenolphthalein rather than iodine or ethanol because phenolphthalein turns pink in an alkaline solution, and the solution turns pink once the reagent is added. Such tasks require world knowledge and reasoning ability, which stem from the foundation model.

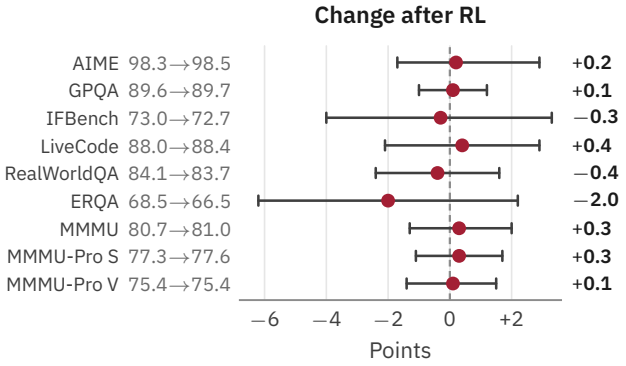


Figure 8. **General benchmarks before and after RL.** The final model against the base model on the same questions: the change in score (dots) with its paired 95% interval (lines), and the scores before and after RL beside each name. The benchmarks are AIME 2026, GPQA Diamond, IFBench, LiveCodeBench v6 (LiveCode), RealWorldQA, ERQA, the MMMU validation set and MMMU-Pro in its standard (S) and vision (V) settings; the mean change over the nine is −0.15 points. Table 6 gives the full numbers and the evaluation settings.

4.5. Retaining general capabilities

We compare the final model with the base model on the same questions using paired comparisons. On all nine benchmarks (MAA, 2026; Rein et al., 2024; Pyatkin et al., 2025; Jain et al., 2025; xAI, 2024; Gemini Robotics Team et al., 2025; Yue et al., 2024; 2025), the 95% interval includes zero (Figure 8). This is consistent with the finding that RL on a new task erodes a model's other skills less than supervised fine-tuning does (Shenfeld et al., 2026).

While reaching 75.0% on RoboDojo's `general_pickup` and `align_blocks`, the final model scores 98.5 on AIME 2026 (MAA, 2026), 89.7 on GPQA Diamond (Rein et al., 2024) and 88.4 on LiveCodeBench (Jain et al., 2025), and remains within two points of the base model on every vision benchmark. The same model now reasons, writes code, answers questions about images and uses a robot.

4.6. Reasoning effort and inference speed

Although the model is trained only at xhigh effort, the effort setting in the chat template controls how long it reasons. No RL rollout runs at a lower setting, yet at low effort the model uses 0.86k thinking tokens per round instead of 3.23k (Figure 9a), so RL at a single effort has not fixed the length of its reasoning. Its success rate falls with that length: on LIBERO, it is 21.9% at low effort (the mean of two evaluation runs) and 74.4% at xhigh. With less than a third of the thinking, it keeps less than a third of its success (Figure 9b). Most of what it can do on LIBERO therefore comes from reasoning at length before each move.

Because the model reasons at length before every move, how fast it decodes determines how fast it can act. With NVFP4 weights (Alvarez et al., 2025) and a DFlash2 drafter (Inco AI, 2026a), it decodes 52 tokens/s on an M5 Max in Inco's Splash engine (Inco AI, 2026b). We optimize our megakernel with KDA (NVIDIA, 2026b). With the megakernel, it decodes 221 tokens/s on a single RTX 5090 and 747 tokens/s on a single GB300 (Figure 9c). At 747 tokens/s, a round of xhigh reasoning takes about 4.3 s and a round at low effort about 1.1 s, so the effort setting also trades success for speed. We use Inco's original DFlash2 checkpoint, with an average acceptance length of 3.2 tokens. A drafter tuned specifically for our model may have a longer acceptance length, which would make decoding faster. The model can therefore run locally on a single consumer GPU or a laptop. However, NVFP4 quantization reduces our model's accuracy and lengthens its chain of thought, so all evaluations use BF16 weights.

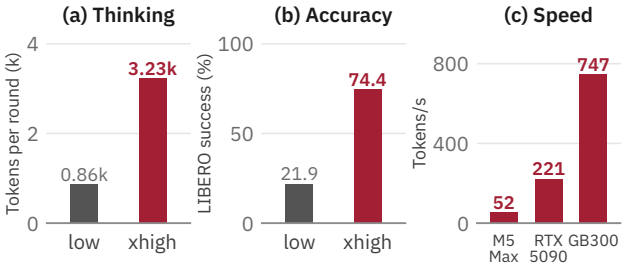


Figure 9. **Reasoning effort and inference speed.** (a, b) Thinking tokens per round and success rate on LIBERO of the LIBERO-only model at low and xhigh effort, with BF16 weights; the low-effort values are the mean of two evaluation runs. (c) Decoding speed on one device, with NVFP4 weights quantized with NVIDIA ModelOpt (NVIDIA, 2024) and Inco's DFlash2 drafter throughout; our megakernel on the GPUs and Inco's Splash engine on the M5 Max.

5. Limitations

- We have not been able to scale up our training. Mainly because of limited GPU resources, we stopped after training on two RoboDojo tasks, and the training was stable. Training so far also covers only two simulators and a small set of tasks, and it starts from a 27B base model. We believe that scaling up, by training in more diverse simulations and on more diverse tasks and by starting from a stronger foundation model, could lead to stronger robot-use performance.
- We did not use supervised fine-tuning (SFT), for two reasons. First, we lacked high-quality in-distribution data and were concerned that SFT would harm the model's general capabilities (Shenfeld et al., 2026). Second, we think that robotics data, even when available, should be added at the pretraining and mid-training stages rather than used for SFT on a model that has already been post-trained. We therefore train with RL alone.

