SKILL-SPACE SHOOTING FOR AUTONOMOUS ROBOT POLICY IMPROVEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Robots deployed in the physical world must be able to improve beyond their initial training as they encounter new situations and failures. For this improvement to scale across tasks, it must make effective use of experience without requiring human demonstration of each correction. Recent agentic systems offer a way to reduce this reliance on human effort by using foundation models to autonomously compose learned behaviors to complete tasks. Yet completing tasks this way does not itself teach a task policy to overcome its own failures; that requires turning these behaviors into learnable corrections for the policy. Our insight is that many such corrections are familiar short behaviors, or skills: they recur across tasks and describe actions that foundation models can reason about from a scene. We introduce *skill-space shooting*, which uses foundation model guidance to explore corrections through these reusable skills and turn successful trials into policy improvement. Real-world experiments show repeated improvement in policies acting autonomously, while skills can also be shared to reduce the teaching needed to improve on new tasks. By making reusable skills a source of corrective supervision, skill-space shooting enables scalable and generalizable policy improvement within and across tasks. Additional results and videos at skill-space-shooting.github.io.

1 INTRODUCTION

Scaling robots to a wider range of real-world tasks requires a way for their policies to keep improving after deployment, without relying on continual human guidance. New objects, changes in the scene, and small execution errors can all expose weaknesses in a task policy. Simply making more attempts does not provide the experience needed for improvement, as a robot can repeat the same mistake without discovering the right fix. While humans can supply such fixes, asking for a new demonstration or correction whenever the robot encounters a failure bottlenecks progress to supervision. We therefore seek an improvement process that generates its own corrective experience and makes initial teaching useful beyond a single task. Such a process would reduce both the supervision needed to improve a task policy, and the additional human effort needed for generalization to new tasks.

Several established approaches offer appealing routes to this type of improvement loop. Reinforcement learning uses rewards to improve through interaction and can concentrate exploration around the behaviors of a pretrained policy (Wagenmaker et al., 2025; Xiao et al., 2026). This makes further learning possible without asking a person to demonstrate every desired action. But sparse rewards on long-horizon tasks can make the required physical exploration too costly. Interactive imitation learning (e.g. DAgger) takes a more direct route: an expert supplies corrective behavior at states the learner actually visits, and the policy trains on those corrections (Ross et al., 2011; Liu et al., 2023). Obtaining these corrections from people, however, imposes a recurring cost as the range of tasks grows. More recently, agentic robotics has connected foundation-model reasoning to executable skills and programs, allowing knowledge about objects and task structure to guide physical behavior (Ichter et al., 2022; Liang et al., 2023; Wang et al., 2023b). These capabilities make model guidance a plausible resource for discovering better behavior, alongside reward-driven exploration and expert supervision. However, coordinating skills to complete an assisted execution does not by itself improve the task policy that needed help. The challenge is thus to use those capabilities to generate corrections at the policy's own failures, so that successful assistance also becomes supervision for learning.

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.

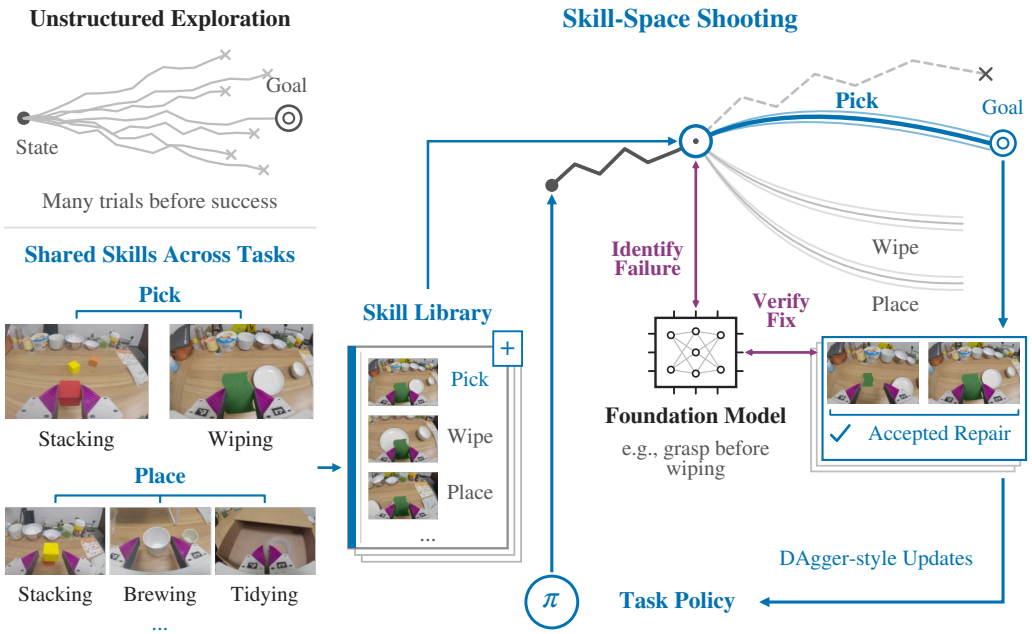


Figure 1: **Skill-space shooting uses reusable skills to search for corrections that improve a task policy.** We treat learned skills as candidate continuations from a state the policy has reached. We employ "shooting" across these skills, with foundation-model knowledge guiding which behavior to try and whether it successfully repairs the failure. Accepted repairs then train the policy for future attempts, closing the improvement loop. Sharing skills across tasks extends this source of corrective supervision while reducing the need for new teaching.

Our key insight is that seemingly disparate long-horizon tasks share many short-horizon behaviors, or skills, that can also supply local corrections. Separately taught skills can provide useful behavior even when the task policy cannot yet perform the entire task reliably. For example, a dedicated picking skill only needs to solve a local grasping problem to supply useful actions within a longer robotic kitting task that a policy might not yet complete end-to-end. Invoking that skill from the policy's current state also addresses the conditions produced by its earlier actions, which a fresh demonstration may not cover. Foundation models can then reason over which of these skills to try, and monitor physical rollouts to test whether the proposed correction works. A successful trial therefore supplies both a way to continue the task and actions that teach the policy how to handle its own difficulties. This turns the skill into a source of autonomous supervision, allowing further trials to address the remaining failures as the policy improves. Because that skill also need not be unique to one task, sharing its training data can also reduce the teaching needed to prepare corrections for another distinct task.

We develop this approach as *skill-space shooting*: autonomous trials of learned skills as candidate continuations of a task policy's rollout. Our central claim is that these skills can be invoked autonomously, stitched into a learner's rollouts to provide corrective supervision, and reused through a shared library to support improvement across tasks. A learned failure detector decides when to invoke a skill, while a foundation model uses semantic knowledge to select the skill and check the repair. Accepted repair segments from completed rollouts then update the task policy through DAgger-style aggregation. Across four real-world tasks, skill-space shooting improves unaided policy performance over successive iterations, including from a policy with no observed initial task successes. Our intervention analysis also shows that the library supplies useful repairs at the policy's failure states. Finally, we demonstrate that sharing skill data reduces the additional human effort needed to adapt existing skills to new motion primitives, and even facilitates improvement on a new task. Together, these results support skill-space shooting as a way to make initial human teaching reusable for autonomous policy improvement within and across tasks.

2 Related Work

We contextualize skill-space shooting against three main bodies of work, ranging from skill composition, to learning from corrections, and finally to autonomous improvement from experience.

