The Right Kind of Wrong: Counterfactual Data for Reward Learning in Manipulation

Gokul Kannan Thiruvenkadam, Vishal M. Patel†, Homanga Bharadhwaj†

Johns Hopkins University †Equal advising

rightkindofwrong.github.io

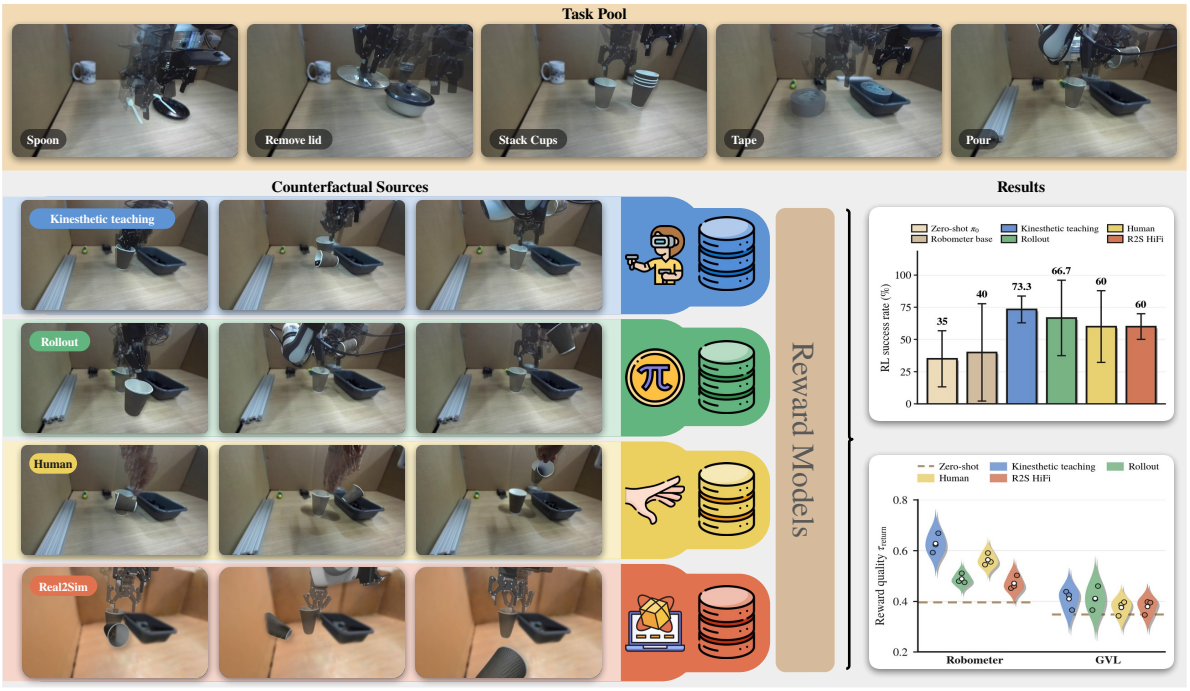


Figure 1: Overview of our analysis. We consider counterfactual trajectories demonstrating failures obtained with kinesthetic teaching, pre-trained policy rollouts, human videos, and real2sim across five different manipulation tasks. We compare how the source of the counterfactuals and the scale affect learned reward models and subsequent robot-policy post-training. Video results are in the website rightkindofwrong.github.io

Abstract. Reinforcement learning (RL) can improve robot policies beyond imitation learning, but it requires rewards that capture task progress and failure. Learning such rewards requires not only successful demonstrations, but also informative counterfactual failures that can guide the model on behaviors to penalize. However, large-scale robot datasets are dominated by successful behavior, and collecting failures on real hardware is expensive. This raises a practical question: given a limited data-collection budget, which sources of counterfactual experience are worth investing in? We compare four sources of failure data: kinesthetic-teaching failures, policy rollouts, human video, and real2sim rollouts, across five real-world manipulation tasks. By keeping the task suite, successful demonstrations, and the evaluation setup fixed, we study the effect of these sources through LoRA adaptation of a reward model and few-shot prompting of a frozen model in two different data scale scenarios. We evaluate the resulting reward models on a custom held-out benchmark of 250 human-annotated policy rollouts, measuring trajectory ranking and success detection. Finally, we use selected reward models to guide RL post-training of π_0 and test whether improvements in offline reward quality translate into better robot behavior. In sum, our study examines whether inexpensive counterfactual sources can match or surpass smaller amounts of faithful real-robot failure data when scaled, for improving rewards models in robot manipulation.

1 Introduction

Reinforcement learning (RL) can learn new behaviors from scratch that are hard to teleoperate [41, 44] or improve large-scale robot policies trained via behavior cloning [10, 16, 20, 38]. Imitation-learned policies are a good starting point, but they still suffer from classic problems like compounding errors and covariate shift [30, 32, 33], and RL can help with these. Rewards are one of the main components for RL to perform well. Classically, there is a lot of engineering involved in shaping rewards [12] to get better performance in RL. A well-calibrated reward needs to track progress for different varieties of rollouts, from successes to suboptimal ones to failures.

A growing body of work therefore studies learned reward models for robot manipulation. These models are typically trained on large-scale robot datasets [14, 26], which consist predominantly of expert demonstrations and contain little to no failure data [2]. Existing methods thus need to obtain counterfactual supervision from several alternative sources. One category of works constructs synthetic counterfactuals from successful trajectories through augmentations such as reversing video segments, early stopping or stalling, and language relabeling [15, 37, 43]. These can be viewed as a form of post-hoc relabeling to synthesize failures from successes. Another viable approach is to use real failures collected through deliberate teleoperation or policy rollouts on hardware [17]. An alternative strategy is to avoid task-specific counterfactual data altogether, estimating rewards zero-shot from frozen vision-language models [7, 23]. Recent work argues that reward models need bad behavior data to be reliable [36], but which kind is worth paying attention to? Synthetic failures are cheap but indirect, while real failures give a direct signal to learn from but are expensive.

This leaves a practical question that we set out to study: **What is the right kind of wrong for learning rewards?** We keep the tasks, reward models, training procedure, evaluation, and RL setup fixed, while varying the counterfactual data sources and how much data there is per source. Our most expensive sources are kinesthetic teaching (KT), where an operator physically guides the robot arm through a failure and the motion is replayed on camera, and zero-shot policy rollouts (RO), where we have no control over the behavior. The cheaper ones are deliberately collected failures in human video (HV), which creates an embodiment gap, and scripted failures in lifted real-to-sim environments (real2sim, R2S), which creates a visual gap. Since the cheaper sources are easier to collect in large amounts, we compare all four at the same scale and then scale the cheaper ones according to their cost.

As outlined in Figure 1, we study the counterfactual data sources across two reward models: one trainable, Robometer [17], and one few-shot promptable method, GVL [23]. We first test the adapted rewards on the real robot, using DSRL [38] to post-train π_0 [4] on three tasks of increasing difficulty. Since running RL for every condition and source is not practical, we also build a held-out benchmark of 250 annotated rollouts to study how source, scale, and the visual fidelity of simulation change the learned reward itself. Surprisingly, the cheaper sources hold up well in RL despite their domain gaps, while more data and better offline scores do not always mean better reward signals.

Our main contributions

- **A controlled study of counterfactual sources.** We collect data from four sources across five real manipulation tasks and build a held-out, 250-episode cross-policy benchmark with human-annotated partial credits.
- **Cheap counterfactuals can also help real-robot RL.** With DSRL on three tasks, we find that every source improves over the base reward. Cheaper sources are competitive with the expensive ones despite their domain gaps.
- **What the offline benchmark shows.** Offline experiments show that the source matters more for trainable reward models than for prompted ones, that scaling helps only up to a point for cheap sources, and that offline metrics only loosely predict RL success.

