CAN CHEAP FAILURES TEACH GOOD REWARDS? STUDYING COUNTERFACTUAL SOURCES FOR REWARD LEARNING IN MANIPULATION

Anonymous authors
Paper under double-blind review

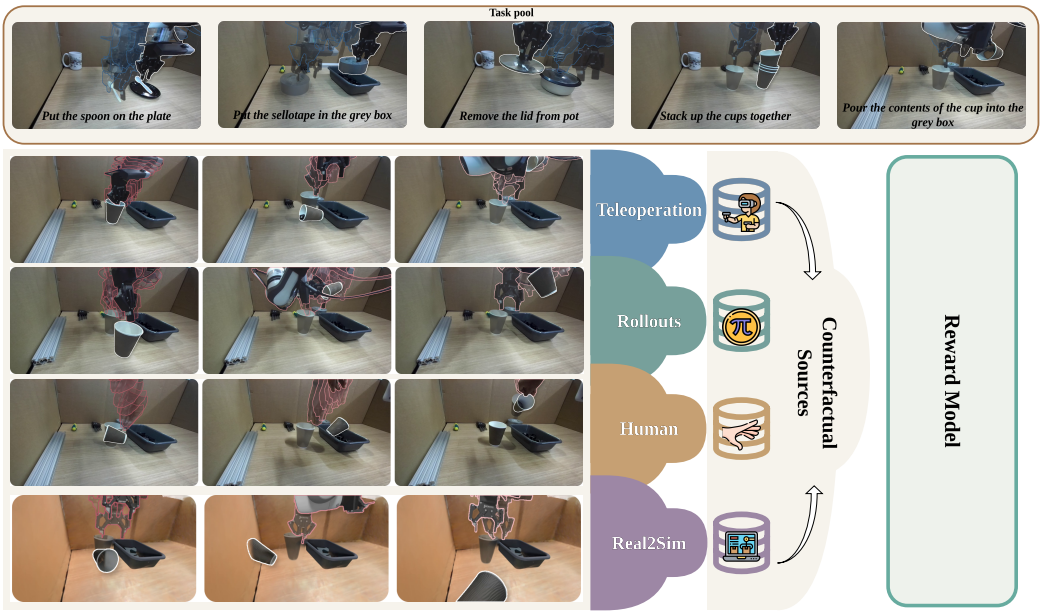

Figure 1: **Overview of the study.** Five real manipulation tasks provide counterfactual trajectories from teleoperation, policy rollouts, human videos, and real2sim. We compare how source and scale affect learned reward models and subsequent robot-policy post-training.

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: teleoperated failures, policy rollouts, human videos, and real-to-sim 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, partial-completion calibration, 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 sur-

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.

pass smaller amounts of faithful real-robot failure data when scaled, for improving rewards models in robot manipulation.

1 INTRODUCTION

Improving robot manipulation policies after deployment is a key step toward robots that can adapt to new scenarios without requiring additional expert demonstrations. A policy trained through imitation learning may provide a strong initalization, but it can still fail when small execution errors compound, when contact must be precise, or when the environment differs from the demonstration setting (Ross & Bagnell, 2010; Ross et al., 2011; Rajaraman et al., 2020). Reinforcement learning (RL) offers a promising framework for adapting such policies, since it allows the robot to improve through its own interaction experience (Yu et al., 2025; Lu et al., 2025; Ball et al., 2023). However, the effectiveness of RL depends critically on the reward function. For sample-efficient robot manipulation, this reward must do more than identify final task success: it must provide dense information about task progress, recognize successful completion, and assign appropriately low scores to failures encountered when the policy drifts far beyond the demonstration distribution.

A growing body of work therefore studies learned reward models for robot manipulation. These models are typically trained on large-scale robot datasets (O'Neill et al., 2024; Khazatsky et al., 2024), which consist predominantly of expert demonstrations and contain little to no failure data (Atreya et al., 2025). Existing methods thus need to obtain counterfactual supervision from several alternative sources. One category of works construct synthetic counterfactuals from successful trajectories through augmentations such as reversing video segments, early stopping or stalling, and language relabeling (Zhang et al., 2025; Lee et al., 2026; Torne et al., 2026). These can be viewed as a form of post-hoc re-labeling to synthesize failures from successes. Another viable approach is to use real failures collected through deliberate teleoperation or policy rollouts on hardware (Liang et al., 2026). An alternate strategy is to avoid task-specific counterfactual data altogether, estimating rewards zero-shot from frozen vision-language models (Chen et al., 2026b; Ma et al., 2025). In summary, counterfactual supervision is not freely available: synthetic augmentations are inexpensive but provide indirect failure signal, while real failures are faithful to the target environment but expensive, with that cost recurring across new tasks, scenes, and embodiments.

This trade-off among counterfactual sources raises a natural question: **What counterfactuals should we invest in, and how much of each should we collect?** We study this by varying only the source and scale of counterfactual data while keeping the tasks, reward model, training procedure, evaluation protocol, and downstream RL setup fixed. We consider four sources with different levels of fidelity, controllability, and collection effort: intentional teleoperated failures, zero-shot policy rollouts on hardware, deliberately collected human-video failures, and scripted failures generated in real-to-sim environments. These sources also differ in how naturally they scale: teleoperated failures are costly to expand because every trajectory requires direct robot interaction, while policy rollouts, human videos, and especially real2sim counterfactuals can be collected at progressively larger scales. We therefore compare sources at matched data amounts and under realistic scaling regimes, where only the more scalable sources are increased.