Agentic composition of primitives. Agentic robotics connects semantic reasoning to physical behavior by giving models access to executable skills and programs (Liang et al., 2023). Temporally extended behaviors have long served as actions for learning and planning, as formalized by options in Sutton et al. (1999). Recent agentic systems add semantic guidance: SayCan grounds language-based skill selection in learned feasibility, while CoPa grounds manipulation through spatial constraints (Ichter et al., 2022; Huang et al., 2024). Programmatic and hierarchical approaches use such connections between reasoning and action to execute sequences of behaviors (Liang et al., 2023; Wang et al., 2023b; Shi et al., 2025). Skill libraries can also be mined from prior data to support adaptation to new tasks, as in SkillPlug (Ding & Wang, 2026). Execution feedback can further improve these systems: ASPIRE revises programs and stores repair knowledge, while RoboClaw refines the skill policies its agent composes (Lu et al., 2026; Li et al., 2026b). Online recovery further improves execution around a fixed task policy (Zhao et al., 2026; Yi et al., 2026; Hu et al., 2026; Lin et al., 2025; Zhang et al., 2026; Du et al., 2024). Together, these methods make learned behaviors available to model guidance during execution. Our learning problem is to make this assistance improve the complete-task policy itself. We use skill executions as corrective supervision, so the policy can retain their benefit when the composition and repair system is removed.

Learning from corrections. Interactive imitation learning addresses the states an imperfect policy actually reaches, including situations absent from its original demonstrations. The classic approach is DAgger: one collects expert behavior at these failure states and aggregates it to train successive policies (Ross et al., 2011). Later work improves how this supervision is weighted and reused (Mandlekar et al., 2020; Liu et al., 2023; Hu et al., 2025; Physical Intelligence et al., 2025), or enables taking corrections in language (Shi et al., 2024; Belkhale et al., 2024; Zha et al., 2024), or learns when such help is needed (Hoque et al., 2021; Cai et al., 2025; Sa et al., 2026). These approaches make supervision more useful or easier to supply, motivating further automation of the corrective behavior itself. UniIntervene goes further toward automating intervention during RL, combining a temporal value-risk critic with recovery toward targets retrieved from previous interventions (Deng et al., 2026). Our framework retains the focus on learner-visited states but uses demonstrated primitives to supply corrective actions. Initial teaching thus supplies behavior that the robot can invoke autonomously as the task policy improves, without asking a person to demonstrate each new correction.

Autonomous improvement from experience. Autonomously collected experience must somehow provide useful supervision for further improvement (Mirchandani et al., 2024; Belkhale et al., 2023). Existing systems use this experience to retrain policies, fit critics and world models, or learn recovery behaviors (Bousmalis et al., 2023; Giridhar et al., 2026; Yang et al., 2026; Li et al., 2026a). To guide exploration with prior experience, SPiRL learns a state-conditioned skill prior, while SkiLD combines transferable skills with target-task demonstrations (Pertsch et al., 2021; 2022). DSRL likewise reuses a pretrained policy, optimizing its latent-noise inputs (Wagenmaker et al., 2025). These approaches already show that exploration can exploit learned structure. Skill-space shooting builds on this to use language-described skills to make candidate corrections accessible to foundation-model reasoning. These skills and accompanying visual and textual descriptions let the model apply its knowledge of objects, actions, and task structure when choosing which behavior to try. This approach also connects to SOAR, which uses VLMs to collect and assess semantically meaningful experience (Zhou et al., 2024). In our work, semantic guidance instead directs physical repair trials at difficulties encountered by the task policy. The resulting behavior must then improve a policy that can act without assistance.

3 METHOD

Problem setting. We begin with an imperfect task policy trained by behavior cloning on initial task demonstrations, together with a library of learned short-horizon skills. For simplicity, we decompose our tasks into stages with a designated skill, or primitive, for each stage. For example, a drawer task might involve three such primitives: opening the drawer, placing an object inside, and closing it. We collect demonstrations of these stages independently to seed the library, allowing each skill to learn its local behavior without having to solve the complete task. Crucially, this initial teaching need not be repeated from scratch for every task: skills and their training data can come from previous tasks and settings, and our own sharing experiments will show how existing skill data reduces the teaching needed to prepare corrections for a new task. We use demonstrations only to establish this starting

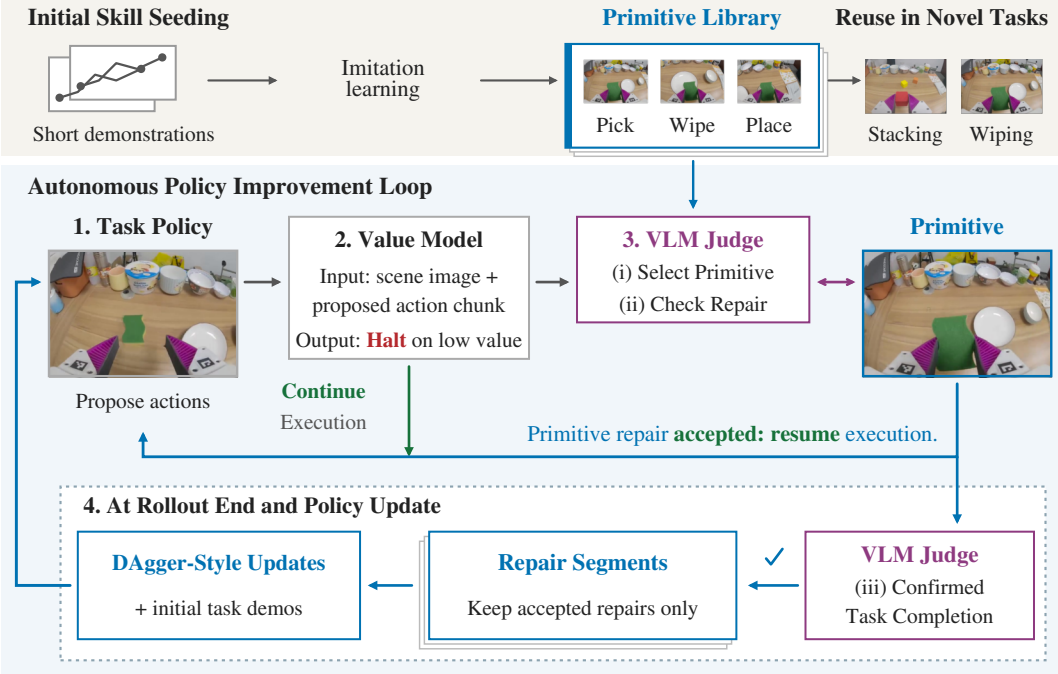


Figure 2: **Trials of reusable skills supply corrective supervision for the task policy.** A value model detects likely failures, and a VLM selects a skill and verifies its repair before the task policy resumes. Accepted repair segments from completed tasks train the next policy, which collects the next round of experience.

library; acquiring skills from human video or autonomous experience offers further work directions for extending it. Given the initial task policy and library, our focused contribution is instead how to turn these skills into corrective experience that improves the policy.

Skill-space shooting uses these learned skills to try corrections where a task policy is likely to fail, then trains the policy on the successful repairs. Our key insight is that a skill can supply a useful local behavior even when the policy cannot yet complete the whole task. To turn that capability into supervision, the loop must recognize when help is needed, test a correction from the learner's current state, and preserve the actions that resolve the difficulty (Fig. 2). The rest of this section describes how the models identify failure to initiate these shooting trials, how repairs become supervision over successive iterations, and how a shared skill library supports improvement on new tasks.

3.1 Deciding When and How to Shoot

Detecting a likely failure. The task policy controls the robot until it needs a correction, so the first decision is when to interrupt its execution. Waiting for an obvious failure can leave little opportunity to recover: an object may appear securely grasped immediately before a motion drops it. The detector must therefore consider what the robot is about to do as well as what it sees. We score the current observation o_t and the policy's proposed H-step action chunk $a_{t:t+H-1}$, and trigger a trial when

$$V_\phi(o_t, a_{t:t+H-1}) < \tau. \tag{1}$$

for some calibrated threshold τ. In our implementation, DINOv2 image features (Oquab et al., 2024) and encoded action tokens are used instead of raw observations and actions, respectively. These are then used to condition a flow-matching head, which we take to be the parameterization of V_ϕ (Lipman et al., 2023). The head predicts the post-chunk robot state together with a scalar value, which determines the trigger condition. We train this value with targets of $+0.5$ for successful execution and -0.5 beginning one action horizon before a recorded failure. These targets encourage the detector to flag an approaching failure while a skill can still correct it. We use $H = 16$ and the midpoint threshold $\tau = 0$ on every task, removing any need for hyperparameter sweeping. Each task's detector is trained on initial labeled episodes and then frozen. This facilitates the fully autonomous improvement loop (beyond initial human demonstrations and subsequent scene resets).