- Our model is still slow at carrying out tasks, because it reasons at length before every move. We apply no length penalty during RL and place no constraint on the length of its chain of thought. The inference speed is high, at 747 tokens/s on a single GB300, but a round of reasoning still takes about 4.3 s on average (Figure 9), and in a few cases the chain of thought can reach 32k tokens, so that a single move takes more than half a minute. Shortening the reasoning without giving up success, for example by rewarding shorter chains of thought during RL, is left to future work.
- Our base model, Qwen3.8-27B, is relatively small, and with its limited capabilities it often lacks knowledge about the embodiment it controls, so this information has to be provided in the prompt. As a result, the task success rate can depend considerably on whether the model can form a clear understanding of the embodiment and the environment from the prompt, and the model reaches its best performance when the prompt gives it enough of the information it needs. For harder tasks, such as those in Figure 7c, our model succeeds only after we describe the embodiment and the task to it in detail in the prompt. Its performance therefore depends heavily on the prompt, but this also demonstrates, to some extent, its ability to follow detailed language instructions.
- Our current model is not yet able to plan action chunks reliably, so it outputs the end-effector's next pose; inverse kinematics and a controller then move the arm to that pose and stop it there. Accuracy therefore depends partly on the controller: a poorly implemented one can move the arm abruptly at some positions and miss the target pose, and an inefficient one consumes the task's step budget and leaves fewer moves, which matters mainly in simulated evaluation. This design also increases the number of rounds a task needs: since each round moves the arm to a single pose, a complex motion that the model could otherwise complete in one round with one action chunk now has to be split into several rounds. We hope future work can improve this.

6. Conclusion

A general reasoning model can learn to use real robots entirely in simulation in the same way that frontier models learn to use computers. It acts through an interface with camera frames as input and end-effector poses as output and is trained with RL on robot tasks. After training only on LIBERO and RoboDojo in simulation, our model succeeds in 60% of our trials on a real dual-arm robot, whereas the base model succeeds in none. Starting from a base model that rarely succeeds at robot tasks, RURL also yields a model that follows instructions it was never trained on, remains robust when its training tasks are perturbed, transfers to a new simulator and a new robot, and retains its general capabilities. RURL in simulation thus offers a simple recipe for real-world robot-use agents with substantial room for improvement, and our results suggest that reasoning allows such agents to generalize well beyond the tasks they were trained on. We hope that these results encourage further work on RURL as a component of foundation-model post-training.

Acknowledgements

We thank MIT-IBM Watson AI Lab, Amazon and MIT Science Hub, MIT AI Hardware Program, National Science Foundation, and Samsung for supporting this research. We thank NVIDIA for donating the DGX server. We thank Shuo Yang for his insightful discussion on model deployment.

References

Alvarez, E., Almog, O., Chung, E., Layton, S., Stosic, D., Krashinsky, R., and Aubrey, K. Introducing NVFP4 for efficient and accurate low-precision inference. https://developer.nvidia.com/blog/introducing-nvfp4-for-efficient-and-accurate-low-precision-inference/, 2025.

Anthropic. Introducing computer use, a new Claude 3.5 Sonnet, and Claude 3.5 Haiku. https://www.anthropic.com/news/3-5-models-and-computer-use, 2024.

Anthropic. Introducing Claude Fable 5.2 and Claude Mythos 5.2. https://www.anthropic.com/claude-fable-and-mythos-5-2, 2026. November 9, 2026.

Bai, H., Zhou, Y., Cemri, M., Pan, J., Suhr, A., Levine, S., and Kumar, A. DigiRL: Training in-the-wild device-control agents with autonomous reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. doi: 10.52202/079017-0397.

Black, K., Brown, N., Darpinian, J., Dhabalia, K., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Galliker, M. Y., et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40, 2025a.

Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., et al. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, 2025b. doi: 10.15607/RSS.2025.XXI.010.

Carion, N., Gustafson, L., Hu, Y.-T., Debnath, S., Hu, R., Suris, D., Ryali, C., Alwala, K. V., Khedr, H., Huang,

A., et al. SAM 3: Segment anything with concepts. In *International Conference on Learning Representations (ICLR)*, 2026.

Chen, T., Xu, B., Zhang, C., and Guestrin, C. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Chen, T., Chen, Y., Li, Z., Tang, J., Su, K., Lu, H., Wan, W., Chen, B., Liu, S., Yan, H., et al. RoboDojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026.

Chu, T., Zhai, Y., Yang, J., Tong, S., Xie, S., Schuurmans, D., Le, Q. V., Levine, S., and Ma, Y. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 10818–10838, 2025.

Chu, Z., Barr, E. T., Le Goues, C., O'Hearn, P., Harman, M., Sarro, F., and Ye, H. OpenRUA: Robot-use agents are zero-shot visuomotor policies. *arXiv preprint arXiv:2610.02459*, 2026.

Fan, Y., He, X., Jiang, Y., Wu, D., Wang, J., and Yue, X. CaPtain: A pipeline for post-training code-as-policy robot agents. *arXiv preprint arXiv:2610.40799*, 2026.

Fang, Y., Feng, Y., Jing, D., Liu, J., Yang, Y., Wei, Z., Szafir, D., and Ding, M. When vision overrides language: Evaluating and mitigating counterfactual failures in VLAs. *arXiv preprint arXiv:2602.17659*, 2026.

Fei, S., Wang, S., Shi, J., Dai, Z., Cai, J., Qian, P., Ji, L., He, X., Zhang, S., Fei, Z., Fu, J., Gong, J., and Qiu, X. LIBERO-Plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 38574–38583, 2026.

Gemini Robotics Team, Abeyruwan, S., Ainslie, J., Alayrac, J.-B., Arenas, M. G., Armstrong, T., Balakrishna, A., Baruch, R., Bauza, M., Blokzijl, M., et al. Gemini Robotics: Bringing AI into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z.

Hou, Y. and Zhao, L. LangGap: Diagnosing and closing the language gap in vision-language-action models. *arXiv preprint arXiv:2603.00592*, 2026.

Huang, R., Zhang, X., Chen, H., Li, Y., Wang, Y.-J., Lyu, B., Ye, H., Hong, C., Wang, Y., Su, C., et al. Scaling online exploration for robot agents. *arXiv preprint arXiv:2610.22496*, 2026.

Inco AI. DFlash 2: Keep drafting parallel. https://inco.ai/blog/dflash2/, 2026a.

Inco AI. Splash: A local engine built around the model. https://inco.ai/blog/splash/, 2026b.

Isola, P. Robot-use agents. https://web.mit.edu/phillipi/www/writing/robot-use-agents.html, 2026.

Jacobs, S. A., Tanaka, M., Zhang, C., Zhang, M., Aminabadi, R. Y., Song, S. L., Rajbhandari, S., and He, Y. System optimizations for enabling training of extreme long sequence Transformer models. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 121–130, 2024. doi: 10.1145/3662158.3662806.

Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., and Stoica, I. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations (ICLR)*, 2025.