As outlined in Figure 1, we use counterfactual data in two ways: to fine-tune Robometer (Liang et al., 2026) with LoRA, and to provide few-shot examples for Generative Value Learning (GVL) (Ma et al., 2025) using a reasoning vision-language model. For both models, we first compare the four sources using the same number of counterfactual trajectories. We then ask whether collecting more policy rollouts, human videos, or real2sim failures can improve the plausibility of reward predictions from the model while keeping the teleoperated data fixed. We also study whether more realistic real2sim renderings lead to better rewards. We use Robometer for the scaling and ablation experiments, since our few-shot GVL setting uses only a small set of examples in its prompt. Finally, we use the selected Robometer variants for RL post-training to test whether better reward predictions lead to better task execution, and the effect of different sources of failures in this setting. In summary, our main contributions are:

- We collect four sources of counterfactuals across five manipulation tasks and construct a held-out, 250-episode cross-policy reward benchmark.

- At matched scale, source choice matters more for trainable Robometer than few-shot GVL; cheaper data helps initially but saturates or degrades, while high-visual-fidelity real2sim benefits more from scaling compared to low-visual-fidelity.
- In real-robot RL, teleop-adapted rewards perform strongest, while real2sim also performs well, suggesting value in controllable failures with matched robot embodiment.

2 Related Work

Reward learning from demonstrations. Learning to estimate reward or value structure from expert behavior is a long-studied alternative to hand-designed task rewards. Classical inverse reinforcement learning infers a reward function that rationalizes demonstrated behavior (Abbeel & Ng, 2004; Ziebart et al., 2008), and later work extrapolates beyond the demonstrator by ranking suboptimal trajectories (Brown et al., 2019). More recent visual representation methods learn embeddings whose distances measure progress toward a goal, including VIP (Ma et al., 2022) and LIV (Ma et al., 2023). These methods benefit directly from the growing supply of successful robot data (O'Neill et al., 2024; Khazatsky et al., 2024), but expert trajectories mainly define what success looks like and give limited guidance for scoring off-distribution states, partial completions, or failures encountered during policy improvement. ReWiND (Zhang et al., 2025) addresses this in the success-only setting by reversing successful video segments to synthesize decreasing progress. These negatives come from temporal manipulation and are thus not necessarily semantically plausible failures. We ask a complementary question: which sources of genuinely off-distribution trajectories are worth collecting with respect to failures, and how their value differs for training reward models and robot policies.

Foundation-model reward and value estimation. A second line uses large pretrained vision-language models as general-purpose reward or value estimators. Generative Value Learning treats a VLM as an in-context value learner, predicting task-completion percentages over shuffled video frames and enabling zero-shot and few-shot progress estimation across tasks and embodiments (Ma et al., 2024). TOPReward targets training-free progress estimation as well, but reads reward signal out of internal token probabilities rather than generating numerical progress values (Chen et al., 2026b). Earlier, RoboCLIP scored rollouts by video-language similarity to a single reference demonstration (Sontakke et al., 2023). These approaches require no task-specific reward training and inherit broad visual-semantic priors, but their dense scores depend on the calibration and reasoning of a frozen model (Bai et al., 2025). The few-shot variant of this regime is one of the two consumers we study: because in-context prompting consumes only a handful of exemplar episodes, it is the setting in which the cheapest counterfactual source has the most room to win, and we measure whether it does.

Learning from failures and trajectory comparisons. A third line of work supervises reward models with data that is not successful. RoboReward trains VLM-based reward models with progress labels together with counterfactual instruction relabeling and video-rewind augmentation (Lee et al., 2026); related process-reward and critic models supervise intermediate task stages for high-precision manipulation and real-world RL (Tan et al., 2025; Zhai et al., 2025). Robometer scales the idea substantially, combining frame-level progress prediction with trajectory-level preferences over successful, suboptimal, and failed rollouts across many embodiments (Liang et al., 2026), and is the trained consumer in our study. Failures also appear as first-class data elsewhere: for failure reasoning and recovery in vision-language-action models (Lin et al., 2025), for autonomous improvement of instruction-following skills (Zhou et al., 2024), and for language-guided recovery policies (Dai et al., 2025). These works establish that real failures are valuable; none of them isolates how much of that value survives when the failures are cheaper. That comparison is our subject.

Where robot counterfactuals come from. The sources we study correspond to existing data-generation traditions. Real teleoperated and policy-rollout data underpins large corpora such as DROID (Khazatsky et al., 2024) and the pooled Open X-Embodiment collection (O'Neill et al., 2024), and distributed real-world evaluation efforts such as RoboArena (Atreya et al., 2025) document how expensive controlled hardware trials remain. Human video has long been used as a cheap demonstration source for robot learning, with RoboCLIP (Sontakke et al., 2023) using one human

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 (Sec. 3.4).

Source	Embod.	Real vis.	On-policy	Scripted	Per-episode cost	Annotation	Collected / task
Teleop	✓	✓	–	✓	$t_{\text{collect}} + t_{\text{replay}}$ (kinesthetic + reset)	human	4 fail. + 1 succ.
Rollout	✓	✓	✓	–	t_{rollout} (policy on hardware)	human	20 π_0 rollouts
Human	–	✓	–	✓	t_{human} (no robot needed)	human	19 fail. + 1 succ.
Real2sim	✓	–	–	✓	t_{asset}/N (amortized setup)	automatic	29 fail. + 1 succ.

video directly as a reward reference. Simulation benchmarks such as LIBERO (Liu et al., 2023) provide cheap, resettable, exactly-labeled rollouts, at the cost of a visual and dynamics gap to the target scene; lifting a specific real scene into simulation with matched assets narrows that gap while preserving scriptability. Generative world models offer a further route: an action-conditioned video model can render a plausible rollout for a counterfactual action sequence without any physical reset (Guo et al., 2025). Each tradition is usually studied in isolation and for policy learning rather than reward learning. We place them under one cost model, one annotation schema, and one evaluation protocol, and ask what a fixed reward-data budget should buy.