Selecting the skill correction. Once a low value flags a risky motion, we search for a corrective continuation in the space of learned skills. Each skill in the library defines a candidate behavior to try from the state the task policy has reached, such as recovering a grasp, finishing a placement, or repeating a sweep. Searching over these recognizable behaviors allows a foundation model to use its knowledge of objects and task structure to guide which continuation to attempt. We instantiate Gemini 2.5 Pro (Gemini Team, Google, 2025) as this foundation model judge, providing it with the current scene, task instruction, and available skills. For example, if the sponge is still in its bowl when the policy attempts to wipe, the judge can identify the missing grasp and select the corresponding skill. The selected skill then supplies the coordinated actions for a physical trial, whose outcome determines whether the proposed correction works. In this way, foundation-model reasoning guides *how* to shoot, while the value model determines *when* a trial is needed.

In our implementation, recall again that for simplicity each task stage has a designated primitive. This means that in practice identifying the stage that needs recovery or correction also determines exactly which skill to shoot. For scalability, the task and skill descriptions are inputs to a fixed prompt, allowing the same selection procedure to apply across tasks.

3.2 Repairing Rollouts and Improving Policies

Shooting a skill. To test a correction behavior, the selected skill takes control where the task policy stopped. Earlier errors may have changed object positions or contacts, so the skill must respond to the state the task policy actually reached. This is also what makes the resulting actions useful supervision: a fresh demonstration from a clean starting state may never encounter the same difficult failure state. Once the stage is repaired, control returns to the task policy so it can attempt the remaining stages. Returning control lets the task policy use the behaviors it already performs well and encounter further difficulties that may need correction within the same rollout.

Because selecting a skill does not guarantee a successful repair, the loop checks its physical outcome before returning control. The value model monitors the skill during shooting, signaling when to request this check. After c consecutive nonnegative checks, or a limit of M executed actions, the foundation model checks whether the stage was repaired. An accepted repair lets the task policy resume; otherwise, the same skill retries from the current state. We allow at most three retries per rollout and discard it if this budget is exhausted without a repair. We use $c = 8$ on Stack-3, Drawer, and Sweeping, and $c = 16$ on Coffee, whose skills take longer; $M = 100$ throughout.

Learning from repairs. The repaired rollout contains actions for the task policy's failure states, but also the mistakes that led there. Training on the full sequence would teach the policy to imitate both. We therefore retain only accepted, skill-executed repair segments from rollouts the judge confirms as complete. Each segment contains observations and executed actions from invocation to handback, relabeled with the task instruction. We exclude the policy's actions and rejected trials.

This produces DAgger-style supervision (Ross et al., 2011) at states visited by the learner, with demonstrated skills supplying the corrective actions. In particular, training need not wait for the policy's first end-to-end success: a completed rollout can contain stages that only the skills could perform. At iteration k, let $\mathcal{R}_k$ contain the retained repairs and $\mathcal{D}_0$ the initial task demonstrations. We update the policy from its previous checkpoint by imitation learning on

$$\mathcal{D}_{k+1} = \mathcal{D}_0 \cup \bigcup_{j=0}^{k} \mathcal{R}_j. \tag{2}$$

The updated policy then collects the next round of experience. This iteration helps the task policy learn corrections and exposes a wider range of failures. For example, a policy that now completes an early grasp may reach a placement stage that previous attempts never reached. Iteratively collecting with the updated policy brings these newly encountered states, along with remaining failures, to the same skill library. The skills thus provide new supervision as the learner progresses, without needing another round of demonstrations. This process remains autonomous: the models decide when to intervene, which corrections to shoot, and whether to accept them. The process is also scalable and minimizes human involvement: skills and judge prompt remain fixed, as does the value model after its initial calibration, so human input is limited to initial demonstrations and scene resets.

3.3 REUSING THE SKILL LIBRARY

Why skills? For our improvement loop to work, corrections must be meaningful and identifiable choices for the judge, as well as executable behaviors for the robot. A short-horizon skill satisfies both these conditions: its identity (visual and textual descriptions) describes a clear local correction, while its policy supplies the coordinated actions needed to attempt it. For example, selecting a grasp invokes an approach, gripper closure, and lift, with the skill responding to observations along the way. The foundation model can therefore reason about the behavior needed to repair a stage while the skill handles its physical execution. Teaching each skill separately also lets it learn this shorter behavior without having to master the entire task, which is a far easier learning problem.

In our framework, the skill library includes basic primitives like picking and placing, dynamic and periodic primitives like sweeping, or fine-grained and complex primitives like hanging a broom on a nail. Together, these skills illustrate the range of corrections our interface accommodates. We train these skills from handheld UMI demonstrations (Chi et al., 2024); both the skill and task policies follow from a $\pi_{0.5}$-class VLA architecture (Black et al., 2025). This gives them compatible observation and action interfaces as well as control behaviors, allowing seamless handoffs.

Sharing across tasks. Because these behaviors recur across tasks, their training data can help prepare corrections for a new task. For example, grasping a new object may require adapting to its shape while retaining the approach-and-lift behavior learned from earlier objects. To this end, we combine existing skill data with new demonstrations to train the adapted primitive, and show that this reduces the need to teach the entire motion primitive from scratch. The new task has its initial policy and value model, but uses the same procedure to invoke skills, check repairs, and learn from them.

4 EXPERIMENTS

Our experiments answer three questions about skill-space shooting as a source of reusable corrective supervision. (1) Do skill-based corrections improve the task policy over successive iterations? (2) Can the system autonomously invoke skills and verify their repairs reliably? (3) Can a shared skill library transfer across distinct tasks, reducing the human effort needed to prepare new skills?

Tasks and baselines. Our four main tasks require a range of corrective behaviors. **Stack-3** stacks three cubes one on top of another in sequence, while **Drawer** requires opening a drawer, placing a roll of tape inside, and closing it. **Coffee** pours milk, coffee, ice, and adds a stirring stick to a cup. Finally, **Sweeping** retrieves a small handheld broom, sweeps six blocks into a target zone, and rehangs the broom. Together, these tasks test corrections within ordered sequences and during dynamic contact.

The baselines test alternatives to collecting corrections at the task policy's failures. DSRL uses sparse terminal success rewards to learn the latent-noise inputs of the pretrained task policy under a matched collection-time budget for each improvement iteration (Wagenmaker et al., 2025). COP executes the same skills from our library, but under human high level guidance, with up to three rewind-and-retry operations per rollout. HUMAN adds complete-task demonstrations whose total duration matches our newly retained repair data, using the same policy-update procedure. These comparisons test exploration, direct skill composition, and further human teaching, respectively.

Evaluation. We evaluate the learned policies without the value model, judge, or skill library, so the corrective behaviors must be internalized in the task policy. The only exception is COP, which is evaluated as a chain of primitives. Our loop collects 21–81 rollouts per round depending on the task, running four rounds on Stack-3, Drawer, and Sweeping, and six on Coffee. Each method has one run per task, with 30 evaluation rollouts per checkpoint on Stack-3 and 20 on each other task. Stack-3 and Drawer use full-task success; Coffee and Sweeping use mean temporal progress, crediting only consecutive units completed before the first failure. Coffee has four equal units; Sweeping has eight: broom pickup, six blocks, and broom return. Progress reveals improvements to earlier stages while preserving the dependencies between them. We report bootstrap 95% confidence intervals for progress and summarize plateau performance by the best of the final three evaluations. Beyond initial task and skill demonstrations, our loop requires human effort only for scene resets.