Khazatsky, A., Pertsch, K., Nair, S., Balakrishna, A., Dasari, S., Karamcheti, S., Nasiriany, S., Srirama, M. K., Chen, L. Y., Ellis, K., et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.120.

Kim, C., Kim, M., Kang, M., Kim, H., and Jung, D. LIBERO-Para: A diagnostic benchmark and metrics for paraphrase robustness in VLA models. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2378–2414, 2026. doi: 10.18653/2026.emnlp-main.101.

Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Sanketi, P., Vuong, Q., et al. OpenVLA: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713, 2025.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165.

Leet, C., Menon, A., Machcha, S., Koenig, S., Chan, T. K., and Chooi, J. The Bitter Lesson comes to robotics. *arXiv preprint arXiv:2610.25287*, 2026.

Liu, B., Zhu, Y., Gao, C., Feng, Y., Liu, Q., Zhu, Y., and Stone, P. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 36, 2023. doi: 10.52202/075280-1939.

Liu, F., Shen, Y., Cheng, A., Mi, W., Tang, C., Liu, C., Xiang, Y., Li, T., Li, Y.-L., and Tang, Y. RoboICL: Embodied in-context learning with GPT-6 Astra. *arXiv preprint arXiv:2609.34261*, 2026a.

Liu, J., Gao, F., Wei, B., Chen, X., Liao, Q., Wu, Y., Yu, C., and Wang, Y. What can RL bring to VLA generalization? An empirical study. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025a.

Liu, X., Xu, B., Bao, W., Li, M., Bao, W., Chen, C., and Shang, Y. AgenticRoboRL: Bringing coding-agent post-training to agentic robotics. *arXiv preprint arXiv:2610.28589*, 2026b.

Liu, Y., Cheng, Y., Yao, J., An, Y., Chen, X., Feng, S., Huang, Y., Shen, S., Zhang, R., Du, K., and Jiang, J. LMCache: An efficient KV cache layer for enterprise-scale LLM inference. *arXiv preprint arXiv:2510.09665*, 2025b.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Lu, R., Wu, Y., Kou, E., Lin, H., Fu, L., Xiao, W., Mandlekar, A., Xu, Y., Shi, G., Goldberg, K., et al. ASPIRE: Agentic /Skills discovery for robotics. *arXiv preprint arXiv:2607.00272*, 2026.

MAA. American Invitational Mathematics Examination (AIME). https://maa.org/maa-invitational-competitions/, 2026.

Marek, M. and Ryabinin, M. Score centering stabilizes off-policy reinforcement learning. *arXiv preprint arXiv:2609.20807*, 2026.

Mittal, M., Roth, P., Tigue, J., Richard, A., Zhang, O., Du, P., Serrano-Muñoz, A., Yao, X., Zurbrügg, R., Rudin, N., et al. Isaac Lab: A GPU-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

ModelScope Team. EvalScope: Evaluation framework for large models. https://github.com/modelscope/evalscope, 2024.

NVIDIA. NVIDIA Model Optimizer. https://github.com/NVIDIA/Model-Optimizer, 2024.

NVIDIA. Cosmos 3: Omnimodal world models for physical AI. *arXiv preprint arXiv:2606.02800*, 2026a.

NVIDIA. KDA: Kernel Design Agents. https://github.com/NVlabs/kda, 2026b.

NVIDIA. RoboLab leaderboard. https://research.nvidia.com/labs/srl/projects/robolab/leaderboard.html, 2026c. Accessed November 9, 2026.

NVIDIA GEAR Team. GR00T N1.6: An improved open foundation model for generalist humanoid robots. https://research.nvidia.com/labs/gear/gr00t-n1_6/, 2025.

OpenAI. Computer-Using Agent. https://openai.com/index/computer-using-agent/, 2025a.

OpenAI. Operator system card. https://cdn.openai.com/operator_system_card.pdf, 2025b. January 23, 2025.

OpenAI. Introducing GPT-6.1 Sol. https://openai.com/index/introducing-gpt-6-1-sol/, 2026a.

OpenAI. GPT-6 Astra: A new generation of intelligence. https://openai.com/index/gpt-6-astra/, 2026b.

Physical Intelligence. openpi. https://github.com/Physical-Intelligence/openpi, 2025. LIBERO results in the README of its LIBERO example, September 8, 2025.

Pikus, B., Saunders, D., Mironenco, M., Mohammadi, S., Mathew, M., Zhu, S., Makkar, D., Huang, X., Li, Q., Narasimhaswamy, S., Cen, S., Xu, D., Buch, S., and Neff, T. Pretraining knows what, simulation teaches how: Compositional generalization for agentic robotics. *arXiv preprint arXiv:2610.35817*, 2026.

Pyatkin, V., Malik, S., Graf, V., Ivison, H., Huang, S., Dasigi, P., Lambert, N., and Hajishirzi, H. Generalizing verifiable instruction following. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 38, 2025.

Qi, Z., Liu, X., Iong, I. L., Lai, H., Sun, X., Sun, J., Yang, X., Yang, Y., Yao, S., Xu, W., Tang, J., and Dong, Y. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025.

Qin, Y., Ye, Y., Fang, J., Wang, H., Liang, S., Tian, S., Zhang, J., Li, J., Li, Y., Huang, S., et al. UI-TARS: Pioneering automated GUI interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

Qwen Team. Qwen3.8-27B. https://huggingface.co/Qwen/Qwen3.8-27B, 2026.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. GPQA: A graduate-level Google-proof Q&A benchmark. In *First Conference on Language Modeling (COLM)*, 2024.

RoboDojo Team. RoboDojo sim leaderboard, agent track. https://robodojo-benchmark.com/leaderboard, 2026. Agent-track entries added September 16 to October 30, 2026; accessed November 8, 2026.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

SenseNova, Joint Research Team, Cai, Z., Cao, Z., Chen, T., Chua, H. C., Dai, L., Deng, H., Deng, H., Diao, H., et al. Has GPT-6-Astra achieved embodied intelligence? *arXiv preprint arXiv:2610.22471*, 2026.

Shah, J., Bikshandi, G., Zhang, Y., Thakkar, V., Ramani, P., and Dao, T. FlashAttention-3: Fast and accurate attention with asynchrony and low-precision. *arXiv preprint arXiv:2407.08608*, 2024.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Shenfeld, I., Pari, J., and Agrawal, P. RL's Razor: Why online reinforcement learning forgets less. In *International Conference on Learning Representations (ICLR)*, 2026.