3 Study Design

3.1 Research Questions

We design our study around one central question: **What counterfactuals should we invest in, and which sources should we scale given a fixed collection budget?** To isolate the effect of counterfactual data, we keep the reward models and reward-training procedure, task suite, offline evaluation benchmark, and downstream RL training setup fixed. The main experimental variables are the *source* of counterfactual data and the *scale* at which each source is collected under a realistic data-collection budget.

We further break this question into four research questions. **RQ1:** When the amount of data is held fixed across counterfactual sources, how does the source and its associated collection cost and alignment with the target setting affect reward-model performance? **RQ2:** What happens when cheaper and less-aligned counterfactual sources are scaled to larger amounts of data? **RQ3:** For simulated counterfactuals, does visual similarity to the real world matter, as measured by comparing visually high-fidelity and low-fidelity simulation? **RQ4:** Do better reward models improve robot behavior?

3.2 Setup of the Tasks and Robot Platform

We evaluate on five manipulation primitives using a standard DROID (Khazatsky et al., 2024) setup. The tasks are: *Pick up the spoon and put it on the plate* (Pick&Place), *Pick up the sellotape and put it in the grey box* (Pick&Place), *Remove the lid from the pot* (Remove), *Pour the contents of the cup into the grey box* (Pour), and *Stack up the cups together* (Stack).

3.3 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 eliminate the annotation cost entirely. This produces a clear cost ordering across sources; source examples and labels are in Appendix A.1.

Teleoperated failures are the most expensive and most aligned source. An operator uses the Franka's program mode to kinesthetically record each trajectory, 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. We collect one expert trajectory and the rest intentional failure modes per task, all recorded at 15–20 Hz and annotated by a human with partial-credit labels. Per episode, the cost is $t_{\text{collect}} + t_{\text{replay}} + t_{\text{annotate}}$, where collection and replay together make this the most time-consuming source.

Rollout failures require the robot but not an operator guiding it. We deploy our base policy π_0-DROID (Black et al., 2024) 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 teleoperated 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 (Ranawaka et al., 2026) and Agentic Real2Sim (Chen et al., 2026a) 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 construction and failure generation are detailed in Appendix A.1.2 and Algorithm 1.

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

$$\underbrace{t_{\text{collect}} + t_{\text{replay}}}_{\text{teleop}} > \underbrace{t_{\text{rollout}}}_{\text{rollout}} \gg \underbrace{t_{\text{human}}}_{\text{human}} > \underbrace{t_{\text{asset}}/N}_{\text{real2sim (amortized)}}$$

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.

3.4 Reward Models

We study two ways of using counterfactual supervision. We fine-tune Robometer-4B (Liang et al., 2026) with LoRA on data pooled across all five tasks. We also use Generative Value Learning (GVL) (Ma et al., 2025), instantiated with a reasoning vision-language model, and provide task-specific trajectories and their whole-trajectory partial-credit labels as few-shot context. 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. Reward-model hyperparameters and GVL prompting details are given in Appendices A.2 and A.2.2.

3.5 Evaluation benchmark and metrics

We use a custom offline benchmark because it is impractical to perform online RL for every reward-model variant across all control and scaling experiments. Offline metrics are computed on a held-out benchmark of $5 \times 50 = 250$ annotated episodes: for each task, 25 zero-shot rollouts of $\pi_{0.5}$ (Physical Intelligence et al., 2025) and 25 of MolmoAct2 (Fang et al., 2026), 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.

We report three metrics, each measuring a different aspect of reward quality. Kendall τ_{mean} measures how well the reward model ranks whole trajectories, averaged across tasks. MAE_{last} measures the mean absolute error between the predicted final progress and the human-assigned partial credit. Finally, success ROC-AUC measures binary success detection, treating the predicted reward as a classifier.

3.6 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 (Wagenmaker et al., 2025), which learns to steer the latent noise of the frozen base policy. We relabel rewards after each episode using the selected reward model.

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 base Robometer and source-adapted rewards. We evaluate each policy on Pour in 20 blind, randomized trials with matched initial conditions and measure task success directly. Selected reward checkpoints and per-task RL hyperparameters are listed in Appendix A.3.

4 Results

4.1 RQ1: Does counterfactual source matter at matched scale?

We first isolate the effect of counterfactual source by holding the amount of supervision fixed across sources. Figure 2 reports this matched-data comparison for both Robometer and GVL.

For Robometer, the choice of counterfactual source has a clear effect on reward quality. Teleoperated failures provide the strongest and most consistent improvement overall, particularly in trajectory ranking and success detection. Interestingly, human-video counterfactuals remain competitive despite their embodiment gap, and show competitive aggregate results relative to rollout and real2sim counterfactuals. In contrast, policy rollouts do not consistently dominate the other sources despite being on-policy. These results suggest that matching the target embodiment or policy distribution alone is not sufficient: the structure and coverage of the failures contained in a small counterfactual set can also matter.

The picture is different for GVL. Across the same matched-data conditions, performance is considerably less sensitive to the source of the few-shot examples, with the four counterfactual sources producing broadly similar results. Thus, at this scale, source choice appears to matter substantially more for a reward model that is explicitly adapted to the counterfactual data than for a frozen model that consumes the same trajectories through in-context examples.