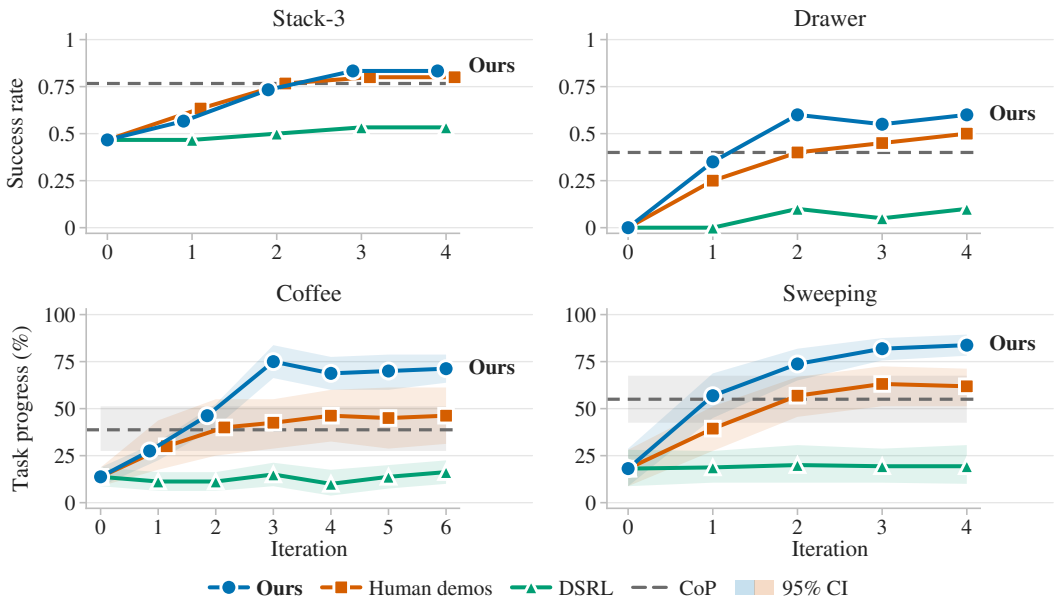


Figure 3: **Skill-space shooting improves policies over iterations, even from zero observed initial successes.** Baselines are latent-space RL (DSRL), additional demonstrations (HUMAN), and fixed primitive composition (COP). Top panels report full-task success; bottom panels show temporal progress with bootstrap 95% intervals.

Method	Stack-3 (N = 30)	Drawer (N = 20)	Coffee (%) (N = 20)	Sweeping (%) (N = 20)
DSRL	0.533	0.100	16.25 ± 6.25	20.00 ± 10.63
COP	0.767	0.400	38.75 ± 12.50	55.00 ± 12.50
HUMAN	0.800	0.500	46.25 ± 15.00	63.13 ± 11.88
OURS	**0.833**	**0.600**	**71.25** ± 7.50	**83.75** ± 5.63

Table 1: **Skill-space shooting reaches the highest observed plateau on every task.** Best of the last three evaluations; COP is evaluated once. Success rates or mean progress ± a bound covering its 95% bootstrap CI.

4.1 Main Results

As shown in Fig. 3, successive updates improve unaided performance on all four tasks without new human demonstrations. Because evaluation uses the task policy alone (i.e. without guidance from the value model or foundation model), these gains show that the benefit of a skill correction extends beyond the rollout in which it was supplied. The task policy retains behavior from its repairs, allowing subsequent collection rounds to address new difficulties or stubborn error modes.

Performance on Drawer is particularly interesting. The initial policy succeeds in none of 20 evaluation attempts, making successful experience difficult to obtain through further policy trials. With sparse terminal rewards, exploration must happen upon a sequence that completes the task before receiving a positive task-level signal. Skill-space shooting narrows this search to learned behaviors that already provide competence at individual stages. Foundation-model guidance selects a skill suited to the current difficulty, allowing the robot to try a coherent correction and continue toward later stages. The robot can therefore construct a successful attempt through targeted local repairs, then learn from the actions that made those repairs work. After one update, unaided success reaches 7/20; DSRL remains at zero after its first update and reaches at most 2/20. This result supports the central motivation for skill-space shooting: existing competence in short behaviors can make successful corrective experience accessible even when exploration through the complete-task policy rarely succeeds.

Comparison to baselines. Skill-space shooting reaches the highest observed plateau on all four tasks, with its clearest advantages on Coffee and Sweeping (Table 1). These tasks demand different kinds of competence: Coffee combines pouring and placement across an extended sequence, whereas

Metric	Stack-3	Drawer	Coffee	Sweeping
Value model trigger accuracy	0.75	0.75	0.83	0.71
Judge stage/skill accuracy	1.00	0.95	0.90	0.90
Skill repair success	0.70	0.70	0.77	0.74
Judge completion accuracy	0.95	0.95	0.90	0.95

Table 2: **Reliable interventions and repairs on all four tasks.** All use one trigger threshold and judge prompt.

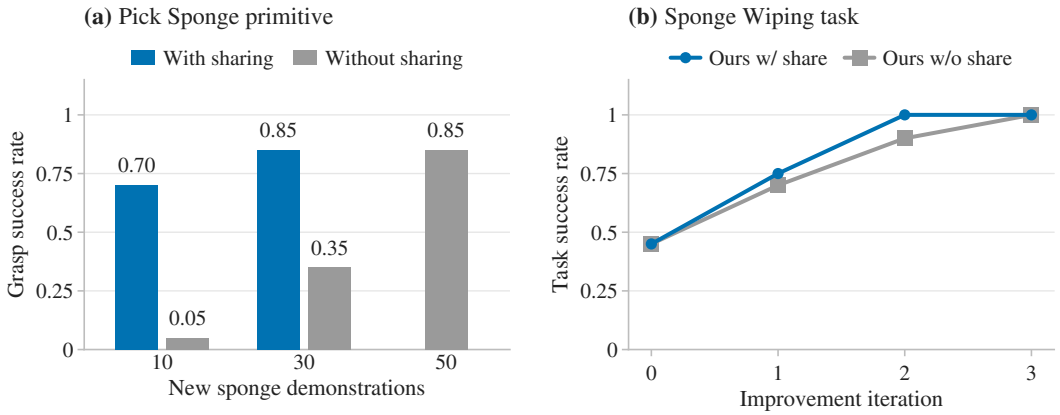


Figure 4: **Sharing skill data reduces new teaching and supports policy improvement on new tasks.** Left: cube-picking data improves sponge grasping at matched new-demonstration counts. Right: shared pick-and-place primitives support Sponge Wiping improvement with fewer new demonstrations than unshared primitives.

Sweeping requires sustained control during contact. The larger gains over additional end-to-end human demonstrations suggest that *where* the new experience is collected matters, particularly when much of a rollout already works in subsequent iterations. The comparisons with DSRL and COP further clarify the role of the skills. Our advantage over DSRL extends beyond Drawer, reaching 55 progress points on Coffee and 63.8 on Sweeping. While DSRL clearly struggles to effectively explore and learn under sparse rewards, shooting in a structured skill space provides coordinated corrective actions directly, reducing the search needed to discover them through terminal rewards. Yet note that simply executing those skills does not achieve the same final performance: our initial policies begin below COP, then exceed it on every task while acting without assistance. The library is therefore useful beyond its performance as a standalone skill chain. By stitching its corrections into the task policy's rollouts and training on those segments, the loop lets the task policy retain its existing competence while acquiring behavior for the states where it struggles. Thus, the resulting policy can outperform both its initial behavior and direct composition of the skills that helped improve it.