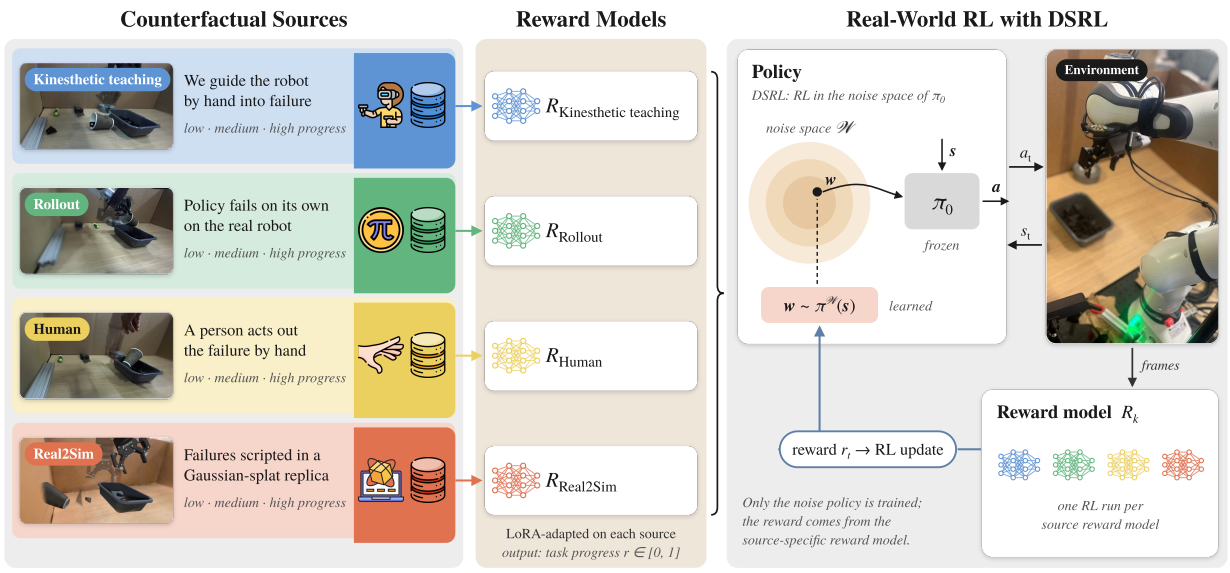


Figure 2: Setup overview. Left: each counterfactual source is used to adapt its own Robometer reward model with LoRA, which predicts task progress in $[0, 1]$. Right: DSRL trains only a noise-space policy on top of the frozen π_0, and the reward for each RL run comes from one source-specific reward model.

2 Related Work

Reward learning from demonstrations. Learning rewards or value from demonstrations is a long-studied alternative to human-engineered rewards. Inverse reinforcement learning infers a reward that explains expert behavior [1, 46], ranking-based methods extrapolate beyond the demonstrator [5], and representations such as VIP [21] and LIV [22] measure progress as distance to a goal. These methods benefit from the growing supply of datasets dominated by expert behavior [14, 26], but successes alone cannot teach a reward to reliably score partial completions or failures. ReWiND [43] addresses this by reversing successful videos to synthesize undo behavior, a common failure seen in RL rollouts, which lets it learn decreasing progress. However, such augmentation-based failures do not cover the plausible failures a robot actually makes. We instead ask which sources of more realistic failures are worth collecting.

Reward learning from failures. A second line of work supervises rewards with mixed sources of data. Recent methods like RoboReward [15], Robometer [17], Robo-Dopamine [35], VLAC [42], and LRM [39] train VLM-based reward or progress models with mixed data sources. We mainly use Robometer, which combines frame-level progress with trajectory-level preferences over successful, suboptimal, and failed rollouts across many embodiments. Failures also appear as first-class data in vision-language-action models [18], autonomous skill improvement [45], and language-guided recovery [8], and a recent position paper [36] argues for the need for reliable bad behavior data. These works establish that failures are important; we study what kind of failures to invest in.

Rewards from foundation models. A third line of work uses large pretrained vision-language models as general-purpose reward or value estimators. GVL [23] treats a VLM as an in-context value learner that predicts progress over shuffled video frames, enabling zero-shot and few-shot progress estimation across tasks and embodiments. TOPReward [7] instead reads progress from an open-source VLM's token probabilities, and RoboCLIP [34] and RARM [40] score rollouts by their video-language similarity with the reference demonstration. These approaches need no task-specific training, or at times no training at all, inheriting from broad visual priors, but their scoring depends on the reasoning capabilities of the foundation model they use [3, 29]. We include few-shot GVL as one of our two reward models to test whether counterfactuals still help when they are only seen in context.

Table 1: Counterfactual sources and their cost-alignment trade-off. *Collection* is the operator time to produce one trajectory; *annotation* is the time to assign partial-credit labels. Robot successes are pooled into a shared set (Section 3.3).

Source	Embod.	Real vis.	On-policy	Scripted	Per-episode cost	Annotation	Data Collected / task
Kinesthetic	✓	✓	–	✓	$t_{\text{collect}} + t_{\text{replay}}$ (kinesthetic + reset)	human	5 (4 fail + 1 succ)
Rollout	✓	✓	✓	–	t_{rollout} (policy on hardware)	human	20 π_0 rollouts
Human	–	✓	–	✓	t_{human} (no robot needed)	human	20 (19 fail + 1 succ)
Real2Ssim	✓	–	–	✓	t_{asset}/N (amortized setup)	automatic	30(29 fail + 1 succ)

Counterfactual data for manipulation.. The sources we study correspond to naturally available data used in robot learning. Teleoperation and policy rollouts underpin large-scale corpora such as DROID [14] and Open X-Embodiment [26], as well as distributed evaluation efforts such as RoboArena [2], and collectively these sources show how expensive controlled hardware trials remain. Human video datasets like Epic-Kitchens [9] and EgoDex [13] have been used as cheap sources for robot learning across many tasks; even Robometer uses Epic-Kitchens in its training recipe. Simulation benchmarks such as LIBERO [19] and RoboCasa [25] provide cheap, exactly labeled rollouts at the cost of visual and dynamics gaps, which real-to-sim methods narrow by lifting the real scene into simulation [6, 31]. Prior works typically study or use these sources in isolation for policy learning, but we compare them for reward learning under a common cost model, annotation scheme, and evaluation protocol.

3 Study Design

We design our study around the question: *Given a limited data-collection budget, which sources of counterfactual data most improves learned reward models for reinforcement learning?* We keep the reward models, task suite, offline evaluation benchmark, RL setup fixed and vary only the *source* of counterfactual data and the *scale* collected from each source.

3.1 Robot setup and tasks

We evaluate on five manipulation primitives using a standard DROID [14] setup with a Franka Emika Panda arm and a Robotiq gripper as seen in Figure 2. The tasks are: *Pick up the spoon and put it on the plate* (Spoon), *Pick up the sellotape and put it in the grey box* (Tape), *Remove the lid from the pot* (Remove Lid), *Pour the contents of the cup into the grey box* (Pour), and *Stack up the cups together* (Stack Cups).

3.2 Counterfactual sources

We collect counterfactual data from four sources (Table 1), each offering a different trade-off between collection effort and alignment to the reward-learning setting. Two activities dominate the per-episode cost of every source: *collection*, the time an operator spends generating a trajectory, and *annotation*, the time a human spends assigning partial-credit labels. Sources that require the physical robot for collection are substantially more expensive per episode than those that do not, and sources whose labels can be obtained automatically remove the annotation cost entirely. This produces a clear cost ordering across sources. Source examples and the partial-credit labeling scheme are given in Appendix A.1.