4.2 RQ2: Can cheaper counterfactual sources benefit from scale?

Given that RQ1 shows substantially greater sensitivity to counterfactual source for the trainable Robometer setting, we study scaling with Robometer. We scale only the sources that are comparatively inexpensive to expand relative to teleoperated failures: policy rollouts, human videos, and real2sim counterfactuals. Figure 3 shows the resulting scaling trends. Specifically, we scale:

- **Policy rollouts:** $5 \rightarrow 10 \rightarrow 20$ trajectories per task.
- **Human videos:** $5 \rightarrow 10 \rightarrow 20$ trajectories per task.

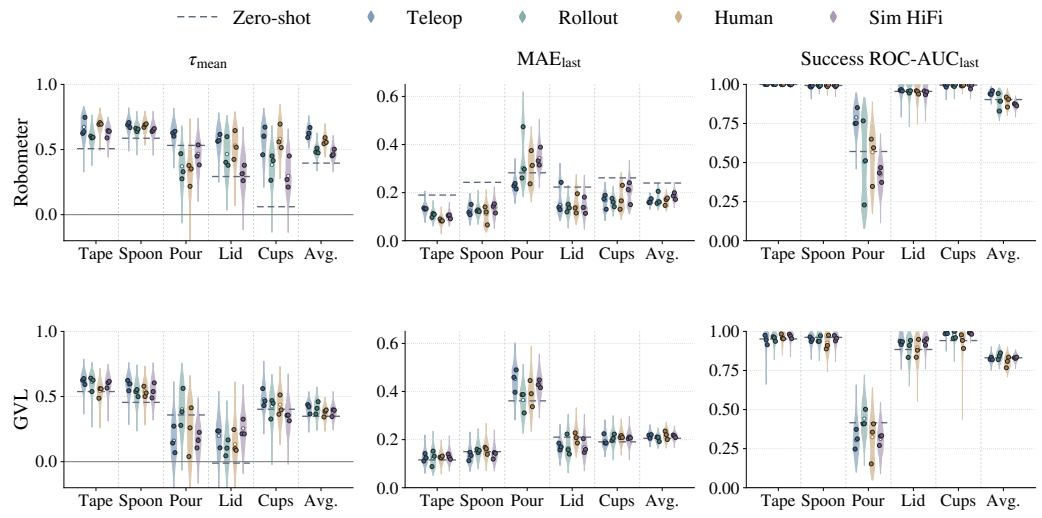

Figure 2: **Matched-data control across counterfactual sources (RQ1).** Robometer and few-shot GVL evaluated on the same held-out benchmark using five trajectories per task from each source. Columns show trajectory ranking (τ_{mean}), final-progress error (MAE_{last}), and success ROC-AUC; dashed lines indicate zero-shot performance.

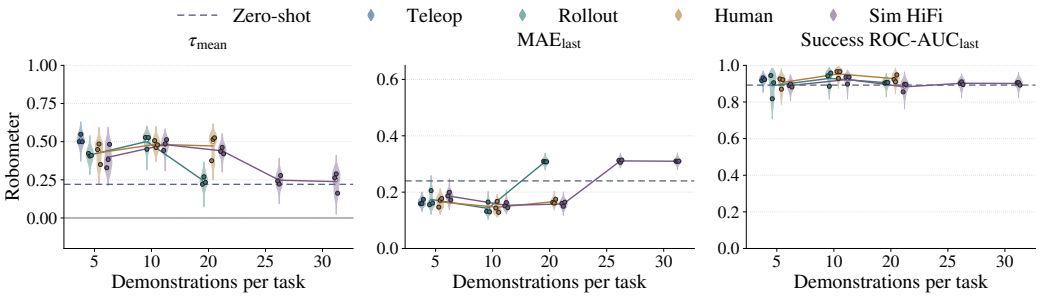

Figure 3: **Scaling Robometer with cheaper counterfactual sources (RQ2).** Performance as the number of training trajectories per task increases for rollout, human-video, and real2sim (HiFi) sources. Teleoperated data and the zero-shot Robometer are shown as fixed references; lower MAE_{last} is better.

- **Real2sim (HiFi):** $5 \rightarrow 10 \rightarrow 20 \rightarrow 25 \rightarrow 30$ trajectories per task.

The results show that simply adding more counterfactuals does not produce monotonic improvements. **Policy rollouts** improve from 5 to 10 trajectories across the reward metrics, but performance degrades substantially when scaled to 20. This suggests that additional on-policy data is not necessarily beneficial if the newly collected rollouts do not add useful coverage of failure behavior. **Human-video counterfactuals** also benefit from the initial increase in scale, with the largest gains appearing by roughly 10 trajectories per task; beyond this point, performance largely saturates, with only small changes at 20. **Real2sim counterfactuals** show a similar pattern: scaling initially improves reward quality, but the gains saturate around 10–20 trajectories and decline at larger scales. One possible explanation is that our scripted simulator begins to add redundant rather than meaningfully new failure modes as the dataset grows.

Overall, Figure 3 suggests that **scale helps primarily when additional data introduces useful variation**. Across all three sources, the strongest gains occur early, while simply increasing the number of counterfactuals beyond that point provides little benefit and can even hurt reward-model quality.