4.2 Reliability of Interventions and Repairs

Can the models identify and verify useful corrections? For autonomous trials to supply supervision, the robot must recognize where help is needed and determine whether the attempted correction worked. Table 2 analyzes these decision junctures. The first two rows support using model guidance to initiate targeted skill trials. The value model achieves 71–83% trigger accuracy, while the judge identifies the appropriate stage or primitive with 90–100% accuracy. Together, these results indicate that the system can usually recognize a need for correction and connect it to a relevant skill in the library, without a person deciding when or how to intervene. Skills also successfully repair 70–77% of correctly triggered failure cases. Note that our retry budget gives a plausible skill multiple opportunities to succeed. Inside of three independently but identically distributed attempts, we should be able to repair failures successfully with probability 97.3–98.8%. Finally, the end of the improvement loop still needs to verify the outcome of each trial and checks overall task completion before retaining repair segments. With completion-judgment accuracy of 90–95%, this final check is remarkably robust.

4.3 Sharing and Extending Skills For New Tasks

Can we adapt existing skills to new affordances? Scaling the loop across tasks requires making existing skill data useful beyond their original objects. We test this with **Pick Sponge**, combining

cube-picking demonstrations from Stack-3 with new demonstrations of grasping a deformable sponge (Fig. 4a). In the shared condition, we finetune a checkpoint pretrained on 150 cube-picking demonstrations: 50 for each of three cubes from Stack-3. Without sharing, we finetune directly from the $\pi_{0.5}$-base model. The horizontal axis counts new sponge-picking demonstrations, so each comparison measures the benefit of prior cube experience at the same budget for new demonstrations.

With only 10 new demonstrations, sharing raises grasp success from 5% to 70%. With 30, sharing matches the success obtained from nearly double the number of demonstrations without sharing. Prior skill learning therefore does more than improve performance at a fixed data budget: it reduces the new demonstrations needed to prepare the same skill for an unfamiliar object/motion primitive.

Can the transfer skills from one task to support improvement of a different task? Next, we establish that an adapted skill can then correct another policy's failures and return control successfully on an entirely new task. Our task of choice is **Sponge Wiping**: the robot picks up a sponge, wipes a white bowl, and finally places the sponge in a storage receptacle. This task requires three primitives: pick, wipe, and place. With sharing, all three are finetuned from the cube-picking checkpoint, using 10 new pick demonstrations, 50 wipe demonstrations, and 10 place demonstrations. Without sharing, each primitive is finetuned from $\pi_{0.5}$-base using 50 end-to-end demonstrations. Both conditions therefore receive the same amount of wiping data, with sharing reduces the new teaching for picking and placing. We then use these libraries in otherwise identical repair-and-update loops (Fig. 4b). Despite requiring fewer new demonstrations to prepare its primitives, the shared condition supports improvement from 45% to full observed task success, reaching that level one iteration earlier than the unshared condition. We thus show that prior skill data can also transfer usefully to improving an entirely new task, further motivating the role of skill libraries.

5 Limitations and Future Directions

Acquiring and expanding skills. Repeated improvement with a fixed library shows that skills can remain useful after the policy learns from earlier corrections, but specifying and teaching them still requires initial human effort. Foundation models will increasingly help decompose tasks and identify reusable behaviors (Hansen & Xu, 2026). Persistent failures could guide which skills to acquire next, with each addition evaluated by how many tasks it benefits. Learning skills from human video (Wang et al., 2023a) could further reduce demonstration effort as the library expands.

Sustained autonomous operation. Although the improvement loop is autonomous, people still reset scenes between rollouts. Learned reset behaviors (Li et al., 2026b) or task loops, where completing one task prepares the next (Gupta et al., 2021), could enable unattended operation over days or weeks on open-ended tasks. Combining them with skill-space shooting would allow sustained collection and test whether corrective supervision remains useful as tasks and surroundings change.

Generalizing failure detection. Our failure detector requires per-task training, though skill selection and repair verification share a common procedure. Foundation models already detect execution failures, track task progress, and locate events in robot video (Hansen & Xu, 2026). As their temporal reasoning improves, extending these capabilities to anticipate failure from proposed motions could remove the need for a separate detector. A shared model would then decide both when to shoot and which skill to try, reducing preparation for new tasks.

6 Conclusion

We introduce skill-space shooting to make reusable short-horizon robot behaviors a renewable source of corrective supervision. Foundation-model guidance identifies meaningful behaviors to shoot at a task policy's failures, and monitors their physical rollout to test whether they provide a correction. Training on accepted repair segments from completed rollouts then incorporates those corrections into the task policy. Our experiments show repeated gains that persist without skill assistance, including improvement from a policy with zero observed initial successes. We also prove that sharing skill data reduces the new teaching needed to prepare another task's improvement loop, connecting improvement within a task to reuse across tasks. Skill-space shooting thus provides a foundation for autonomous improvement at scale: reusable skills let robots learn from their own failures and extend that learning across tasks, without requiring humans to teach every possible correction.

Reproducibility Statement

The appendix provides task and primitive definitions, value-model training and stopping rules, baseline configurations, per-iteration results, and intervention counts. The judge uses the `gemini-2.5-pro` model identifier (Gemini Team, Google, 2025); reproducing its decisions depends on access to the provider's model. We will release the implementation.

AI-Use Statement

AI tools assisted with manuscript editing, figure code, and consistency checks. Gemini 2.5 Pro is also a component of the proposed framework, as described in the method.

References

Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Data quality in imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 80375–80395, 2023. doi: 10.52202/075280-3524. 3

Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quan Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. RT-H: Action hierarchies using language. In *Robotics: Science and Systems (RSS)*, 2024. doi: 10.15607/RSS.2024.XX.049. 3

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of the 9th Conference on Robot Learning (CoRL)*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40, 2025. URL https://proceedings.mlr.press/v305/black25a.html. 6

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X. Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, Antoine Laurens, Claudio Fantacci, Valentin Dalibard, Martina Zambelli, Murilo Martins, Rugile Pevceviciute, Michiel Blokzijl, Misha Denil, Nathan Batchelor, Thomas Lampe, Emilio Parisotto, Konrad Żołna, Scott Reed, Sergio Gómez Colmenarejo, Jon Scholz, Abbas Abdolmaleki, Oliver Groth, Jean-Baptiste Regli, Oleg Sushkov, Tom Rothörl, José Enrique Chen, Yusuf Aytar, Dave Barker, Joy Ortiz, Martin Riedmiller, Jost Tobias Springenberg, Raia Hadsell, Francesco Nori, and Nicolas Heess. RoboCat: A self-improving generalist agent for robotic manipulation. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=vsCpILiWHu. 3

Haoyuan Cai, Zhenghao Peng, and Bolei Zhou. Robot-gated interactive imitation learning with adaptive intervention mechanism. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 6243–6256, 2025. URL https://proceedings.mlr.press/v267/cai25e.html. 3

Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal Manipulation Interface: In-the-wild robot teaching without in-the-wild robots. In *Robotics: Science and Systems (RSS)*, 2024. doi: 10.15607/RSS.2024.XX.045. 6, 15

Haoyuan Deng, Yitong Gao, Yudong Lin, Haichao Liu, Zhenyu Wu, and Ziwei Wang. UniIntervene: Agentic intervention for efficient real-world reinforcement learning. *arXiv preprint arXiv:2606.12372*, 2026. 3

Zi-Han Ding and Ziwei Wang. SkillPlug: Unsupervised skill mining for few-shot adaptation in robotic manipulation. *IEEE Robotics and Automation Letters*, 11(8):9511–9518, 2026. doi: 10.1109/LRA.2026.3703998. 3