Sheng, G., Zhang, C., Ye, Z., Wu, X., Zhang, W., Zhang, R., Peng, Y., Lin, H., and Wu, C. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys)*, pp. 1279–1297, 2025. doi: 10.1145/3689031.3696075.

Si, R., Bi, J., Yang, S., Ni, R., Huang, W., Wang, Q., Jiang, S., Zhang, Y., Ruan, F., Zhang, K., Wang, D., Li, X., Feng, H., Dong, Z., and Zhou, D. Fewer tokens, better action: GPT-6 Astra robot agents with 14% higher success rate but 65% fewer tokens. *arXiv preprint arXiv:2610.01939*, 2026.

Wang, H., Zou, H., Song, H., Feng, J., Fang, J., Lu, J., Liu, L., Luo, Q., Liang, S., Huang, S., et al. UI-TARS-2 technical report: Advancing GUI agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992. doi: 10.1007/BF00992696.

xAI. Grok-1.5 Vision preview. https://x.ai/news/grok-1.5v, 2024.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shin, D., Lei, F., et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. doi: 10.52202/079017-1650.

Yan, G., Liu, J., Fan, Y., Cai, L., Liao, M., Zhang, J., and Fox, D. Flex-π: A multi-stream world-action model with compute flexibility. *arXiv preprint arXiv:2608.10860*, 2026.

Yang, J. X., Khetan, A., and Zhang, S. Astranomy: An empirical study of GPT-6 Astra as a native robot policy. https://research.nvidia.com/labs/srl/projects/astranomy/, 2026a. NVIDIA Seattle Robotics Lab technical report (CoRL 2026 LBR), October 14, 2026.

Yang, X., Dagli, R., Zook, A., Hadfield, H., Goyal, A., Birchfield, S., Ramos, F., and Tremblay, J. RoboLab: A high-fidelity simulation benchmark for analysis of task generalist policies. In *Proceedings of Robotics: Science and Systems*, 2026b. doi: 10.15607/RSS.2026.XXII.096.

Yang, Y., Chen, S., Gao, J., and Zhang, H. ORA: Open Robot-Use Agents. https://haoailab.com/blogs/ora/, 2026c.

Ye, S., Ge, Y., Zheng, K., Gao, S., Yu, S., Kurian, G., Indupuru, S., Tan, Y. L., Zhu, C., Xiang, J., et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, J., et al. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025.

Yuan, J., Zuo, Y., Huang, X., Zhang, W., Park, K., Yang, J., Zhang, T., Chen, R., Ouyang, Y., He, B., et al. Democratizing robot-use agents: A unified harness with open models. *arXiv preprint arXiv:2611.01560*, 2026.

Yue, X., Ni, Y., Zhang, K., Zheng, T., Liu, R., Zhang, G., Stevens, S., Jiang, D., Ren, W., Sun, Y., et al. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9556–9567, 2024.

Yue, X., Zheng, T., Ni, Y., Wang, Y., Zhang, K., Tong, S., Sun, Y., Yu, B., Zhang, G., Sun, H., Su, Y., Chen, W., and Neubig, G. MMMU-Pro: A more robust multi-discipline multimodal understanding benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15134–15186, 2025. doi: 10.18653/v1/2025.acl-long.736.

Zadouri, T., Hoehnerbach, M., Shah, J., Liu, T., Thakkar, V., and Dao, T. FlashAttention-4: Algorithm and kernel pipelining co-design for asymmetric hardware scaling. *arXiv preprint arXiv:2603.05451*, 2026.

Zhang, N., Zhu, B., Zhou, S., and Chen, J. Restoring linguistic grounding in VLA models via train-free attention recalibration. *arXiv preprint arXiv:2603.06001*, 2026a.

Zhang, W., Wang, K., Ouyang, Y., Huang, X., Li, L., Su, K., Jin, W., Chai, W., Liang, H., Dou, Z., Chen, Y., and Chen, T. An unexpected robot policy: Early evaluations of GPT-6 Astra on RoboDojo and beyond. *arXiv preprint arXiv:2609.24170*, 2026b.

Zhang, Y., Zhang, H., Gao, F., Li, X., Liu, Z., Nie, Y., Zhu, C., Qiu, J., Yan, Y., Liu, J., et al. Harness VLA: Steering frozen VLAs into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*, 2026c.

Zhao, Y., Gu, A., Varma, R., Luo, L., Huang, C.-C., Xu, M., Wright, L., Shojanazeri, H., Ott, M., Shleifer, S., et al. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *Proceedings of the VLDB Endowment*, 16(12): 3848–3860, 2023. doi: 10.14778/3611540.3611569.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., and Neubig, G. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*, 2024.

Zhou, X., Xu, Y., Tie, G., Chen, Y., Zhang, G., Chu, D., Zhou, P., and Sun, L. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.

Zitkovich, B., Yu, T., Xu, S., Xu, P., Xiao, T., Xia, F., Wu, J., Wohlhart, P., Welker, S., Wahid, A., et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183, 2023.

A. Training details

Table 1 lists the training setup. Training runs on LIBERO (Liu et al., 2023) for 225 steps and then on two RoboDojo (Chen et al., 2026) tasks for 115 steps (Section 4); Section 3.2 describes the RL algorithm and the training infrastructure. During training, the framework executes the JSON object of an action as a tool call to the simulator, and the result of the call is the next observation. The controller that executes an action is specific to each robot: it drives LIBERO's operational-space controller on the Franka and follows Cartesian waypoints with inverse kinematics on RoboDojo's ARX. The reasoning effort is xhigh in every rollout. Earlier rounds remain in the context, and an episode ends when the simulator reports success, when the step limit or the token budget of the context is reached, or when a response contains no valid action.