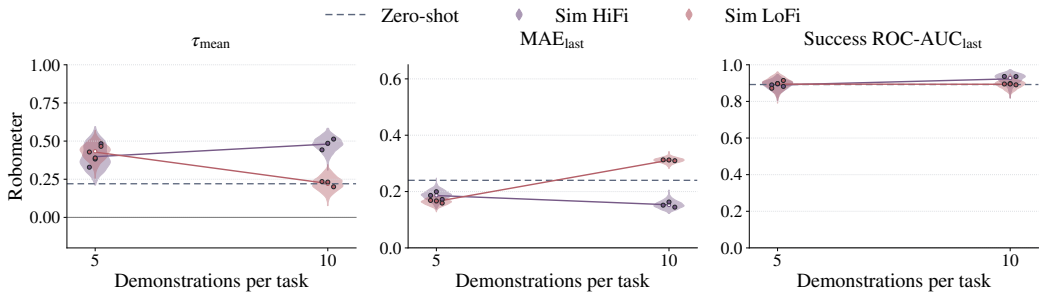

Figure 4: **Real2sim visual-fidelity ablation (RQ3).** Robometer adapted with high-fidelity (HiFi) and low-fidelity (LoFi) real2sim counterfactuals at five and ten trajectories per task, using matched failure-mode distributions and partial-credit buckets. Dashed lines show zero-shot performance.

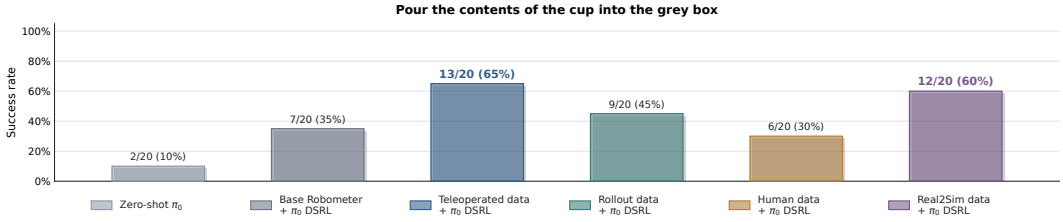

Figure 5: **Real-robot RL results on Pour (RQ4).** Success rates over 20 trials for zero-shot π_0 and DSRL using base Robometer and rewards adapted with teleoperated, policy-rollout, human-video, or real2sim counterfactuals.

4.3 RQ3: Does simulation visual fidelity matter?

We study visual fidelity as an ablation of the real2sim source. We compare a *high-fidelity* (HiFi) setup with Gaussian splatting and textured assets in OmniGibson (Li et al., 2023) against a *low-fidelity* (LoFi) setup rendered in MuJoCo (Todorov et al., 2012). We keep the failure-mode distribution uniform and match the partial-credit reward buckets across the two settings. Figure 4 shows little difference at five demonstrations per task, but a clearer separation at ten: HiFi gives higher trajectory-ranking correlation, lower reward error, and slightly better success detection. This suggests a growing benefit from the HiFi real2sim pipeline as data is scaled, although the simulators also differ (Appendix A.1.2).

4.4 RQ4: Do better reward models improve robot behavior?

Real-world RL requires substantial robot interaction and evaluation effort, so we use the offline study to select reward variants for post-training on Pour. Figure 5 shows the finalized plotted conditions under the same RL setup, with 20 evaluation trials each. Selected checkpoints and RL hyperparameters are given in Appendix A.3.

Zero-shot π_0 succeeds in $2/20$ trials, and DSRL with base Robometer reaches $7/20$. Teleop adaptation improves this to $13/20$, consistent with its stronger offline results, while rollout adaptation reaches $9/20$. Rollouts improve over the base reward but less than teleop, possibly because their failure behaviors are not deliberately controlled. Real2sim also performs well at $12/20$, while human-video adaptation reaches $6/20$ despite competitive offline results. These observations suggest that controllable failures with a matched robot embodiment can provide useful RL supervision despite the visual gap; the human embodiment gap may also contribute to the weaker online result, although the comparison does not isolate these factors.

Figure 6 gives two qualitative examples: teleop adaptation preserves partial credit on an incomplete cup-stack, while real2sim HiFi recognizes a completed pour. Additional examples are in Appendix A.4.

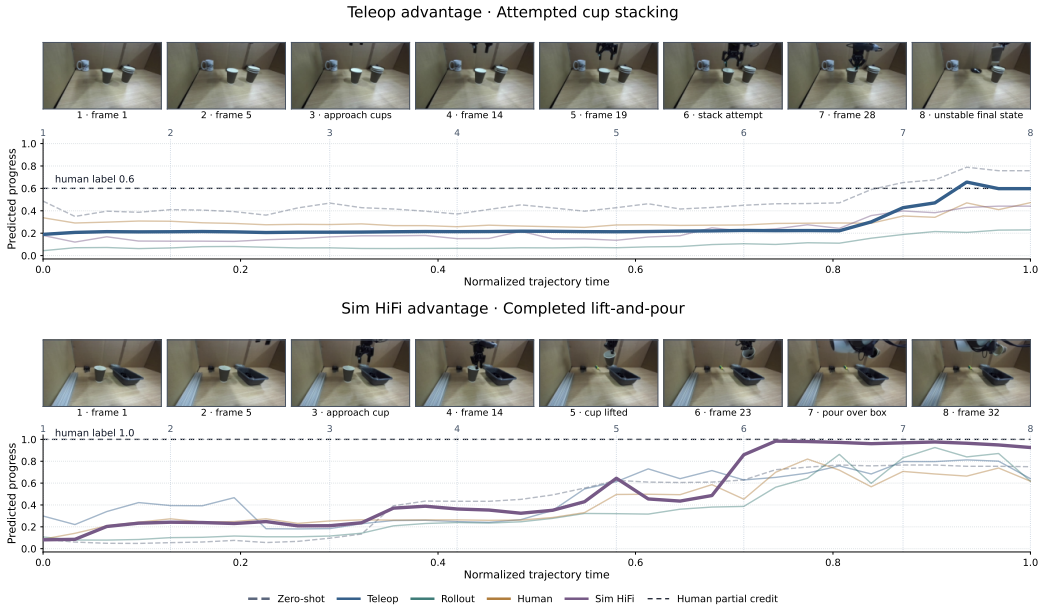

Figure 6: **Qualitative reward predictions.** Top: partial-progress cup-stacking attempt (human credit 0.6), where the teleop-adapted model recognizes progress without assigning full success. Bottom: completed lift-and-pour (human credit 1.0), where the real2sim HiFi-adapted model predicts high completion progress. Curves compare the zero-shot and source-adapted Robometer rewards.