Maximilian Du, Alexander Khazatsky, Tobias Gerstenberg, and Chelsea Finn. To err is robotic: Rapid value-based trial-and-error during deployment. *arXiv preprint arXiv:2406.15917*, 2024. 3

Gemini Team, Google. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025. 5, 10, 16

Varun Giridhar, Anant Khandelwal, Jeremy A. Collins, Ignat Georgiev, and Animesh Garg. Beyond imitation: Self-improving robot policies via off-policy Q-planning. *arXiv preprint arXiv:2608.21204*, 2026. 3

Abhishek Gupta, Justin Yu, Tony Z. Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, Thomas Devlin, and Sergey Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. In *IEEE International Conference on Robotics and Automation*, 2021. URL https://arxiv.org/abs/2104.11203. 9

Steven Hansen and Peng Xu. Introducing Gemini Robotics ER 2. Google Blog, July 2026. URL https://blog.google/innovation-and-ai/models-and-research/google-deepmind/gemini-robotics-er-2/. Published July 30, 2026. 9

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 *Proceedings of the 5th Conference on Robot Learning (CoRL)*, volume 164 of *Proceedings of Machine Learning Research*, pp. 598–608, 2021. URL https://proceedings.mlr.press/v164/hoque22a.html. 3

Haodi Hu, Chung-Ta Huang, Jing Liu, Ye Wang, Kei Suzuki, Matthew Brand, and Toshiaki Koike-Akino. ReCoVLA: VLM-guided reward compilation for failure recovery in vision-language-action policies. *arXiv preprint arXiv:2606.09630*, 2026. 3

Zheyuan Hu, Robyn Wu, Naveen Enock, Jasmine Li, Riya Kadakia, Zackory Erickson, and Aviral Kumar. RaC: Robot learning for long-horizon tasks by scaling recovery and correction. *arXiv preprint arXiv:2509.07953*, 2025. 3

Haoxu Huang, Fanqi Lin, Yingdong Hu, Shengjie Wang, and Yang Gao. CoPa: General robotic manipulation through spatial constraints of parts with foundation models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9488–9495, 2024. doi: 10.1109/IROS58592.2024.10801352. 3

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T. Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318, 2022. URL https://proceedings.mlr.press/v205/ichter23a.html. 1, 3

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. *arXiv preprint arXiv:2606.18328*, 2026a. 3

Ruiying Li, Yunlang Zhou, YuYao Zhu, Kylin Chen, Jingyuan Wang, Sukai Wang, Kongtao Hu, Minhui Yu, Bowen Jiang, Zhan Su, Jiayao Ma, Xin He, Yongjian Shen, Yang Yang, Guanghui Ren, Maoqing Yao, Wenhao Wang, and Yao Mu. RoboClaw: An agentic framework for scalable long-horizon robotic tasks. *arXiv preprint arXiv:2603.11558*, 2026b. 3, 9

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as Policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500, 2023. doi: 10.1109/ICRA48891.2023.10160591. 1, 3

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. *arXiv preprint arXiv:2510.01642*, 2025. 3

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t. 4, 15

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 (RSS)*, 2023. doi: 10.15607/RSS.2023.XIX.005. 1, 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. 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

Suvir Mirchandani, Suneel Belkhale, Joey Hejna, Evelyn Choi, Md Sazzad Islam, and Dorsa Sadigh. So you think you can scale up autonomous robot data collection? In *Proceedings of the 8th Conference on Robot Learning (CoRL)*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2791–2810, 2024. URL https://proceedings.mlr.press/v270/mirchandani25a.html. 3

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=a68SUt6zFt. 4, 15

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In Jens Kober, Fabio Ramos, and Claire Tomlin (eds.), *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 188–204. PMLR, 16–18 Nov 2021. URL https://proceedings.mlr.press/v155/pertsch21a.html. 3

Karl Pertsch, Youngwoon Lee, Yue Wu, and Joseph J Lim. Demonstration-guided reinforcement learning with learned skills. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 729–739. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/pertsch22a.html. 3

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025. 3

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, 2011. URL https://proceedings.mlr.press/v15/ross11a.html. 1, 3, 5

Inkyu Sa, Konstantin Stulov, and Rajat Bhageria. ValueFormer: A causal transformer value function with stage-aware labels for semi-autonomous vision-language-action policies. *arXiv preprint arXiv:2608.02958*, 2026. 3

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 (RSS)*, 2024. doi: 10.15607/RSS.2024.XX.025. 3

Lucy Xiaoyang Shi, Brian Ichter, Michael Robert Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi Robot: Open-ended instruction following with hierarchical vision-language-action models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 54919–54933, 2025. URL https://proceedings.mlr.press/v267/shi25d.html. 3

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1. URL https://www.sciencedirect.com/science/article/pii/S0004370299000521. 3

Andrew Wagenmaker, Yunchu Zhang, Mitsuhiko Nakamoto, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *Proceedings of the 9th Conference on Robot Learning (CoRL)*, volume 305 of *Proceedings of Machine Learning Research*, pp. 258–282, 2025. URL https://proceedings.mlr.press/v305/wagenmaker25a.html. 1, 3, 6, 18

Chen Wang, Linxi Fan, Jiankai Sun, Ruohan Zhang, Li Fei-Fei, Danfei Xu, Yuke Zhu, and Anima Anandkumar. MimicPlay: Long-horizon imitation learning by watching human play. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 201–221, 2023a. URL https://proceedings.mlr.press/v229/wang23a.html. 9

Renhao Wang, Jiayuan Mao, Joy Hsu, Hang Zhao, Jiajun Wu, and Yang Gao. Programmatically grounded, compositionally generalizable robotic manipulation. In *International Conference on Learning Representations (ICLR)*, 2023b. URL https://openreview.net/forum?id=rZ-wylY5VI. 1, 3

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi Fan, Guanya Shi, and Yuke Zhu. Self-improving vision-language-action models with data generation via residual RL. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=eUGoqrZ6Ea. 1

Jiazhi Yang, Kunyang Lin, Wencong Zhang, Jinwei Li, Tianwei Lin, Longyan Wu, Ya-Qin Zhang, Hao Zhao, Ping Luo, Zhizhong Su, Hongyang Li, Xiangyu Yue, and Li Chen. Self-improving robot policy with compositional world model. In *Robotics: Science and Systems (RSS)*, 2026. doi: 10.15607/RSS.2026.XXII.012. 3

Pengfei Yi, Yingjie Ma, Wenjiang Xu, Yanan Hao, Shuai Gan, Wanting Li, and Shanlin Zhong. Critic in the Loop: A tri-system VLA framework for robust long-horizon manipulation. *arXiv preprint arXiv:2603.05185*, 2026. 3

Lihan Zha, Yuchen Cui, Li-Heng Lin, Minae Kwon, Montserrat Gonzalez Arenas, Andy Zeng, Fei Xia, and Dorsa Sadigh. Distilling and retrieving generalizable knowledge for robot manipulation via language corrections. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15172–15179, 2024. doi: 10.1109/ICRA57147.2024.10610455. 3

Fan Zhang, Seongbin Park, Baharan Mirzasoleiman, Shahriar Talebi, and Nader Sehatbakhsh. ProbeAct: Probe-guided training-free failure recovery in vision-language-action models. *arXiv preprint arXiv:2606.09740*, 2026. 3

Ganlong Zhao, Zijia Tang, Xingping Chen, Zhanghui Kuang, Ye Tian, and Guanbin Li. FLARE: A failure-aware framework for autonomous correction and recovery in visual-language robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22391–22401, 2026.