Kinesthetic-teaching failures are the most expensive and most aligned source. An operator uses the Franka's program mode to kinesthetically record each trajectory by physically guiding the robot arm, then rearranges the objects to the exact same initial configuration and replays the recorded action. Each episode therefore requires both a kinesthetic-teaching phase and a physical replay phase, both of which demand direct robot access. Per task, we collect one expert trajectory and the rest as intentional failure modes, all recorded at 15–20 Hz and annotated by a human with partial-credit labels. Per episode, the cost is

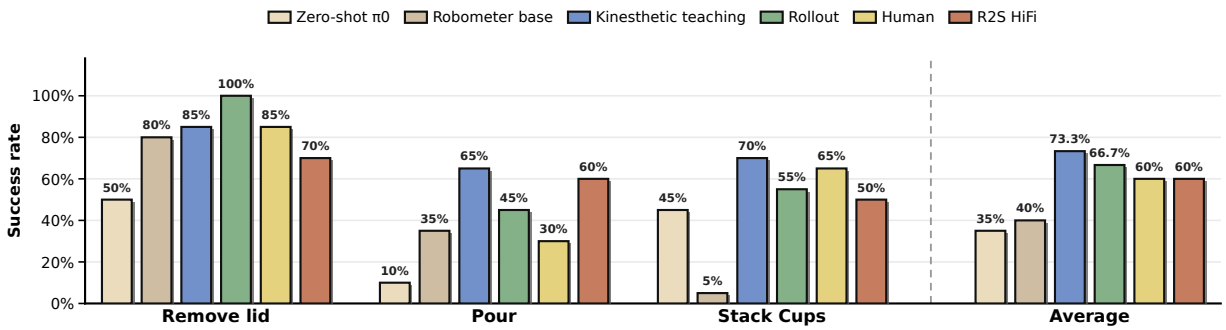


Figure 3: Real-robot RL results (RQ1). Success rate over 20 trials for zero-shot π_0 and DSRL with the base Robometer or source-adapted Robometer on three tasks. On average, every adapted reward improves over the base reward. Cheaper sources such as human video and real2sim are competitive with kinesthetic teaching despite their domain gaps.

$t_{\text{collect}} + t_{\text{replay}} + t_{\text{annotate}}$, where collection and replay together make this the most time-consuming source.

Policy-rollout failures require the robot but not an operator guiding it. We deploy our base policy π_0-DROID [4] on hardware and collect approximately 20 rollouts per task, recorded at 8–10 Hz due to policy latency. Because the trajectories come from the policy itself, rollouts are on-policy and faithful to the states the learner actually reaches, but we have no control over the distribution of successes and failures. Each episode requires a rollout on the robot (t_{rollout}) followed by human annotation (t_{annotate}). The collection cost per episode is roughly half that of kinesthetic-teaching data, since there is no kinesthetic teaching.

Human-video failures remove the robot from the loop entirely. A person performs the task by hand in the same scene, producing one expert trajectory and the rest as intentional failure modes per task. Collection is fast (t_{human}): it requires no robot setup, no kinesthetic teaching, and no physical replay, making it substantially cheaper per episode than either robot-based source. However, the resulting data crosses an embodiment gap between human hands and the robot arm, reducing its alignment to the target setting. Trajectories are annotated with partial-credit labels by a human (t_{annotate}). We collect 20 episodes per task for this source.

Real2sim failures have the lowest marginal cost per episode. We use components of real-to-sim methods such as SimFoundry [31] and Agentic Real2Sim [6] to build articulated 3D assets from the real scene and arrange them in simulation. The upfront effort is asset generation (t_{asset}), which is a one-time cost per scene. Once the scene is built, failures are generated by scripted perturbations and labels are assigned automatically from the simulator state, so both collection and annotation scale at near-zero marginal cost. Each task requires only one real-world successful demonstration (obtained from kinesthetic replay) to define the task; we then generate 29 failure trajectories and one replayed success per task. Real2sim scene construction and failure generation are detailed in Appendix A.1.1 and Algorithm 1.

Taken together, the per-episode cost ordering is intuitively:

$$\underbrace{t_{\text{collect}} + t_{\text{replay}}}_{\text{KT}} > \underbrace{t_{\text{rollout}}}_{\text{RO}} \gg \underbrace{t_{\text{human}}}_{\text{HV}} > \underbrace{t_{\text{asset}}/N}_{\text{R2S (amortized)}}$$

where all sources except real2sim additionally incur t_{annotate} for human labeling. This cost structure is what makes the scaling question meaningful: the cheaper sources can buy more data per unit of operator time, and our experiments test whether that quantity compensates for their weaker alignment.

3.3 Reward models

We study two ways of using counterfactual supervision. We fine-tune Robometer-4B [17] with LoRA on data pooled across all five tasks. We also use Generative Value Learning (GVL) [23], instantiated with GPT-4o, and provide task-specific trajectories and their whole-trajectory partial-credit labels as few-shot context. Unlike TOPReward [7], which reads progress from the probability of a completion token, GVL outputs progress values directly, which makes it a natural fit for few-shot prompting with our partial-credit labels.

We keep the successful robot demonstrations fixed across source comparisons, so that the main difference is the failure supervision each source provides.

For each model, we keep the adaptation procedure fixed across counterfactual sources. We include both models in the matched-data comparison to test whether source choice matters for both parameter adaptation and in-context prediction. We use Robometer for the scaling, simulation-fidelity, and downstream RL experiments, since our GVL setting accommodates only a small number of examples in its prompt. Robometer training hyperparameters and GVL prompting details are given in Appendix A.2 and Appendix A.2.2, respectively.

3.4 Real-world RL setup

We use real-robot RL to test whether the choice of counterfactual supervision changes the behavior learned by a policy. We post-train the DROID checkpoint of π_0 with DSRL [38], which learns to steer the latent noise of the frozen base policy. After each episode, we relabel rewards with the selected reward model, combining its predicted progress with a sparse human success label (Appendix A.3).

From the five-task pool in Figure 1, we select three tasks spanning a range of manipulation difficulty for the RL experiments: *Remove Lid* (easy: a single contact-and-lift), *Pour* (medium: requires controlled tilting and sustained contact), and *Stack Cups* (hard: demands precise alignment and release of a deformable grasp). We keep the base policy, warm-up rollouts, interaction budget, and optimization hyperparameters fixed across reward conditions. We compare zero-shot π_0 with DSRL using the base Robometer and Robometer adapted with each of the four counterfactual sources. For each source, we use the reward checkpoint at its best data scale from the scaling experiments (Section 4.2): ten trajectories per task for policy rollouts, human video, and real2sim, and five for kinesthetic teaching. We evaluate each condition in 20 blind, randomized trials with matched initial conditions and measure task success directly from the physical execution. The selected reward checkpoints and RL hyperparameters are listed in Appendix A.3.

3.5 Offline evaluation benchmark and metrics

We use a custom offline benchmark to complement the RL experiments, since it is impractical to run online RL for every reward-model variant across all control and scaling conditions. Offline metrics are computed on a held-out benchmark of $5 \times 50 = 250$ annotated episodes: for each task, 25 zero-shot rollouts of $\pi_{0.5}$ [28] and 25 of MolmoAct2 [11], each with human-assigned partial-credit labels. No π_0 rollout appears in the benchmark, so every offline evaluation is cross-policy: counterfactuals come from π_0, an operator, a human, or real2sim, while the reward model is evaluated on the behaviors of two other policies. No benchmark episode is used for fine-tuning, although we use the benchmark to select the data scale of the RL reward checkpoints (Section 3.4).

We report two primary offline metrics. **Trajectory ranking** (τ_{return}) measures how well the reward model orders whole trajectories by their mean predicted progress, computed as the Kendall rank correlation between the human-assigned partial credits and the trajectory-averaged reward, averaged across tasks. **Success detection** (ROC-AUC$_{\text{return}}$) measures binary success classification, treating the trajectory-averaged predicted reward as a classifier score, with success defined as partial credit 1.0. Both metrics use the trajectory-averaged prediction rather than the final-frame prediction, since trajectory-level scores better reflect how RL integrates dense per-step rewards over an episode. Frame sampling and bootstrapping details are given in Appendix A.2.

4 Results

We curate our results around five research questions. We first ask whether the source of counterfactuals affects how RL performs (RQ1). We then use the offline benchmark to study the sources at a fixed scale (RQ2) and when scaled according to budget (RQ3), followed by simulation fidelity (RQ4) and how well the offline metrics agree with RL (RQ5).