Table 1. **Training setup.**

Setting	Value
Base model	Qwen3.8-27B (Qwen Team, 2026)
RL algorithm	GRPO (Shao et al., 2024) with DAPO's group filter (Yu et al., 2025) and score centering (Marek & Ryabinin, 2026); a REINFORCE (Williams, 1992) policy loss in bypass mode, without PPO clipping (Schulman et al., 2017)
Batch	128 groups of 8 rollouts per step, counted after the group filter
Optimization	AdamW (Loshchilov & Hutter, 2019) with a learning rate of 10^{-6} and one update per step; the loss is averaged over the generated tokens of a batch (token-mean), without a KL penalty or an entropy bonus
Sampling	Temperature 1.0 over the full distribution
Curriculum	LIBERO: Goal, Spatial and Object for steps 1–148; all four suites from step 149 to step 225 (the LIBERO-only model). RoboDojo: `general_pickup` and `align_blocks` for 115 steps from the LIBERO-only model (steps 226–340)
Max sequence length	256k tokens on LIBERO, 160k on RoboDojo
Reasoning effort	xhigh throughout training
Framework	veRL (Sheng et al., 2025), fully asynchronous
Rollout	veRL + LMCache (Liu et al., 2025b). H100: tensor parallelism 4; 64 rollouts per engine at a 160k cap and 32 at a 256k cap; 95%+ prefix-cache hits. GB300: tensor parallelism 2, without LMCache
Training	H100: FSDP (Zhao et al., 2023) over 8 GPUs with 8-way sequence parallelism (Jacobs et al., 2024) (FSDP8 $\times$ SP8), 32,512 tokens per GPU, FlashAttention-3 (Shah et al., 2024), activation offloading. GB300: FSDP4 $\times$ SP4, 69,632 tokens per GPU, FlashAttention-4 (Zadouri et al., 2026). Gradient checkpointing (Chen et al., 2016) throughout; one FSDP group per node, InfiniBand between nodes
Simulators	LIBERO: rendered on CPU on the rollout nodes. RoboDojo: Isaac (Mittal et al., 2025) containers on separate GPU nodes

Infrastructure. Our training infrastructure, PEAR (a Post-training Engine for Agentic Robotics), is built on veRL (Sheng et al., 2025) and LMCache (Liu et al., 2025b). We train our models on H100, H200 and GB300 GPUs and optimize PEAR for each of them. On H100 GPUs, we serve rollouts with tensor parallelism 4 and train the policy with FSDP (Zhao et al., 2023) over 8 GPUs combined with 8-way sequence parallelism (Jacobs et al., 2024) (FSDP8 $\times$ SP8), so that each GPU holds 32,512 tokens. On GB300 GPUs, we serve rollouts with tensor parallelism 2 and train the policy with FSDP4 $\times$ SP4, with 69,632 tokens per GPU. Each node forms one FSDP group, and nodes communicate over InfiniBand. To fit such long sequences in memory, we use gradient checkpointing (Chen et al., 2016) throughout. On H100 GPUs, we also offload activations and use FlashAttention-3 (Shah et al., 2024), and on GB300 GPUs we use FlashAttention-4 (Zadouri et al., 2026). Each RL step uses 128 groups of 8 rollouts. LIBERO is rendered on the CPUs of the rollout nodes, whereas RoboDojo runs in Isaac (Mittal et al., 2025) containers on separate GPU nodes. Training took about 50,000 H100-equivalent GPU-hours in total.

During RL on LIBERO and RoboDojo, an episode contains about 94k tokens on average (1.8k of prompt and 92k of response), 19.7 rounds and 39 to 59 images. Episodes become shorter as training proceeds, from 119k tokens over 19.9 rounds at the start to 82k tokens over 18.5 rounds at the end. Because each round extends the same context, most of each request is a prefix that has already been processed. On H100 nodes, LMCache offloads the KV cache of these prefixes to CPU memory. We set the number of concurrent rollouts so that the prefix-cache hit rate stays at about 95% or above: each engine runs 64 rollouts concurrently when sequences are capped at 160k tokens and 32 when they are capped at 256k. On GB300 nodes, we serve without LMCache because their host memory is too small to serve as an offload tier. An H100 node with 8 GPUs and a GB300 node with 4 GPUs each generate about 2,000 to 3,000 tokens/s during rollout. Because of some bugs, we did not enable multi-token prediction (MTP) for rollouts, but we will support it later.

RL runs fully asynchronously in veRL, so the policy that generates a rollout lags one or two versions behind the policy being trained. We correct for this mismatch with score centering (Marek & Ryabinin, 2026), which subtracts from each token's score function its expected value under the rollout policy's next-token distribution. We use score centering alone, without truncated importance sampling.

B. LIBERO by suite and effort

Table 2 breaks down the LIBERO results of Figure 3b and Figure 9b by suite, and Figure 10a shows them at max effort. Among the vision-language models, the LIBERO-only model has the highest Spatial and Object scores; on Goal and LIBERO-Long, its scores fall between those of the frontier models at medium and at max effort, and 19% and 33% of its episodes on these two suites end when the context reaches the 256k-token budget.

Table 2. **LIBERO success rate (%) by suite.** The low-effort row is the mean of two evaluation runs. The $\pi_{0.5}$ row is from the openpi repository (Physical Intelligence, 2025). Bold: best VLM in each column, among the max and xhigh rows and among the medium rows.

Model	Effort	Spatial	Object	Goal	Long	Average
Vision-language models						
Qwen3.8-27B	xhigh	1.2	1.8	2.8	0.4	1.6
GPT-6.1 Sol (OpenAI, 2026a)	max	78.8	83.4	76.8	65.8	76.2
GPT-6 Astra (OpenAI, 2026b)	max	82.6	82.2	**80.2**	**77.4**	**80.6**
GPT-6.1 Sol	medium	22.8	**69.2**	58.2	42.2	48.1
GPT-6 Astra	medium	**26.0**	67.0	**59.4**	**44.6**	**49.3**
Vision-language-action model, supervised fine-tuned on LIBERO						
$\pi_{0.5}$	–	98.8	98.2	98.0	92.4	96.9
This work						
Ours (LIBERO only)	xhigh	**85.8**	**97.0**	61.4	53.2	74.4
Ours (LIBERO only)	low	30.4	35.9	18.9	2.5	21.9