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. Teleoperated 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. Our pouring experiments also showed that these choices of the source and scale of counterfactuals are crucial for adapting robot manipulation policies in the real-world through reinforcement learning

Our study covers five tabletop tasks on one robot embodiment, with online RL evaluated on Pour over 20 trials per condition. Collection costs are setup-dependent, and the HiFi/LoFi ablation changes the simulator as well as visual fidelity. Broader task evaluations, source mixtures, and direct measures of failure coverage are important topics for future work. We showed that useful failure supervision can come from several sources, each offering a different balance between collection effort and reward quality. Deliberate robot failures offer a strong starting point, and cheaper sources can become competitive at larger scales. We hope this study helps guide how to allocate collection effort when learning rewards for robot manipulation.

AI use statement

We used generative AI tools to assist with code generation and writing. AI tools were not used to come up with ideas for research, just for helping with writing code, and assistance in editing the paper. The authors reviewed and cross-checked all AI-assisted content, tested generated code where applicable, and take responsibility for the final work.

Ethics statement

All experiments use a fixed-base manipulator performing benign tabletop tasks in a controlled setting. Human videos were recorded with informed consent and shared examples are anonymized.

Benchmark annotations followed a consistent labeling protocol, and RL evaluations used blind, randomized trials with matched initial conditions to ensure fair comparison.

Reproducibility statement

The appendix provides data collection procedures, reward-model and RL hyperparameters, and checkpoint details. We use publicly available base models and will release benchmarks and code upon publication.

References

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. 3

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. 2, 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. 3

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pp. 1577–1594. PMLR, 2023. 2

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. 3

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*, 2026a. 5

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*, 2026b. 2, 3

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. 3

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. 6

Yanjiang Guo, Lucy Xiaoyang Shi, Jianyu Chen, and Chelsea Finn. Ctrl-World: A controllable generative world model for robot manipulation. *arXiv preprint arXiv:2510.10125*, 2025. 4

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. 2, 3, 4

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. 2, 3

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pp. 80–93. PMLR, 2023. 8

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. 2, 3, 5

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

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. 4

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. VLA-RL: Towards masterful and general robotic manipulation with scalable reinforcement learning. *arXiv preprint arXiv:2505.18719*, 2025. 2

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. 3

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. 3

Yecheng Jason Ma, Joey Hejna, Ayzaan Wahid, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, et al. Vision language models are in-context value learners. *arXiv preprint arXiv:2411.04549*, 2024. 3

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. 2, 5

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. 13

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. 2, 3

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. 6

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. 2

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. 5, 13

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. 2

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. 2

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. 3

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. 3

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012. 8

Marcel Torne, Anubha Mahajan, Abhijnya Bhat, and Chelsea Finn. Freeform preference learning for robotic manipulation. *arXiv preprint arXiv:2606.32027*, 2026. 2

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. 6

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, et al. RLinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation. *arXiv preprint arXiv:2509.15965*, 2025. 2

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. 3

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. 2, 3

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. 3

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. 3

A Appendix

A.1 Counterfactual Sources and Annotation

We provide additional details on counterfactual collection, partial-credit annotation, and the real2sim failure-generation pipeline. Representative trajectories from all four sources, together with their corresponding partial-credit annotations, are available in our **counterfactual data explorer**.

The explorer contains teleoperated failures, policy rollouts, human-video counterfactuals, and real2sim trajectories across the five manipulation tasks.

A.1.1 Partial-Credit Annotation

We assign partial-credit labels from $\mathcal{P} = \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$, where 0.0 indicates no meaningful task progress and 1.0 indicates successful task completion. Intermediate labels capture different levels of partial task completion. Teleoperated, policy-rollout, and human-video trajectories are manually annotated, while real2sim labels are obtained automatically from simulator state and task-specific milestone predicates.

The same annotation scale is used across all four counterfactual sources. Additional source-wise examples and their assigned labels can be inspected in the data explorer linked above.

A.1.2 REAL2SIM SCENE CONSTRUCTION

We use one successful kinesthetic demonstration per task to construct corresponding scenes in two simulators: OmniGibson for high-fidelity (HiFi) real2sim and MuJoCo for low-fidelity (LoFi) real2sim. The resulting scenes preserve the target robot embodiment, task objects, initial configurations, and task semantics, while differing in their rendering pipelines and simulator implementations.

High-fidelity real2sim. We reconstruct the real workspace using components of the SimFoundry pipeline (Ranawaka et al., 2026). Calibrated exterior and wrist camera recordings are used to estimate scene geometry, object poses, and camera alignment. Objects are segmented and reconstructed as textured 3D meshes, with physics-ready collision geometry generated before importing them into OmniGibson. A Gaussian-splat reconstruction of the workspace provides the background rendering.

We additionally define appropriate collision geometry for hollow and thin objects, including cups, the grey box, sellotape, and the pot lid, to support task interactions such as grasping, insertion, and pouring. The scene is aligned with the physical robot using camera calibration and robot kinematics.

Low-fidelity real2sim. The MuJoCo scene is derived from the corresponding HiFi scene rather than reconstructed independently. We transfer the object meshes, initial poses, robot configuration, camera calibration, and task predicates into MuJoCo. The same Franka Panda and Robotiq gripper are used in both setups. This produces a lower-fidelity rendering of the same task configuration while maintaining correspondence between the selected HiFi and LoFi trajectories.