4.1 Real-world RL

RQ1: Can rewards learned from cheap counterfactual sources still give gains in RL?. Real-world RL is the most direct test for reward models and also the most expensive one, so we run it on three tasks of varying

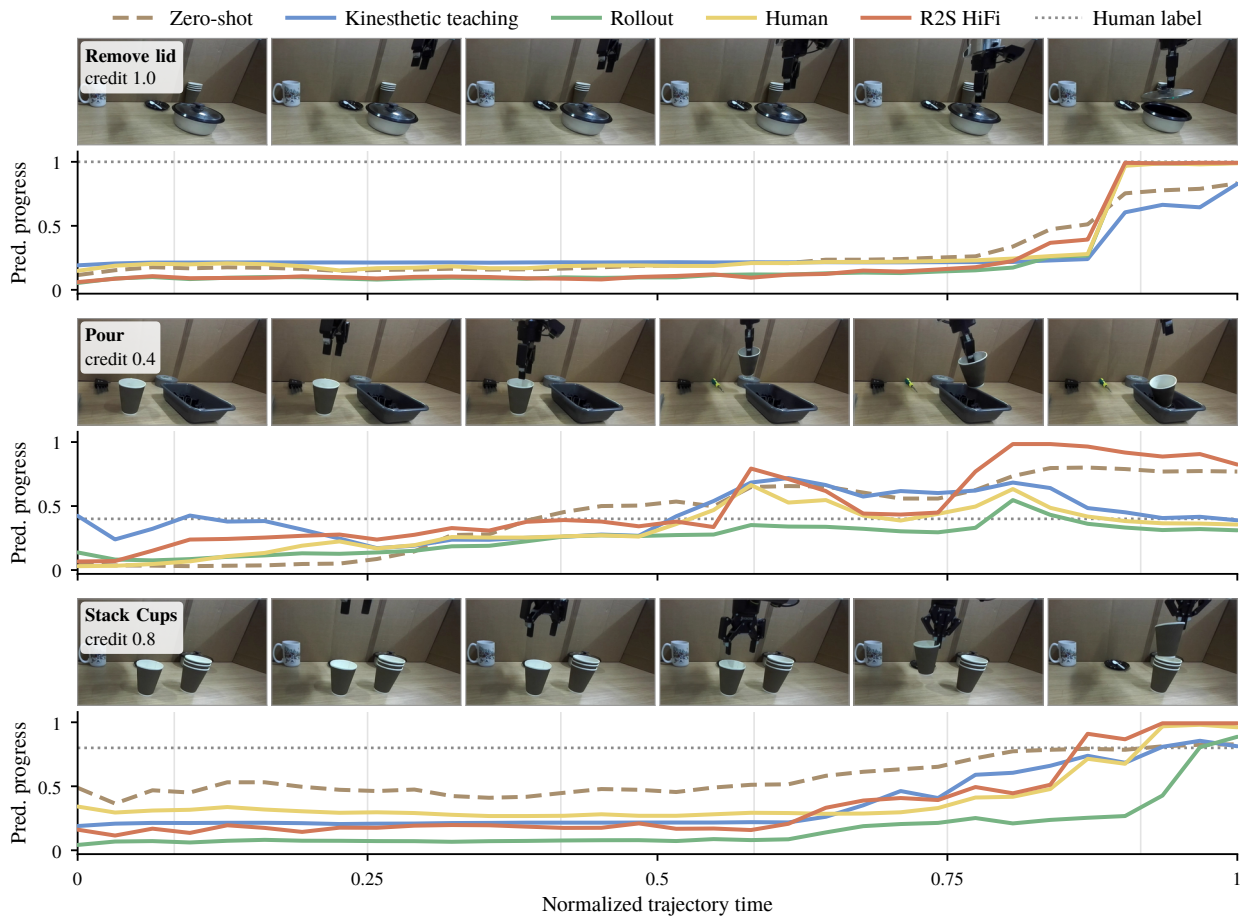


Figure 4: Reward predictions on the three RL tasks. Base and source-adapted Robometer predictions on one benchmark episode per task, with frames placed above the time they show; the dotted line is the human-assigned credit. **Remove Lid:** every model jumps when the lid comes off. **Pour** (failed, credit 0.4): the base Robometer climbs to 0.77 from the cup motion alone. **Stack Cups:** the base Robometer predicts around 0.5 before any cup moves. More examples are in Appendix A.4.

difficulty (Section 3.4). For each task, we compare the zero-shot policy, DSRL with the base Robometer, and DSRL with Robometer adapted to each source over 20 trials (Figure 3).

All the counterfactual sources improve RL, including cheap sources with large domain gaps. Averaged over the three tasks, every adapted reward improves over the base reward, from 40% to between 60% and 73.3% (Figure 3, right). Kinesthetic teaching, the costliest source, gives the highest average, but surprisingly the cheaper sources perform competitively despite the embodiment gap in human video and the visual domain gap in real2sim. Kinesthetic teaching is the most reliable source, but its advantage is minuscule compared to its collection cost.

To better understand the individual results, we look at a few reward predictions from the offline benchmark (Figure 4). On Remove Lid, every model jumps when the lid comes off, which matches the RL results staying close to the base reward (70% to 100% vs. 80%). On Stack Cups, the base Robometer rewards the scene rather than the task: the already stacked cups make it predict around 0.5 progress from the start, which drops RL below the zero-shot policy (45% to 5%). Pour is less clear-cut. The base Robometer climbs to 0.77 on a failed episode from the cup motion alone, while kinesthetic teaching ends close to the label. Real2sim stays close to the base here, yet gives one of the larger RL gains (60% vs. 35%), a gap we return to in RQ5 (Section 4.2).

Takeaway. Every source improves RL over the base reward, and the cheaper sources are competitive with kinesthetic teaching despite their domain gaps.

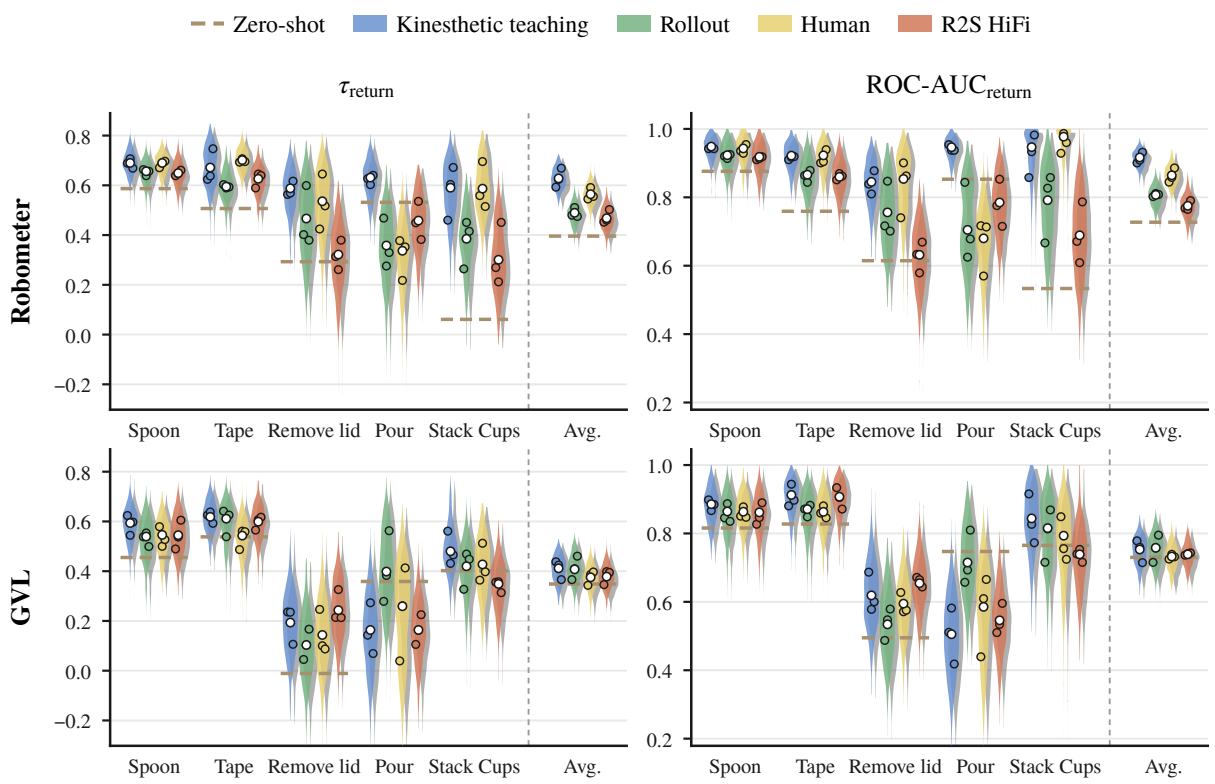


Figure 5: Matched-scale comparison of counterfactual sources (RQ2). Robometer (top) and few-shot GVL (bottom) adapted with five trajectories per task from each source and evaluated on the held-out benchmark, using trajectory ranking (τ_{return}, left) and success detection (ROC-AUC$_{\text{return}}$, right). All sources improve Robometer kinesthetic teaching helps most, and human video outperforms policy rollouts. GVL barely responds to any source.