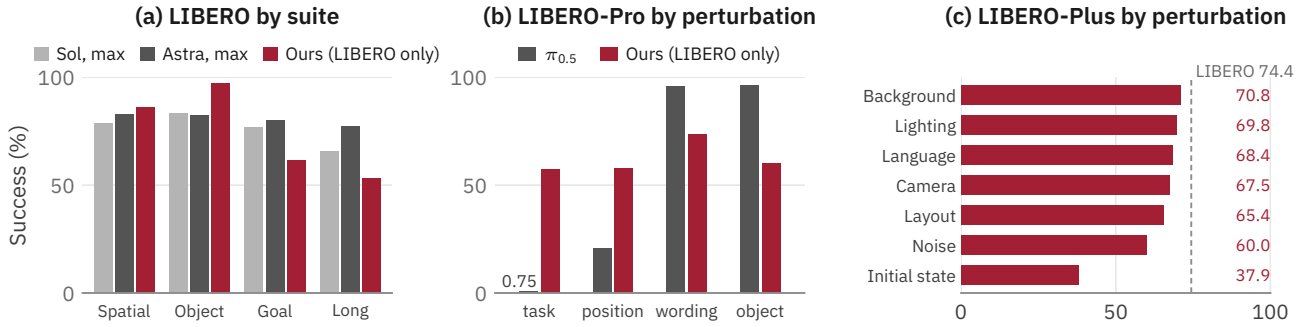


Figure 10. **LIBERO by suite, and LIBERO-Pro and LIBERO-Plus by perturbation.** (a) Success rate (%) on LIBERO by suite, against the frontier VLMs at max effort (Table 2). (b) Success rate (%) on LIBERO-Pro by perturbation type: an unseen task in a training scene, swapped object positions, a reworded instruction and restyled objects (Table 3). (c) The LIBERO-only model on LIBERO-Plus by perturbation dimension (10,030 episodes); the dashed line marks its success rate on LIBERO.

C. Robustness by perturbation and by suite

Table 3 reports LIBERO-Pro (Zhou et al., 2025) results by perturbation type, on which Figure 4b and c and Figure 10b are based. Table 4 reports the success rate of the LIBERO-only model by suite on LIBERO and the two perturbed sets. On LIBERO-Plus (Fei et al., 2026) (Figure 10c), the LIBERO-only model is most sensitive to the robot's initial state, which costs it 36.5 points.

Table 3. **LIBERO-Pro success rate (%) by perturbation type**, both policies trained on LIBERO; 2,000 episodes per type for our model. The $\pi_{0.5}$ row averages the per-suite results of Zhou et al. (2025, Tables 2–5 of the first arXiv version) over the four suites. Bold: best in each column. Task: the same scene with a different task. Position: the objects' positions swapped. Semantic: the instruction reworded. Object: the objects restyled. Mean: over the four types.

Model	Task	Position	Semantic	Object	Mean
$\pi_{0.5}$, supervised fine-tuned on LIBERO	0.75	20.8	**95.8**	**96.0**	53.3
Ours (LIBERO only), xhigh effort	**57.4**	**57.5**	73.3	60.2	**62.1**

Table 4. **The LIBERO-only model by suite, on LIBERO and on the two perturbed sets** (success rate, %); perturbations reduce every suite's success rate by 7–15 points. LIBERO-Pro's numbers include its unperturbed tasks (64.5 on average; 62.1 over the four perturbations alone). LIBERO-Pro results by perturbation type are in Table 3, and LIBERO-Plus results by perturbation dimension in Figure 10c.

Suite	LIBERO	LIBERO-Plus	LIBERO-Pro
Spatial	85.8	75.7	78.9
Object	97.0	82.7	86.4
Goal	61.4	46.8	48.7
Long	53.2	44.5	43.8

D. RoboLab-120 by task difficulty

Table 5 breaks down the RoboLab-120 (Yang et al., 2026b) results of Section 4.3 by task difficulty and adds RoboLab's Score, which gives partial credit for the subtasks that a policy completes. Among the policies in Table 5, the final model achieves the highest success rate on simple and moderate tasks. On complex tasks, Cosmos3-Nano-Policy (NVIDIA, 2026a) leads with 25.3%, whereas the final model reaches 13.2%. The LIBERO-only model improves over the base model mainly on simple and moderate tasks (from 5.2% to 28.6% and from 7.7% to 19.5%).

Table 5. **RoboLab-120 by task difficulty.** Success rate (%) on all 120 tasks and by difficulty (64 simple, 39 moderate and 17 complex tasks, ten episodes each), and Score, RoboLab's graded measure of progress through a task's subtasks, where a success scores 100. The final model is trained with RL on LIBERO and RoboDojo, and its values are the mean of two evaluation runs (2,400 episodes). Other policies: the RoboLab leaderboard (NVIDIA, 2026c, data of 29 October 2026). Bold: best in each column.

Model	All	Simple	Moderate	Complex	Score
Our base model					
Qwen3.8-27B	5.4	5.2	7.7	1.2	12.1
Generalist robot policies					
Cosmos3-Nano-Policy (NVIDIA, 2026a)	36.8	40.6	35.4	**25.3**	**51.9**
$\pi_{0.5}$ (Black et al., 2025a)	28.0	29.7	31.5	13.5	43.4
DreamZero (Ye et al., 2026)	25.7	26.1	30.0	14.1	39.8
GR00T N1.6 (NVIDIA GEAR Team, 2025)	7.2	8.8	7.9	0.0	17.1
This work					
Ours (LIBERO only)	22.1	28.6	19.5	3.5	35.0
Ours	**37.3**	**44.6**	**35.8**	13.2	51.3

E. The full RoboDojo benchmark

Figure 11 shows the average success rate on the full RoboDojo simulation benchmark (Chen et al., 2026), of which Section 4.4 reports two tasks. The benchmark has 42 tasks in five capability dimensions: Generalization, Memory, Long-Horizon, Precision and Open. Its 12 Generalization tasks are evaluated in a standard and a randomized setting, which gives 54 task settings, and we run 10 episodes of each, 540 episodes in total. We follow RoboDojo's official scoring rule: the success rate measures binary task completion, the success rate of a dimension is the mean over its tasks, and the overall success rate is the mean of the five dimensions rather than of all tasks. The leaderboard entries of GPT-5.5 and DeepSeek-Flash are scored with the same rule, so their results can be compared with ours.

The base model succeeds in 1 of the 540 episodes, on `stack_blocks_random`. The final model succeeds in 21 episodes: 8 of 10 on `general_pickup` and 7 of 10 on `align_blocks`, the two tasks used in training, and 3 of 10 on each of `match_and_pick_from_conveyor` and `solve_equation`.

A large part of the failures is due to the length of RoboDojo's tasks. The context window of our model is 256k tokens, and 370 of the final model's 540 episodes (68.5%) end in failure because this window is exhausted. We believe that a stronger foundation model with a longer context window, trained with RURL, or a harness that manages the context could reach higher success rates.