Since the two settings also use different simulators, this ablation compares the complete HiFi and LoFi real2sim pipelines rather than isolating rendering fidelity alone.

A.1.3 SUCCESSFUL DEMONSTRATION REPLAY

For each task, we first collect one successful kinesthetic demonstration on the physical robot. The recorded trajectory contains joint positions, gripper commands, end-effector poses, and camera calibration.

We remove sufficiently long idle portions of the demonstration while preserving contact-rich manipulation, grasp and release events, and task-specific motion windows. A short settling period is retained where necessary. Each compressed demonstration is replayed in both simulators and accepted only when it satisfies all ordered task milestones, reaches the successful terminal state, and remains physically stable.

This successful replay serves as the reference trajectory for generating counterfactual failures.

A.1.4 FAILGEN: REAL2SIM FAILURE GENERATION

Our failure-generation pipeline follows a replay-and-perturb approach, using the successful demonstration as a reference and generating controlled deviations from different task stages. This follows the general idea of trajectory-based data generation used in works such as MimicGen (Mandlekar et al., 2023).

We sample changes to the initial object configuration and retarget the reference trajectory relative to the manipulated object. For an end-effector pose T_{WE} and reference-object pose T_{WO}, the retargeted pose is computed as

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

where W, O, and E denote the world, reference-object, and end-effector frames, respectively. This preserves the relative end-effector/object configuration after changing the object's initial pose.

Failures are generated using parameterized operators including `truncate_before`, `no_grip`, `collision_nudge`, `under_actuate`, `halt_mid_transport`, `early_release`, and `goal_offset`. We preserve the valid trajectory prefix and modify the selected task stage, followed by inverse kinematics and command interpolation.

Automatic reward assignment. Each task is divided into five ordered physical milestones, corresponding approximately to object contact, secured grasp, lifting or partial manipulation, alignment or placement, and task completion. These milestones are task-specific and are detected from simulator state rather than from the intended failure operator.

Let k be the length of the longest ordered milestone prefix observed during execution. The assigned partial credit is

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

Milestones are sticky: once achieved, they remain achieved, but later milestones count only if the preceding milestones have been observed. Candidate failures are retained only when both simulators agree on the assigned credit, the task remains unsuccessful, and the final state is stable.

Algorithm 1 FailGen: 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 HiFi/LoFi videos, metadata, and rewards
return $\mathcal{D}^-, \tau^+$

The final dataset used in this paper contains **29 selected failures and one successful replay per task**. We select candidates to cover different partial-credit levels and failure behaviors, followed by visual-diversity selection. The same selected episode identities are retained in HiFi and LoFi for the matched real2sim ablation.

The generation pipeline requires human effort during the initial task setup, successful demonstration collection, scene reconstruction, collision geometry preparation, and milestone definition. Once these components are prepared, candidate generation, simulator replay, reward assignment, valida-

Table 2: **Robometer adaptation hyperparameters.** The same configuration is used across counterfactual sources.

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 selected trajectory count
Maximum frames / training sample	8

tion, and data export are automated. Individual generated failures are not manually teleoperated or annotated.

A.2 Reward Model Tuning Hyperparameters

We provide the Robometer adaptation settings and the model and prompting configuration used for the GVL baseline. All source comparisons use the same adaptation procedure within each reward-model setting.

A.2.1 Robometer Model and Training Setup

We initialize from `robometer/Robometer-4B`, with the Qwen3-VL-4B-Instruct backbone. We freeze the vision encoder and adapt the language-model backbone using LoRA. The preference/ranking, per-frame progress, and binary success heads are trained jointly.

The LoRA configuration uses rank $r = 16$, scaling factor $\alpha = 32$, dropout 0.05, and no bias adaptation. Adapters are applied to the language-model modules `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`. No LoRA adapters are attached to the vision encoder.

Each prediction head consists of a linear projection to half the backbone hidden width, LayerNorm, GELU, dropout 0.1, and a final output projection. The preference and success heads output one logit each, while the progress head predicts ten discrete progress bins.

Sampling and supervision. Preprocessing retains at most 32 frames per trajectory, from which at most eight are sampled for each training example. Trajectories with fewer than five frames are excluded. The loader samples preference pairs using rewind, same-task suboptimal, different-task, and reverse-progress strategies. Partial-credit trajectories are paired when their rewards differ by at least 0.2.

For a trajectory with partial credit $p < 1$, the label supervises the final original frame when it is included in the sampled frames; intermediate frames are not assigned an artificial linear progress curve from the terminal label. Additional progress supervision comes from successful trajectories and synthetic transformations.

Each training pair contributes preference, progress, and success losses:

Table 3: **GVL baseline configuration.** Seeds change frame-presentation order rather than model weights.

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 demonstration
Image detail	Low
Maximum response tokens	4096
Prompt version	`gvl-trajectory-average-reward-v2`
Seeds	42, 43, 44
Output	JSON containing per-frame completion percentages

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

We use binary cross-entropy for the Bradley–Terry preference objective, cross-entropy over ten discrete progress bins for progress prediction, and masked binary cross-entropy for success prediction. All three terms have unit weight.

The training duration follows $\texttt{max_steps} = 3N$, with N denoting the total number of selected training trajectories across the five tasks. We train each condition with seeds 42, 43, and 44. Smaller scaling subsets are nested within the corresponding larger subsets, while the LoFi real2sim experiments use the same selected episode identities as HiFi.

For offline evaluation, we use the 250 held-out benchmark trajectories described in Sec. 3.5. Each trajectory is uniformly subsampled to 30 frames. The reported metrics use 10,000 bootstrap samples.