4.2 Offline evaluation

Since real-robot RL is too expensive to run for every condition, we use the offline benchmark (Section 3.5) to measure how source, scale, and simulation fidelity change the reward itself, and whether these metrics agree with RL (RQ5).

RQ2: Which sources teach the reward model most at fixed scale?. Holding the data fixed at five trajectories per task isolates what each source contributes. All four sources help Robometer (Figure 5, top). Kinesthetic teaching gives the largest gains, but human video ranks trajectories better than policy rollouts, even with the embodiment gap. This suggests that cheap sources still carry useful failure signal despite their domain gaps, which may explain why they are competitive in RL. GVL, which only sees the counterfactuals as few-shot examples (Section 3.3), barely changes with any source (Figure 5, bottom).

Takeaway. Kinesthetic teaching based coutnerfactuals improve Robometer the most, with human video second and ahead of policy rollouts. GVL barely changes with any counterfactual source.

RQ3: Is more counterfactual data always better?. We scale each source according to its collection budget: kinesthetic teaching stays at five trajectories per task, policy rollouts and human video go from 5 to 10 to 20, and real2sim, the cheapest, goes up to 30 (Figure 6, left). Human video barely changes with scale, while real2sim improves up to ten and then degrades, falling below the base Robometer at 25–30. Policy rollouts show the sharpest effect: ten per task are as good as five kinesthetic-teaching trajectories, but twenty are worse than no adaptation at all. Since training length grows with the amount of data (Appendix A.2), we cannot yet separate more data from longer training. Either way, cheap sources need to be scaled carefully, which is why our RL experiments use the best scale for each source (Section 3.4).

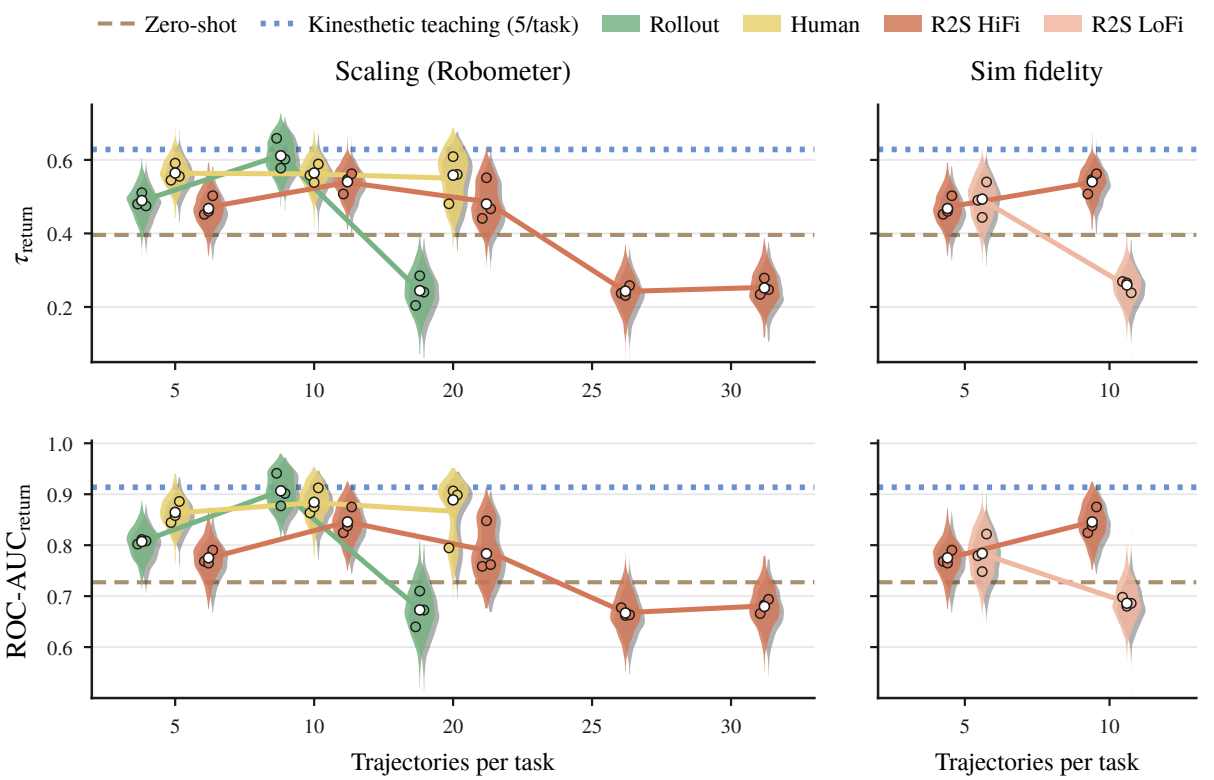


Figure 6: Scaling and simulation fidelity (RQ3, RQ4). Left: Robometer adapted with increasing cheap counterfactual data; Kinesthetic teaching at five per task (dotted) and base Robometer (dashed) for reference. Cheap sources help up to 10 trajectories per task and can hurt beyond. **Right**: high-fidelity (R2S-HiFi) vs low-fidelity (R2S-LoFi) simulation. At small scale the two are equivalent at ten per task, only R2S-HiFi improves, showing that visual realism matters when scaling simulated data.

Takeaway. More data is not always better. Cheap sources help up to about ten trajectories per task and can make the reward modeling worse beyond that.

RQ4: How real does a simulated failure need to look?. How much realism a simulated failure needs depends on how much simulated data is used. At five trajectories per task, textureless MuJoCo renderings (R2S-LoFi) are as useful as Gaussian-splat renderings of the real scene (R2S-HiFi) on both metrics (Figure 6, right). Realism starts to pay off when scaling: at ten per task, R2S-HiFi improves while R2S-LoFi drops below the base Robometer. As the two also use different simulators, this compares the full pipelines rather than rendering alone (Appendix A.1.1).

Takeaway. Low-fidelity simulation is enough at small scale, but realism pays off once simulated data is scaled.

RQ5: Can offline evaluation tell us which reward to deploy?. We compare the offline metrics with the RL results for the 15 reward and task pairs we ran on the robot (Figure 7). Trajectory ranking shows moderate correaltion in comparison to success detection ($r = 0.40$, $p = 0.139$ for τ_{return}; $r = 0.24$, $p = 0.391$ for ROC-AUC$_{\text{return}}$). The only reward that is poor on both axes is the base Robometer on Stack Cups, so the offline metrics can catch a broken reward, but a better offline score does not reliably mean a better reward for RL. Closing this gap needs better offline evaluations that reflect how a reward is actually used during RL.

Takeaway. Offline metrics catch a broken reward but do not reliably rank good ones. Better offline evaluation is needed.