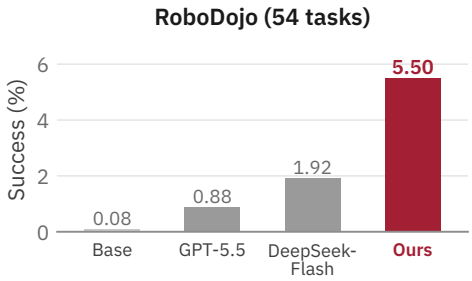


Figure 11. **Average success rate on the full RoboDojo benchmark.** All four models are scored with RoboDojo's official rule, the mean success rate of its five capability dimensions. The base model and the final model are evaluated with 10 episodes per task setting. GPT-5.5 and DeepSeek-Flash are entries of the leaderboard (RoboDojo Team, 2026) from the RoboDojo team's evaluation of language models as policies (Zhang et al., 2026b), with 50 and 10 episodes per task.

F. General benchmarks

Table 6 lists the numbers behind Figure 8, together with the category of each benchmark and its number of questions and samples per question.

Table 6. **General benchmarks.** The final model and the base model on the same questions, with paired 95% intervals and the number of questions (and samples per question). The intervals are from a paired bootstrap over questions with 2,000 resamples. evalscope (ModelScope Team, 2024) 1.13.0, vLLM (Kwon et al., 2023) in BF16, thinking-mode sampling (temperature 1.0, top-p 0.95, top-k 20). Changes and means are computed before rounding. Nine-benchmark means: 81.64 for the base model and 81.49 for the final model. For reference, the LIBERO-only model averages 81.31.

Benchmark	Category	Questions	Base	Ours	Change	95% CI
AIME 2026	math	30×16	98.3	98.5	+0.2	[−1.7, +2.9]
GPQA Diamond	science	198×8	89.6	89.7	+0.1	[−1.0, +1.2]
IFBench	instruction following	300, prompt strict	73.0	72.7	−0.3	[−4.0, +3.3]
LiveCodeBench v6	coding	131×4	88.0	88.4	+0.4	[−2.1, +2.9]
RealWorldQA	visual QA	765	84.1	83.7	−0.4	[−2.4, +1.6]
ERQA	embodied reasoning	400	68.5	66.5	−2.0	[−6.2, +2.2]
MMMU (val)	multimodal	900	80.7	81.0	+0.3	[−1.3, +2.0]
MMMU-Pro standard	multimodal	1730, 10 options	77.3	77.6	+0.3	[−1.1, +1.7]
MMMU-Pro vision	multimodal	1730	75.4	75.4	+0.1	[−1.4, +1.5]

G. Real-robot trials

Table 7 lists every real-robot trial and the cause of each failure. All three models use the same protocol on the same robot at xhigh effort, and none of them was trained on this robot or on any real-robot data. The final model was trained on the two corresponding tasks in RoboDojo with a different robot, whereas the LIBERO-only model and the base model were not. The final model succeeds in 18 of 30 trials, the LIBERO-only model in 7 and the base model in none. The successful trials of the final model take between 7 and 24 rounds, and its failed trials end after between 6 and 25 rounds.

Failure causes. A trial fails when it exceeds 25 rounds or 32k reasoning tokens in a single round (Section 4), or when the model does not complete the task. The base model runs out of rounds in 25 of its 30 trials, knocks the object over in 4 and finishes with the blocks not aligned in 1. The 23 failures of the LIBERO-only model are more varied: it runs out of rounds in 10 trials, knocks the object over in 4, reasons for more than 32k tokens in a single round in 4, stops automatically in an unsafe pose in 2, does not fully lift the object in 2 and picks up the wrong object in 1. The final model shows none of these last four kinds of failure: in its 12 failed trials, it runs out of rounds in 6, finishes with the blocks not aligned in 3 and knocks the object over in 2, and the tennis ball rolls away in 1. No model picks up the air pump, which is knocked over in all 6 of its trials, or the brush, with which 5 of the 6 trials run out of rounds.

Table 7. **Every real-robot trial**, all models at xhigh effort on a dual-arm YAM. Align blocks: two trials in each of five scenes, which differ in the initial layout of the blocks and the set square, all with the instruction "Use the set square to push the three blocks into a straight, aligned row, then release the set square and reset the robot arm." General pickup: two trials per object, "Pick up the ⟨object⟩ by 10 cm." **S**: success within the budget of 25 rounds; F: failure, followed by its cause. *Rounds*: ran out of rounds; *32k tokens*: reasoned for more than 32k tokens in a single round; *toppled*: knocked the object over; *unaligned*: finished with the blocks not aligned; *not lifted*: did not fully lift the object; *wrong object*: picked up the wrong object; *rolled*: the ball rolled away; *unsafe*: stopped automatically in an unsafe pose. Rounds: the number of rounds, at most 25. Context: tokens used by the last of these rounds, out of 260,096.

		Qwen3.8-27B (base)			Ours (LIBERO only)			Ours		
Scene or object	Trial	Result	Rounds	Context	Result	Rounds	Context	Result	Rounds	Context
Align blocks										
Scene 1	1	F: unaligned	12	40.0k	F: unsafe	20	78.2k	**S**	10	54.8k
	2	F: rounds	25	77.7k	**S**	24	57.8k	F: unaligned	9	34.5k
Scene 2	1	F: rounds	25	103.4k	F: 32k tokens	10	99.4k	**S**	14	47.1k
	2	F: rounds	25	60.6k	F: rounds	25	106.0k	**S**	8	26.1k
Scene 3	1	F: rounds	25	69.6k	F: rounds	25	131.8k	**S**	18	80.0k
	2	F: rounds	25	54.6k	F: rounds	25	72.0k	F: rounds	25	119.6k
Scene 4	1	F: rounds	25	72.1k	F: rounds	25	121.0k	F: unaligned	8	44.5k
	2	F: rounds	25	176.4k	F: rounds	25	117.2k	**S**	20	72.0k
Scene 5	1	F: rounds	25	66.2k	F: rounds	25	136.8k	**S**	13	40.1k
	2	F: rounds	25	141.8k	F: rounds	25	98.5k	F: unaligned	11	40.7k
General pickup										
Air pump	1	F: toppled	5	17.6k	F: toppled	6	22.0k	F: toppled	7	28.9k
	2	F: toppled	3	7.4k	F: toppled	13	33.4k	F: toppled	6	20.6k
Brush	1	F: rounds	25	66.0k	F: unsafe	19	86.4k	F: rounds	25	107.5k
	2	F: rounds	25	71.8k	F: rounds	25	126.2k	F: rounds	25	92.1k
Dishwashing liquid	1	F: rounds	25	40.0k	F: 32k tokens	15	79.7k	F: rounds	25	108.3k
	2	F: rounds	25	74.8k	F: 32k tokens	17	131.4k	**S**	13	33.8k
Stapler	1	F: rounds	25	85.0k	F: wrong object	10	59.7k	**S**	13	52.5k
	2	F: rounds	25	103.1k	F: rounds	25	93.4k	F: rounds	25	138.8k
Remote control	1	F: rounds	25	100.0k	F: rounds	25	157.9k	**S**	14	48.9k
	2	F: rounds	25	87.7k	**S**	21	76.5k	**S**	7	23.5k
Rope	1	F: rounds	25	75.4k	**S**	22	135.7k	**S**	9	47.6k
	2	F: rounds	25	38.4k	**S**	7	15.2k	**S**	9	25.6k
Screw	1	F: rounds	25	46.6k	**S**	13	31.8k	**S**	18	82.3k
	2	F: rounds	25	77.6k	**S**	11	33.4k	**S**	24	132.0k
Spray can	1	F: toppled	16	51.9k	F: toppled	6	21.1k	F: rounds	25	129.4k
	2	F: toppled	15	31.3k	F: toppled	8	21.5k	**S**	8	36.8k
Tennis ball	1	F: rounds	25	60.8k	F: 32k tokens	15	141.1k	**S**	22	91.5k
	2	F: rounds	25	90.3k	**S**	19	69.1k	F: rolled	17	81.9k
Turtle plush toy	1	F: rounds	25	54.2k	F: not lifted	11	33.1k	**S**	15	28.2k
	2	F: rounds	25	67.3k	F: not lifted	10	27.3k	**S**	10	23.6k
Successes		0/30			7/30			**18/30**		