A.2.2 GVL Model and Prompting Details

We implement the Generative Value Learning (GVL) baseline using `gpt-4o` through multimodal API requests. Unlike Robometer, GVL is frozen and does not receive parameter updates. Counterfactual trajectories are supplied as labeled in-context examples.

We use the same task description and evaluation trajectories across source comparisons. In the matched-data experiment, the zero-shot condition receives no demonstrations, while each few-shot condition receives five same-task demonstrations from its respective source.

For each target trajectory, up to 30 frames are uniformly sampled while retaining temporal endpoints. Up to 30 frames are also sampled from each context demonstration. Frames are supplied as JPEG images and deterministically shuffled before inference. The prompt includes the initial scene as a zero-progress reference and asks the model to predict a task-completion percentage between 0 and 100 for every shuffled target frame.

Each context demonstration is accompanied by one human-assigned whole-trajectory average-behavior partial-credit label. The prompt explicitly distinguishes this trajectory-level label from a per-frame or terminal-state target.

The model is instructed to return a JSON array containing the frame number, frame description, and task-completion percentage for every requested frame. We validate the output, divide percentages by 100, and restore chronological order to obtain the predicted progress curve. Invalid or incomplete outputs are rejected and retried. Successful responses are cached using the model, prompt version, context selection, and image-content hashes.

A.3 Per-Task RL Hyperparameters

We report the DSRL configuration used for the Pour task, *Pour the contents of the cup into the grey box*, together with the selected reward checkpoints and evaluation protocol.

Table 4: **Pour-task policy and observation configuration.**

Component	Configuration
Frozen base policy	`pi0-DROID`
Flow horizon	10
Executed action chunk	8 actions
SAC noise action	8×32, bounded to $[-2, 2]$
Left exterior camera	`34401917`
Wrist camera	`10401831`
Right exterior camera	`36002232`; recorded but unused
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.3.1 Policy and Observations

We initialize from the frozen `pi0-DROID` policy. Only the DSRL SAC noise actor and critics are trained. The π_0 flow horizon is ten, while SAC predicts eight noise rows and executes eight actions per transition. Noise rows 0–1 are copied into the unused horizon positions 8–9.

The SAC noise action has shape 8×32 with bounds $[-2, 2]$. The policy receives the physical left exterior camera, wrist camera, joint/gripper state, and the exact Pour task instruction.

The reward model uses `observation/exterior_image_1_left`. The physical right camera is recorded but is not selected for this training run.

A.3.2 DSRL Actor, Critics, and SAC

The DSRL actor and critic use Robometer transformer modules. Each observation-key featurizer has width 384. The transformer has hidden width 256, eight attention heads, six encoder layers, GELU activations, and no transformer dropout.

The actor feature MLP has dimensions $[512, 512]$, uses ReLU, and has no layer normalization. The critic feature MLP has dimensions $[768, 512]$, uses ReLU and layer normalization, and uses first-token pooling. We use four critics, sample two per update, and apply mean critic reduction.

A.3.3 Warmup and Online Training

The episode limit is 500 physical control steps, with a target control rate of 15 Hz. We first collect a reusable warmup using uniform noise. Although the requested warmup length is 5,000 physical steps, the saved warmup contains 11 complete episodes, corresponding to 5,117 physical steps and 645 replay transitions. The additional 117 steps arise from saving complete episodes.

Every reward-model condition relabels this same saved warmup independently from scratch. We then train for 10,000 additional physical interaction steps, giving a target of

$$N_{\text{physical}} = 5{,}117 + 10{,}000 = 15{,}117. \tag{4}$$

An atomic checkpoint is saved after every completed training episode. The base policy, warmup data, interaction budget, and SAC configuration are kept fixed across reward variants.

A.3.4 Robometer Reward and Selected Checkpoints

We use either the base Robometer-4B reward model or a compatible PEFT LoRA adapter loaded with its base model and Robometer heads. At the end of each episode, the reward model scores trajectory prefixes from the left exterior camera and returns predicted progress and success probabilities.

Table 5: **DSRL/SAC hyperparameters for Pour.**

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: **Reward-model checkpoints considered for Pour RL.** Each adapted checkpoint is loaded with the corresponding Robometer base model and prediction heads.

Reward variant	Checkpoint identifier
Base Robometer	Robometer-4B (zero-shot)
Teleoperated	`teleop5_seed44`
Policy rollout	`rollout10_seed44`
Human-video	`human10_seed44`
Real2sim (HiFi)	`hifi10_seed42`

The stored dense reward combines the human sparse reward (-1 per transition, with terminal reward 0 on success) with Robometer-predicted progress. Human-provided success and failure labels are authoritative; predicted model success is used only for diagnostics. If reward-server inference fails, the episode is dropped instead of mixing sparse-only and Robometer-shaped transitions.

A.3.5 RL Evaluation and Videos

We evaluate each trained reward variant using 20 real-robot rollouts. The SAC actor is deterministic during evaluation. Evaluation does not add data to replay and performs no gradient updates. Task success is measured directly from the physical execution rather than from the reward model's predicted success.

Sped-up real-robot rollouts for the evaluated policies and their different reward conditions are available in our **RL rollout explorer**.

A.4 Additional Qualitative Reward Results

We provide additional qualitative reward predictions across tasks, counterfactual sources, and levels of partial task completion. The **reward benchmark explorer** contains the computed reward curves and allows comparisons between the zero-shot Robometer and the source-adapted reward models.

The explorer includes predictions on the held-out evaluation trajectories and their human-assigned partial-credit labels. It complements the qualitative examples shown in Figure 6 and allows inspection of reward behavior beyond the examples included in the main paper.