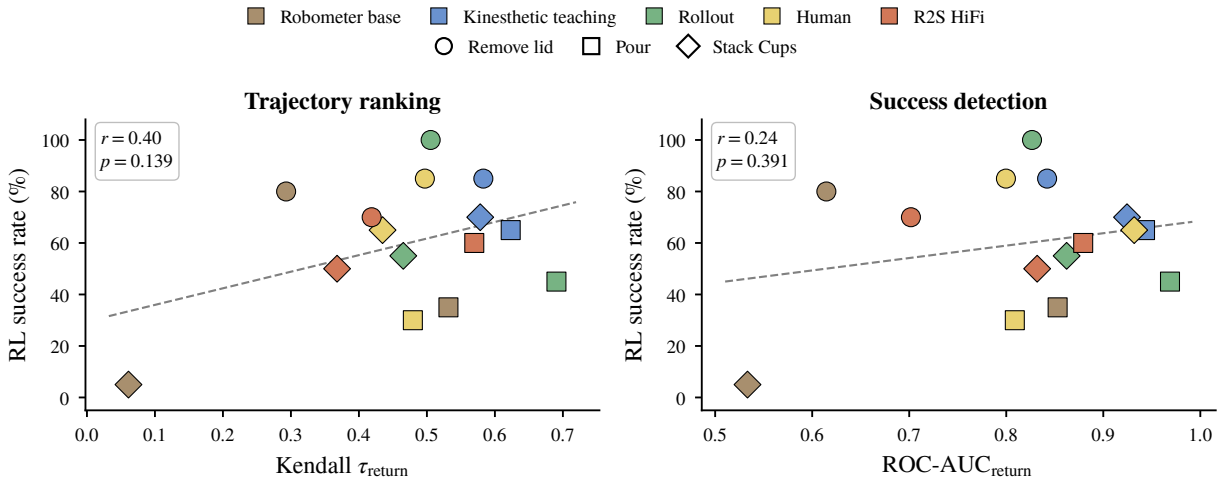


Figure 7: Do offline metrics predict RL success? (RQ5) Offline score vs. RL success rate for 15 (reward, task) pairs. τ_{return} (r=0.40, p=0.139) tracks RL success slightly better than ROC-AUC$_{return}$ (r=0.24, p=0.391).

5 Discussion, Limitations, and Conclusion

Improving robot policies with RL requires rewards that recognize both progress and failure. We studied how the source and scale of counterfactual data affect these rewards and the policies trained with them. Kinesthetic-teaching failures provide strong supervision but require substantial robot and operator time. We showed that cheaper human-video and counterfactuals generated in simulation through real2sim techniques approached their performance with more data, although further scaling did not consistently improve rewards.

These conclusions come with caveats. Our trainable reward model is Robometer throughout, which already has good priors; training a reward model from scratch or adapting other VLM-based architectures may shift the source rankings, and we leave this exploration to future work. We cover five tabletop tasks on one embodiment, with RL on three of them—more tasks and embodiments would make the picture more complete. Our offline benchmark uses discrete partial credits for computing metrics, but a dense human-labeled reward curve would give more information; the cost of collecting such annotations at scale is impractical.

Our results show that a small budget of counterfactual data combined with strong priors can improve the reward regardless of the source, and cheaper sources, when scaled carefully, can match expensive ones.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pp. 1, 2004.

[2] Pranav Atreya, Karl Pertsch, Tony Lee, Moo Jin Kim, Arhan Jain, Artur Kuramshin, Clemens Eppner, Cyrus Neary, Edward Hu, Fabio Ramos, et al. RoboArena: Distributed real-world evaluation of generalist robot policies. *arXiv preprint arXiv:2506.18123*, 2025.

[3] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[4] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[5] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal

demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning*, pp. 783–792. PMLR, 2019.

[6] Guanxiong Chen, Qianjun Xia, Jiawei Peng, Heng Zhang, Bole Ma, Justin Qian, Ziyi Jiao, Bingyang Zhou, Luoxin Ye, Kaifeng Zhang, et al. Agentic real2sim: Physics-based world modeling with vision-language agents. *arXiv preprint arXiv:2607.19190*, 2026.

[7] Shirui Chen, Cole Harrison, Ying-Chun Lee, Angela Jin Yang, Zhongzheng Ren, Lillian J Ratliff, Jiafei Duan, Dieter Fox, and Ranjay Krishna. TOPReward: Token probabilities as hidden zero-shot rewards for robotics. *arXiv preprint arXiv:2602.19313*, 2026.

[8] Yinpei Dai, Jayjun Lee, Nima Fazeli, and Joyce Chai. RACER: Rich language-guided failure recovery policies for imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15657–15664. IEEE, 2025.

[9] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. The epic-kitchens dataset: Collection, challenges and baselines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):4125–4141, 2020.

[10] Perry Dong, Kuo-Han Hung, Tian Gao, Dorsa Sadigh, and Chelsea Finn. Expo-ft: Sample-efficient reinforcement learning finetuning for vision-language-action models. *arXiv preprint arXiv:2605.25477*, 2026.

[11] Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, et al. Molmoact2: Action reasoning models for real-world deployment. *arXiv preprint arXiv:2605.02881*, 2026.

[12] Abhishek Gupta, Aldo Pacchiano, Yuexiang Zhai, Sham Kakade, and Sergey Levine. Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity. *Advances in Neural Information Processing Systems*, 35:15281–15295, 2022.

[13] Ryan Hoque, Peide Huang, David Yoon, Jian Zhang, et al. Egodex: Learning dexterous manipulation from large-scale egocentric video. In *International Conference on Learning Representations*, volume 2026, pp. 4218–4237, 2026.

[14] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. DROID: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[15] Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. RoboReward: General-purpose vision-language reward models for robotics. *arXiv preprint arXiv:2601.00675*, 2026.

[16] Kun Lei, Huanyu Li, Dongjie Yu, Zhenyu Wei, Lingxiao Guo, Zhennan Jiang, Ziyu Wang, Shiyu Liang, and Huazhe Xu. Rl-100: Performant robotic manipulation with real-world reinforcement learning. *arXiv preprint arXiv:2510.14830*, 2025.

[17] Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S Huang, Luke Zettlemoyer, Dieter Fox, et al. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. *arXiv preprint arXiv:2603.02115*, 2026.

[18] 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.

[19] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[20] Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105):eads5033, 2025.

[21] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

[22] Yecheng Jason Ma, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. LIV: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning*, pp. 23301–23320. PMLR, 2023.

[23] Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, et al. Vision language models are in-context value learners. In *International Conference on Learning Representations*, volume 2025, pp. 33984–34009, 2025.

[24] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023.

[25] Soroush Nasiriany, Sep Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations*, volume 2026, pp. 98643–98667, 2026.

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

[27] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[28] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[29] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

[30] Nived Rajaraman, Lin Yang, Jiantao Jiao, and Kannan Ramchandran. Toward the fundamental limits of imitation learning. *Advances in Neural Information Processing Systems*, 33:2914–2924, 2020.

[31] Nadun Ranawaka, Josiah Wong, Wei-Lin Pai, Wei-Teng Chu, Tianyuan Dai, Masoud Moghani, Hang Yin, Yunfan Jiang, Wesley Durbano, Brandon Huynh, et al. Simfoundry: Modular and automated scene generation for policy learning and evaluation. *arXiv preprint arXiv:2606.28276*, 2026.

[32] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 661–668. JMLR Workshop and Conference Proceedings, 2010.

[33] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

[34] Sumedh Sontakke, Jesse Zhang, Séb Arnold, Karl Pertsch, Erdem Bıyık, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. RoboCLIP: One demonstration is enough to learn robot policies. *Advances in Neural Information Processing Systems*, 36:55681–55693, 2023.

[35] Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Yuheng Ji, Cheng Chi, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, et al. Robo-Dopamine: General process reward modeling for high-precision robotic manipulation. *arXiv preprint arXiv:2512.23703*, 2025.

[36] Ran Tian, Yilin Wu, and Andrea Bajcsy. Position: Good embodied reward models need bad behavior data. *arXiv preprint arXiv:2606.01036*, 2026.

[37] Marcel Torne, Anubha Mahajan, Abhijnya Bhat, and Chelsea Finn. Freeform preference learning for robotic manipulation. *arXiv preprint arXiv:2606.32027*, 2026.

[38] Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

[39] Yanru Wu, Weiduo Yuan, Ang Qi, Vitor Guizilini, Jiageng Mao, and Yue Wang. Large reward models: Generalizable online robot reward generation with vision-language models. *arXiv preprint arXiv:2603.16065*, 2026.

[40] Pengzhi Yang, Xinyu Wang, Pengyu Jing, Kehan Wen, Yiduo Qu, Zhenhao Huang, Minghao Fu, Xin Liu, Yaheng Shen, and Fan Shi. Rarm: Confidence-gated progress reward modeling for rl in manipulation. *arXiv preprint arXiv:2606.22027*, 2026.

[41] Patrick Yin, Tyler Westenbroek, Zhengyu Zhang, Ignacio Dagnino, Eeshani Shilamkar, Numfor Mbiziwo-Tiapo, Simran Bagaria, Xinlei Liu, Galen Mullins, Andrey Kolobov, et al. Emergent dexterity via diverse resets and large-scale reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 88343–88356, 2026.