URL https://openaccess.thecvf.com/content/CVPR2026/html/Zhao_FLARE_A_Failure-Aware_Framework_for_Autonomous_Correction_and_Recovery_in_CVPR_2026_paper.html. 3

Zhiyuan Zhou, Pranav Atreya, Abraham Lee, Homer Rich Walke, Oier Mees, and Sergey Levine. Autonomous improvement of instruction following skills via foundation models. In *Proceedings of the 8th Conference on Robot Learning (CoRL)*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4805–4825, 2024. URL https://proceedings.mlr.press/v270/zhou25b.html. 3

A TASKS AND DATA COLLECTION

We collect task and primitive demonstrations with handheld UMI (Chi et al., 2024). Task demonstrations cover full executions; primitive demonstrations teach single behaviors used during failures.

A.1 TASK AND PRIMITIVE INVENTORY

The tasks span different forms of manipulation. Stack-3 tests precise placement, Drawer combines pulling with object transfer, Coffee requires pouring and insertion, and Sweeping adds repeated contact with moving objects. The library provides primitives for the stages in Table 3. These semantic primitives control the whole action needed to complete a stage, rather than fixed Cartesian motions.

Task	Stages	Metric	N	K
Stack-3	Pick red; place red on orange; pick yellow; place yellow on red	Success	30	4
Drawer	Open drawer; put tape inside; close drawer	Success	20	4
Coffee	Pour milk; pour coffee; add ice; put stick in cup	Progress	20	6
Sweeping	Pick broom; sweep six blocks into the target; hang broom	Progress	20	4

Table 3: **The task suite spans placement, articulation, pouring, and sweeping.** Each stage has a primitive. n counts policy-only evaluation rollouts per checkpoint; K counts improvement rounds.

A.2 EVALUATION METRICS

Stack-3 and Drawer count only complete-task successes. Coffee and Sweeping also report partial progress, since earlier stages can improve before the whole task succeeds. Coffee awards 25% for each of its four stages. Sweeping awards 12.5% for each of eight units: picking up the broom, moving each of six blocks into the target, and returning the broom. Both metrics credit progress only up to the first failed stage or unit. We average these scores over 20 evaluation rollouts.

A.3 PRIMITIVE TRANSFER AND THE NEW WIPING TASK

Pick Sponge evaluates an individual grasp, transferring cube-picking behavior to a deformable object. Its source data contain 50 grasp demonstrations for each of the red, yellow, and orange cubes. Sponge Wiping then uses adapted pick and place primitives in a complete task: pick up the sponge, wipe the bowl, and place the sponge in a receiving bowl. The sharing comparison changes the data used to train the pick and place primitives, while both conditions use the same policy-improvement procedure.

B METHOD DETAILS

B.1 VALUE MODEL

The value model has about 300M parameters and receives the current image and a proposed $H = 16$-step action chunk. A DINOv2 encoder (Oquab et al., 2024) produces image tokens, which are combined with encoded action tokens to condition a flow-matching head (Lipman et al., 2023). The output decodes into the predicted post-chunk robot state and a scalar value. Positive and negative training targets are $+0.5$ and -0.5, respectively; the intervention threshold is $\tau = 0$ on every task.

Successful episodes receive positive targets throughout. In a failed episode with failure frame F, negative targets begin one action horizon before F. This lookahead trains the detector to flag a motion before the recorded failure, without task-specific hand-written failure rules.

Every task uses the same initial calibration procedure. We first train on task data, then incorporate the first collection round and retrain the value model from scratch on the combined data. The model is then frozen; its threshold remains zero throughout. Table 4 gives first-round collection sizes.

At execution time, the robot skips the stale front portion of a newly inferred action chunk. When the next chunk arrives, it replaces any unexecuted commands from the previous one. Repair segments therefore store the actions actually executed, rather than every action the policy proposed.

Task	Stack-3	Drawer	Coffee	Sweeping
First-round calibration rollouts	78	42	24	29

Table 4: **One recalibration per value model.** We add first-round rollouts to initial data, then freeze the model.

B.2 Primitive selection and acceptance

The framework uses Gemini 2.5 Pro (`gemini-2.5-pro`) as its judge (Gemini Team, Google, 2025). At an intervention, the judge receives the current scene, the task instruction, and the available primitive names. It selects a primitive for the unfinished stage. Instructions and primitive names are inputs to a shared prompt, so changing tasks does not require restructuring it.

The judge also checks whether a primitive repaired its stage and whether the rollout completed the task. A repair is eligible for training only if both checks pass. This separates local progress from task completion: repairing one stage alone does not make a successful training rollout.

B.3 Returning control and retrying

While the primitive runs, the value model continues to score its proposed actions. A trial ends after c consecutive nonnegative value checks or at a limit of $M = 100$ executed actions. We use $c = 8$ on Stack-3, Drawer, and Sweeping and $c = 16$ on Coffee, whose primitives take longer. The acceptance check then determines whether the task policy can resume.

After a rejected repair, the same primitive can retry from the current state, with a budget of three retries per rollout. If the stage remains unrepaired when that budget is exhausted, the rollout is discarded. When a retry succeeds, only that accepted trial contributes a repair segment.

B.4 Policy updates

After collection, we aggregate accepted repair segments from completed tasks with the initial task demonstrations. Each segment contains observations and executed actions from primitive start to hand-back, relabeled with the task instruction. We exclude rejected trials, incomplete rollouts, and the task policy's own actions to avoid imitating the suboptimal behavior preceding a repair.

Each policy update starts from the previous iteration's checkpoint. We evaluate the updated policy with the value model, judge, and primitives disabled, then collect the next round. Evaluation thus measures learning from past repairs rather than performance with continued intervention.

C Per-iteration results and intervention outcomes

C.1 Per-iteration values

Each evaluation uses 30 rollouts on Stack-3 and 20 on Drawer, Coffee and Sweeping. We report 95% Wilson intervals for binary success and percentile bootstrap intervals for mean progress, using 10,000 resamples of each evaluation batch. For the ± notation in Table 1, we use the larger distance from the mean to either bootstrap endpoint, rounded upward to two decimal places. This bound covers the original interval; exact endpoints appear below. The plateau summary uses the best of the final three evaluations: iterations 2–4 on Stack-3, Drawer and Sweeping, and 4–6 on Coffee. This excludes Coffee's iteration 3 peak; averaging these three points preserves the method ordering on every task.

C.2 Intervention outcomes

The record separates erroneous triggers (E), successful repairs after a correct trigger (S), and unsuccessful repairs after a correct trigger (F). We compute the trigger and repair rows of Table 2 as

$$\text{Trigger accuracy} = \frac{S+F}{E+S+F}, \qquad \text{Repair success} = \frac{S}{S+F}.$$

The first rate describes the recorded intervention cases; the second measures whether the invoked primitive repaired the stage. Rollouts completed without a primitive are recorded separately as C.

Stack-3	it. 0	it. 1	it. 2	it. 3	it. 4
HUMAN	14/30 [0.30, 0.64]	19/30 [0.46, 0.78]	23/30 [0.59, 0.88]	24/30 [0.63, 0.90]	24/30 [0.63, 0.90]
COP	23/30 [0.59, 0.88] (single batch)				
DSRL	14/30 [0.30, 0.64]	14/30 [0.30, 0.64]	15/30 [0.33, 0.67]	16/30 [0.36, 0.70]	16/30 [0.36, 0.70]
OURS	14/30 [0.30, 0.64]	17/30 [0.39, 0.73]	22/30 [0.56, 0.86]	25/30 [0.66, 0.93]	25/30 [0.66, 0.93]

Drawer	it. 0	it. 1	it. 2	it. 3	it. 4
HUMAN	0/20 [0.00, 0.16]	5/20 [0.11, 0.47]	8/20 [0.22, 0.61]	9/20 [0.26, 0.66]	10/20 [0.30, 0.70]
COP	8/20 [0.22, 0.61] (single batch)				
DSRL	0/20 [0.00, 0.16]	0/20 [0.00, 0.16]	2/20 [0.03, 0.30]	1/20 [0.01, 0.24]	2/20 [0.03, 0.30]
OURS	0/20 [0.00, 0.16]	7/20 [0.18, 0.57]	12/20 [0.39, 0.78]	11/20 [0.34, 0.74]	12/20 [0.39, 0.78]