H. Prompts

The system prompt consists of a protocol that is the same for every environment, an ENVIRONMENT section that is specific to the benchmark, and the task. The template is given below, where {`environment`} and {`instruction`} stand for the benchmark's ENVIRONMENT section and the task instruction and all other text is given to the model as shown.

```
You are controlling a robot. Each turn you see the robot's current state and its camera images, and
you respond with the next action.

## OBSERVATION

The robot is made of named parts, and each part has named quantities you can set. Every turn begins
with the current value of every such quantity, grouped by part, followed by the camera images:

{"<part>": {"<quantity>": <value>, ...}, ...}

Lengths are in centimeters and angles in radians; any other quantity is a plain number. Nothing
beyond these names is given about what each quantity corresponds to.

## ACTION

The action changes the listed quantities by these amounts:

{"<part>": {"<quantity>": <change>, ...}, ...}

Write the change of every quantity; 0 leaves a quantity as it is.

## EXECUTION

The action ends once every target is reached and the robot is at rest. If something keeps a target
from being reached, the robot stops there and the action ends. Either way, you then receive the next
observation.

## RESPONSE FORMAT

End your response with exactly one fenced json block containing the action, and nothing after it:

```json
<action>
```

ENVIRONMENT

{environment}

TASK

{instruction}
```

LIBERO's ENVIRONMENT section describes its two cameras.

```
- agentview: a fixed camera; the robot's base is at the top of its image, +x points toward the
bottom of the image and +y toward its left.
- wrist: a camera on the gripper; with the gripper pointing straight down at yaw 0, +x points toward
the top of its image and +y toward its right.
- the episode ends by itself as soon as the task is done: while observations keep coming, the task
is not done yet.
```

RoboDojo's ENVIRONMENT section describes the frame of the two arms, the orientation of the grippers and the three cameras. On the real YAM, the prompt follows the same template with this section.

```
Robot and frame:
- left_arm and right_arm share one frame. The origin is midway between the two arms' bases, at the
height they are mounted at; +x points forward, +y to the robot's left, +z up.
- left_arm works on the +y side, right_arm on the -y side.

Gripper orientation:
- Both grippers start pointing forward (roll -1.57, pitch 0, yaw 1.57).
- To grasp from above, first turn the gripper to point down: from the start, a roll change of +1.57
points it straight down.
- Pointing down, yaw turns the gripper about the vertical: at yaw 1.57 its fingers sit on its left
and right and pinch along y; at yaw 0 they sit in front and behind and pinch along x.

Cameras:
- head: fixed above the robot, looking forward and down. Up in the image is +x (farther from the
robot), left is +y.
- left_wrist, right_wrist: one on each gripper, looking along its fingers, which show at the left
and right edges of the image. With the gripper pointing straight down at yaw 1.57, up in the image
is +x and left is +y.

Ending:
- The episode ends by itself as soon as the task is done: while observations keep coming, the task
is not done yet.
- Unless the task is to end up holding something, it is done only once both arms are back near the
values they started with.

Answer format:
- Every action lists all four parts with every quantity; write 0 for whatever should not change. For
example, this action only lowers the right gripper by 5 and closes it:
```json
{"left_arm": {"x": 0, "y": 0, "z": 0, "roll": 0, "pitch": 0, "yaw": 0}, "right_arm": {"x": 0, "y":
0, "z": -5, "roll": 0, "pitch": 0, "yaw": 0}, "left_gripper": {"width": 0}, "right_gripper":
{"width": -1}}
```
```

RoboLab's ENVIRONMENT section describes the frame of the arm, the gripper and the two cameras.

```
Robot and frame:
- The arm's frame has its origin at the robot's base, at the height of the table top: +x points
forward, +y to the robot's left, +z up.
- The arm's position is the point between the gripper's finger pads, about 2 above the fingertips.

Gripper:
- Pointing straight down is roll 3.14 (or -3.14) and pitch 0; pointing down, yaw turns the gripper
about the vertical, and at yaw 0 its fingers pinch along y. Yaws 3.14 apart pinch alike, and the
wrist turns a little less than half a turn either way from yaw 0: of two such yaws, turn to the one
nearer 0.
- Open, the fingers are about 8 cm apart. A close lasts until an open: when it left the fingers wide
apart, closing again changes nothing. Closing also draws the gripper back a little along its axis,
so the fingers close where they were.

Cameras:
- over_shoulder: fixed on the robot's left, looking forward and down across the table; +x points
toward the upper left of the image and +y toward its left.
- wrist: on the gripper, behind the fingers and to their right, looking down past them; with the
gripper pointing down at yaw 0, +x points toward the bottom of the image and +y toward its right,
and the spot straight below the fingers lies right of the image centre, and above it while the
gripper is high.

Ending:
- The episode ends by itself as soon as the task is done.
- The task has a time limit, and every action spends some of it.
```