[42] Shaopeng Zhai, Qi Zhang, Tianyi Zhang, Fuxian Huang, Haoran Zhang, Ming Zhou, Shengzhe Zhang, Litao Liu, Sixu Lin, and Jiangmiao Pang. A vision-language-action-critic model for robotic real-world reinforcement learning. *arXiv preprint arXiv:2509.15937*, 2025.

[43] Jiahui Zhang, Yusen Luo, Abrar Anwar, Sumedh Anand Sontakke, Joseph J Lim, Jesse Thomason, Erdem Biyik, and Jesse Zhang. ReWiND: Language-guided rewards teach robot policies without new demonstrations. *arXiv preprint arXiv:2505.10911*, 2025.

[44] Octi Zhang, Mateo Guaman Castro, Patrick Yin, Ignacio Dagnino, Abhishek Gupta, Rosario Scalise, and Byron Boots. A balanced data diet: Addressing the exploration bottleneck in mega-scale rl for robot control. *arXiv preprint arXiv:2610.12465*, 2026.

[45] Zhiyuan Zhou, Pranav Atreya, Abraham Lee, Homer Walke, Oier Mees, and Sergey Levine. Autonomous improvement of instruction following skills via foundation models. *arXiv preprint arXiv:2407.20635*, 2024.

[46] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pp. 1433–1438, 2008.

A Appendix

The appendix gives more details on how we collect and label counterfactual data (Appendix A.1), how we train and prompt the reward models (Appendix A.2), and how we run RL (Appendix A.3). Our robot is set up with the `droid-fr3` codebase[1], the reward models are trained with the Robometer codebase[2], and RL is run with the Robometer policy-learning codebase[3].

A.1 Counterfactual sources and annotation

To make data easy to inspect for all four sources, with their partial-credit labels, can be explored in our **counterfactual data explorer**. It has kinesthetic-teaching failures, policy rollouts, human video, and real2sim trajectories for all five tasks.

Partial-credit annotation. We label every trajectory with a partial credit from $\mathcal{P} = \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$, where 0.0 means no real progress and 1.0 means the task was completed. Values in between say on average how the task was done. Kinesthetic-teaching, policy-rollout, and human-video trajectories are labeled by human, while real2sim get their labels automatically from the task milestones in the simulator (Appendix A.1.2). All four sources use the same scale.

A.1.1 Real2sim scene construction

For each task, we use one successful kinesthetic demonstration to build the same scene in two simulators: OmniGibson for high fidelity (R2S-HiFi) and MuJoCo for low fidelity (R2S-LoFi). Both keep the robot, task objects, initial configurations, and task definition, but they differ in rendering and in the simulator itself.

High-fidelity real2sim. The high-fidelity scene is built with parts of SimFoundry [31]. From calibrated exterior and wrist camera recordings, we get the scene geometry, object poses, and camera alignment. Each object is segmented and reconstructed as a textured 3D mesh with collision geometry before being imported into OmniGibson, and a Gaussian-splat reconstruction of the workspace gives the background. We had to define the collision geometry for hollow and thin objects (the cups, grey box, tape, and pot lid) so that grasping, insertion, and pouring behave properly. Finally, the scene is aligned with the real robot using the camera calibration and robot kinematics.

Low-fidelity real2sim. Instead of reconstructing the scene again, we reuse the R2S-HiFi assets. We move over the object meshes, initial poses, robot configuration, camera calibration, and task predicates, and use the same Franka arm and Robotiq gripper. The only catch is that we use textureless assets, and the background is a standard simulator background with no details, unlike R2S-HiFi. This gives a visually low-fidelity version of the exact same task, so every R2S-LoFi trajectory has a matching R2S-HiFi trajectory. Since the simulator also changes, the fidelity ablation in RQ4 (Section 4.2) compares the two full pipelines.

Successful demonstration replay. From the successful kinesthetic demonstration, we keep the joint positions, gripper commands, end-effector poses, and camera calibration. Long idle parts are cut out, but anything that matters for the task, like contact, grasping, and release, is kept, with a short settling period wherever needed. We then replay this edited trajectory in both simulators and keep it only if it succeeds and ends in a stable state. This replay is then used to make failures with the algorithm described below (Appendix A.1.2).

A.1.2 Real2sim failure generation

We make failures by replaying the successful demonstration and perturbing it at different task stages, similar in spirit to MimicGen [24] but for failures. First, we change the initial object configuration and retarget the reference trajectory relative to the manipulated object. For an end-effector pose T_{WE} and object pose T_{WO}, the new retargeted end-effector pose is

$$T'_{WE} = T'_{WO}\, T_{WO}^{-1}\, T_{WE}, \tag{1}$$

[1] https://github.com/EmbodiedForge/droid-fr3
[2] https://github.com/robometer/robometer
[3] https://github.com/robometer/robometer-policy-learning

where W, O, and E are the world, object, and end-effector frames. This keeps the end-effector in the same pose relative to the object.

Each failure comes from one operator applied to one task stage. The operators are `truncate_before`, `no_grip`, `collision_nudge`, `under_actuate`, `halt_mid_transport`, `early_release`, and `goal_offset`. These operators range over a set of commonly observed, realistic failures. The part of the trajectory before the chosen stage is kept as it is, and the edited part is solved with inverse kinematics and interpolated (Algorithm 1).

Automatic labels. Each task is split into roughly five ordered milestones (contact with the object, a secure grasp, lifting or partial manipulation, alignment or placement, and task completion), detected from the simulator states. The partial credit is assigned using

$$p = 0.2k, \qquad k \in \{0, 1, 2, 3, 4, 5\}, \tag{2}$$

where k is the number of milestones reached in order. A milestone counts only if the ones before it were reached. A failure is kept only if the labels are agreed upon in both simulators and the final state is stable.

Algorithm 1: Real2sim counterfactual generation

Require: Successful demonstration τ; task milestones $\mathcal{M}$; failure operators $\mathcal{O}$; target failure count $N_{\text{fail}} = 29$
Ensure: Selected failures $\mathcal{D}^-$ and successful replay τ^+

// Phase 1: Prepare a validated successful replay
$\tau^+ \leftarrow \textsc{CompressIdle}(\tau)$
Preserve contact, grasp, release, and task-critical windows
Replay τ^+ in MuJoCo and OmniGibson
Verify ordered milestones, task success, and stable final state

// Phase 2: Generate candidate failures
$\mathcal{C} \leftarrow \emptyset$
while insufficient validated candidate coverage **do**
    Sample matched initial object-state perturbation
    Retarget the reference trajectory in the object frame
    Sample failure stage, operator $o \sim \mathcal{O}$, and operator parameters
    Modify the selected stage while preserving the valid prefix
    Solve inverse kinematics and interpolate commands
    $\tau_{\text{MJ}} \leftarrow \textsc{ReplayMuJoCo}(\tilde{\tau})$
    $\tau_{\text{OG}} \leftarrow \textsc{ReplayOmniGibson}(\tilde{\tau})$
    Compute the longest ordered milestone prefix in each simulator
    Assign $p = 0.2k$
    if both simulators fail, agree on p, preserve milestone order, and terminate stably **then**
        Add the candidate, reward, states, and videos to $\mathcal{C}$
    end if
end while

// Phase 3: Select and export the dataset
Cover different failure-operator outcomes and partial-credit levels
Select $N_{\text{fail}} = 29$ failures using visual-diversity sampling
$\mathcal{D}^- \leftarrow \textsc{SelectFailures}(\mathcal{C}, 29)$
Export matched R2S-HiFi/R2S-LoFi videos, metadata, and rewards
return $\mathcal{D}^-, \tau^+$

Out of all the saved failures, we select 29 failures per task. The selection makes sure that diverse failure types and credit levels are covered, using visual-diversity sampling. Together with the successful replay, this gives 30 episodes per task in both R2S-HiFi and R2S-LoFi. Human effort is only needed once per task, to collect the successful demo and set up the scene and milestones. Everything after that runs automatically.

Table 2: Robometer adaptation hyperparameters. The same configuration is used for every counterfactual source.