Coffee	it. 0	it. 1	it. 2	it. 3	it. 4	it. 5	it. 6
HUMAN	13.75 [8.75, 18.75]	30.00 [17.50, 43.75]	40.00 [25.00, 55.00]	42.50 [28.75, 55.00]	46.25 [32.50, 60.00]	45.00 [28.75, 61.25]	46.25 [31.25, 61.25]
COP	38.75 [27.50, 51.25] (single batch)						
DSRL	13.75 [8.75, 18.75]	11.25 [6.25, 16.25]	11.25 [6.25, 16.25]	15.00 [8.75, 21.25]	10.00 [3.75, 17.50]	13.75 [7.50, 20.00]	16.25 [10.00, 22.50]
OURS	13.75 [8.75, 18.75]	27.50 [22.50, 32.50]	46.25 [40.00, 52.50]	75.00 [66.25, 83.75]	68.75 [60.00, 77.50]	70.00 [60.00, 78.75]	71.25 [63.75, 78.75]

Sweeping	it. 0	it. 1	it. 2	it. 3	it. 4
HUMAN	18.125 [8.75, 28.125]	39.375 [27.50, 51.25]	56.875 [45.625, 66.875]	63.125 [51.25, 72.50]	61.875 [51.25, 71.25]
COP	55.00 [42.50, 67.50] (single batch)				
DSRL	18.125 [8.75, 28.125]	18.75 [10.625, 27.50]	20.00 [10.625, 30.625]	19.375 [10.625, 28.75]	19.375 [10.00, 30.625]
OURS	18.125 [8.75, 28.75]	56.875 [45.00, 68.75]	73.75 [64.984, 81.875]	81.875 [75.625, 87.50]	83.75 [78.125, 89.375]

Table 5: **Per-iteration policy performance.** Successes (Stack-3, Drawer) have 95% Wilson intervals; mean progress (Coffee, Sweeping) has 95% bootstrap intervals. Shading marks the final three evaluations in Table 1.

Round k uses policy π_k to collect data for update $k + 1$. For every task, the value model is calibrated once using the first collection round and then remains fixed for subsequent rounds.

The judge rows summarize author assessments of stage/primitive selection and task completion. Selection accuracies are 1.00, 0.95, 0.90, and 0.90 for Stack-3, Drawer, Coffee, and Sweeping; completion accuracies of 0.95, 0.95, 0.90, and 0.95 come from the authors' manual review of rollouts. These informal estimates differ from the trigger and repair rates computed from collection counts.

The source categories differ from the recorded rollout totals in three rounds: Stack-3 round 0 sums to 82 rather than 78, Drawer round 2 to 45 rather than 46, and Coffee round 4 to 23 rather than 25. We retain the recorded counts and calculate both rates from their stated category totals. The no-primitive completions describe collection rollouts, separate from the policy-only evaluation batches in Table 5.

Task	Round	Rollouts	C	E	S	F
Stack-3	0	78	26	20	28	8
	1	80	36	9	25	10
	2	79	48	6	16	9
	3	81	52	5	15	9
	Total	318	162	40	84	36
Drawer	0	42	0	15	20	7
	1	45	13	5	20	7
	2	46	22	5	13	5
	3	44	16	6	13	9
	Total	177	51	31	66	28
Coffee	0	24	0	6	15	3
	1	21	0	3	14	4
	2	22	0	4	15	3
	3	26	6	3	12	5
	4	25	4	3	12	4
	5	25	5	2	13	5
	Total	143	15	21	81	24
Sweeping	0	29	0	10	16	3
	1	28	2	7	14	5
	2	30	4	7	13	6
	3	29	6	6	12	5
	Total	116	12	30	55	19

Table 6: **Intervention outcomes across every task and collection round.** C counts completions without a primitive; E, S and F are defined above. Total rows provide the pooled counts used in Table 2.

D BASELINES AND HUMAN INPUT

D.1 BASELINE CONFIGURATIONS

Additional demonstrations. HUMAN starts from the same initial task policy and adds complete-task UMI demonstrations at each iteration. Their total duration matches the new repair segments retained by our method that round. It uses the same policy-update procedure, testing whether complete executions provide supervision as useful as corrections at the policy's failure states.

Primitive composition. COP uses the same primitive library, with a human choosing the primitive for each stage. When failure is imminent or has occurred, the operator can rewind to a state a few seconds earlier and retry, at most three times per rollout. The primitives are fixed, so this baseline is evaluated once rather than retrained across iterations.

Real-world reinforcement learning. DSRL optimizes the latent-noise inputs of the pretrained flow policy (Wagenmaker et al., 2025). Each collection round receives the same time budget as our method. A person labels each rollout as a success or failure, giving a terminal reward of one or zero. We use `query_freq`=2, `multi_grad_step`=30, and discount 0.995; other settings follow the original real-world configuration. This yields about 25 rollouts and 50,000–60,000 gradient updates per hour.

D.2 HUMAN INPUT

All methods require scene resets. Beyond these resets, HUMAN collects new demonstrations each round, COP requires primitive selection and recall decisions during execution, and DSRL requires outcome labels for its reward. For our method, human input is limited to initial task-policy demonstrations, initial primitive demonstrations, and scene resets. The value model determines when to intervene; the VLM selects a primitive, checks the repair, and verifies task completion. The one-time value-model calibration uses the first collection round on each task; subsequent rounds reuse the frozen model. No human selection, interruption, repair verification, or completion judgment enters the loop. These comparisons describe human input but do not measure total human time.

E SHARING

E.1 SHARING OPERATION

We first test whether a primitive benefits from data collected for other objects, then use the adapted primitives to improve a new task policy. For Pick Sponge, the shared source contains 50 grasping demonstrations each for red, yellow and orange cubes, or 150 demonstrations in total. We train the shared primitive on the cube data together with 10 or 30 new sponge demonstrations, and compare it with primitives trained on 10, 30 or 50 sponge demonstrations without cube data. The matched 10- and 30-demonstration conditions isolate the contribution of prior experience; the 50-demonstration condition provides a higher-data comparison without sharing.

E.2 GRASP SUCCESS WITH AND WITHOUT SHARING

Sharing improves grasp success at both matched demonstration counts (Table 7). With 30 new demonstrations, the shared primitive matches the observed success rate of the unshared primitive trained with 50. This saves new sponge demonstrations; the cube data came from the existing library.

	With sharing		Without sharing		
Primitive	10 demos	30 demos	10 demos	30 demos	50 demos
Pick Sponge	0.70	0.85	0.05	0.35	0.85

Table 7: **Shared grasping experience reduces new-object demonstrations.** Entries give Pick Sponge grasp success; demonstration counts refer to new sponge data.

E.3 SPONGE WIPING TASK

Sponge Wiping requires picking up a sponge, wiping a white bowl and placing the sponge in another bowl. Both conditions run our improvement loop, using pick and place primitives seeded with 10 sponge demonstrations per primitive when sharing cube data, or 50 without sharing. These demonstrations initialize the primitives before improvement begins. Both task policies then learn from accepted repair segments under the same loop, and evaluation runs the task policy alone.

Iteration	0	1	2	3
OURS with sharing	0.45	0.75	1.00	1.00
OURS without sharing	0.45	0.70	0.90	1.00

Table 8: **Shared primitives support faster Sponge Wiping improvement.** Full-task success by iteration.

With fewer new demonstrations for its primitives, the shared condition supports the same improvement from 45% to full observed task success, one update earlier in this run. Sharing therefore reduces the teaching needed for a new task's library while preserving policy gains from iterative correction.