Parameter	Configuration
Starting checkpoint	Robometer-4B
Backbone	Qwen3-VL-4B-Instruct
Trainable components	Language-model LoRA + three Robometer heads
Vision encoder	Frozen
LoRA rank / alpha / dropout	16 / 32 / 0.05
Optimizer	AdamW
Learning rate	1×10^{-4}
Weight decay	0.01
Scheduler / warm-up	Cosine / 10% of training steps
Adam $(\beta_1, \beta_2, \epsilon)$	$(0.9, 0.999, 10^{-8})$
Maximum gradient norm	1.0
Batch size / accumulation	2 / 1
Precision	BF16
Gradient checkpointing	Enabled
Maximum sequence length	1024
Training seeds	42, 43, 44
Training steps	$3N$, where N is the number of training trajectories
Maximum frames per training sample	8

A.2 Reward model training

For a fair comparison, we keep the training setup for Robometer and the prompting setup for GVL fixed across all sources. Only the counterfactual data source and its scale change.

A.2.1 Robometer

We start from the released `Robometer-4B` checkpoint, which uses Qwen3-VL-4B-Instruct as its backbone. We keep the vision encoder frozen and only adapt the language model with LoRA, applied to all attention and MLP projections. Robometer has three heads that are trained together: preference, progress, and success. Each head is a small MLP; the preference and success heads output a single logit, and the progress head outputs logits for ten progress bins. The full configuration is in Table 2.

Sampling and supervision. Each training data point samples up to eight frames per trajectory. Preference pairs are built from suboptimal trajectories of the same task, rewound trajectories, trajectories from different tasks, and reversed progress. For suboptimal trajectories, two partial-credit trajectories are paired only if their labels differ by at least 0.2. For a failed or partial trajectory, the label only supervises the last frame, with no progress labels in between. The rest of the progress signal comes from augmented trajectories and successful ones. The training loss is

$$\mathcal{L} = \mathcal{L}_{\text{pref}} + \mathcal{L}_{\text{progress}} + \mathcal{L}_{\text{success}}, \tag{3}$$

where $\mathcal{L}_{\text{pref}}$ is a binary cross-entropy loss over the constructed preference pairs, $\mathcal{L}_{\text{progress}}$ is cross-entropy over ten bins, and $\mathcal{L}_{\text{success}}$ is a masked binary cross-entropy, all weighted equally. For more details, refer to the Robometer paper [17].

Training runs for $3N$ steps, where N is the number of training trajectories across the five tasks, so larger datasets also train longer. Each smaller training set is a subset of the larger ones, and R2S-LoFi always uses the same episodes as R2S-HiFi for a fair visual-fidelity comparison.

Offline evaluation. We evaluate every model on the 250 held-out benchmark trajectories mentioned in Section 3.5. Each trajectory is uniformly subsampled to 30 frames, and the reported metrics are computed with 10,000 bootstrap samples.

Table 3: GVL configuration. Seeds change the frame order, not the model.

Parameter	Configuration
Model	`gpt-4o`
Adaptation	Frozen model; zero-shot or few-shot prompting
Few-shot context	Five same-task trajectories per source
Target frames	Up to 30, uniformly sampled
Context frames	Up to 30 per example trajectory
Maximum response tokens	4096
Seeds	42, 43, 44
Output	JSON with per-frame completion percentages

Table 4: Policy and observation configuration, shared by all three RL tasks.

Component	Configuration
Frozen base policy	`pi0-DROID`
Flow horizon	10
Executed action chunk	8 actions
SAC noise action	8×32, bounded to $[-2, 2]$
Robot-state observation	8 dimensions
Visual observation	384-D frozen DINOv2-small embedding
Policy feature	2048-D π_0 prefix/VLM feature
Reward-model input	Left exterior view only

A.2.2 GVL

GVL is our frozen, in-context reward model. We query `gpt-4o` through the API, keeping the prompt and task description fixed and only changing the in-context bank for each counterfactual source. For zero-shot GVL results, we just use the default GVL setup to evaluate. For the few-shot setting, we sample five examples with their assigned partial credits from each counterfactual source. While sampling, we make sure the labels per source are well spread, giving diverse partial credits. Table 3 summarizes the settings.

For each trajectory, we sample up to 30 frames uniformly, and we do the same for each few-shot example trajectory, which is appended with its partial credit. The evaluation frames are sent in a shuffled order, and the model returns a JSON list with a completion percentage for every frame. The parsed output is then divided by 100, and the completion of each frame is put back in the original order.

A.3 RL details

For all three RL tasks (Remove Lid, Pour, and Stack Cups), we share the same policy, network, and SAC settings (Tables 4 and 5). We only change the maximum number of control steps and the number of warm-up steps (Table 6).

A.3.1 Policy and observations

We use `pi0-DROID` as the base policy, which stays frozen, and only train the DSRL noise-space actor and critics. We use the π_0 flow horizon of ten, while SAC predicts eight noise rows and the robot executes eight actions per transition. We copy the first two noise rows into the two unused positions. The policy uses two cameras, one exterior and one wrist, along with proprioception and the task instruction. The reward model uses only the exterior camera.

A.3.2 DSRL actor, critics, and SAC

We use Robometer's adaptation [17] of the original DSRL setup [38], which changes both the actor and critic in noise space to use DINOv2 [27] features with a chunking transformer architecture. Table 5 lists the hyperparameters we use. For further details, refer to the Robometer policy-learning codebase (Appendix A)

Table 5: DSRL SAC hyperparameters, shared by all three RL tasks.

Parameter	Configuration
Transformer width / heads / layers	256 / 8 / 6
Actor MLP	$[512, 512]$, ReLU
Critic MLP	$[768, 512]$, ReLU + LayerNorm
Critics / critics per update	4 / 2
Batch size	64
Replay capacity	10,000 transitions
SAC train steps	10 per rollout transition step
Critic updates per actor update	4
Actor learning rate	5×10^{-5}
Critic learning rate	1×10^{-4}
Entropy-coefficient learning rate	1×10^{-4}
Actor Adam ϵ / weight decay	10^{-8} / 10^{-6}
Critic Adam (β_1, β_2)	$(0.95, 0.999)$
Critic Adam ϵ / weight decay	10^{-8} / 10^{-6}
Entropy coefficient	Automatic; target entropy 0
Base discount γ	0.995
Effective transition discount	$0.995^8 \approx 0.9607$
Target-network τ	0.005
Target update interval	1
Precision / seed	FP16 / 0

Table 6: Per-task RL settings. Everything else is shared across tasks (Tables 4 and 5).

	Remove Lid	Pour	Stack Cups
Max control steps	400	500	400
Warm-up steps	2,000	5,000	10,000
Online interaction steps	10,000	10,000	10,000

and the appendix of the Robometer paper [17].

A.3.3 Warm-up and online training

Each task has a different maximum number of control steps per rollout, all at 15 Hz. Before online training starts, we collect a warm-up dataset for SAC by sampling uniform noise. Every reward condition uses this same warm-up from scratch, so all RL runs across the different counterfactual sources start from the same point. For the per-task max control steps, warm-up steps, and online interaction steps, refer to Table 6.

A.3.4 Reward and selected checkpoints

After each episode, Robometer scores the trajectory with its progress prediction, which is then combined with the sparse human reward of -1 for every step and 0 when the episode ends successfully. We use one Robometer checkpoint per source for RL, since it is trained jointly on all tasks. Since the cheap sources can be collected at several scales, we pick the scale and seed of each source that did best offline. This ends up being ten trajectories per task for policy rollouts, human video, and R2S-HiFi, while kinesthetic teaching uses five, as seen in Table 7.

A.4 Additional qualitative reward results

All reward curves, per task and per reward condition, can be found in our **reward benchmark explorer**. It shows the predictions together with their human partial-credit labels, so the reward behavior can be compared on all benchmark trajectories beyond the examples in Figure 4.

Table 7: Reward checkpoints used for RL. For each source, we use the checkpoint that performs best offline; the same checkpoint per source is used for all three RL tasks.

Source	Trajectories per task
Kinesthetic	5
Rollout	10
Human	10
R2S-HiFi	10