RoboRef: A Foundation Reward Model Built for Robot RL, Not Offline Metrics

Platon Karageorgis[1,2]* **Leonardo Barcellona**[1,3] **Rahaf Aljundi**[2]
Stratis Gavves[1] **Christian Gumbsch**[1]

[1]University of Amsterdam [2]Toyota Motor Europe [3]NXAI

Abstract

Reinforcement learning (RL) offers a general route for robots to learn to solve tasks from their own experience. However, designing the reward that guides this learning remains a bottleneck. Recent work therefore replaces hand-crafted reward functions with reward models fine-tuned from vision-language models (VLMs) on large robotic datasets, reporting strong prediction accuracy on offline benchmarks. Yet whether this accuracy carries over to successful RL training remains underexplored. We identify two gaps shared by most VLM reward models. First, the annotated robot data these models train on mostly contain successful behavior, which leaves them poorly equipped to judge the failures produced by a policy being optimized. Second, most models penalize over- and under-prediction of the reward equally, although false positives are the main cause of reward hacking, which poses a severe problem for downstream RL. We introduce RoboRef to close these gaps. To train RoboRef, we re-curate RBM-1M, a large-scale robot reward dataset, densely annotating its failures and adding new ones so that the model learns to score the failures it might encounter during RL. Furthermore, we introduce an asymmetric loss that penalizes over-prediction more harshly than under-prediction, yielding conservative rewards that are harder for a policy to exploit. We evaluate on offline metrics and downstream RL across three simulated benchmarks and two real-world manipulation tasks. We observe that the models that lead on offline benchmarks can fail to train a policy at all. RoboRef trains competent policies where these baselines never learn and delivers the strongest downstream performance, making it, to our knowledge, the state-of-the-art VLM reward model for robot RL.

1 Introduction

Reinforcement learning (RL) offers a principled framework for teaching robots to act through trial and error. However, it requires a reward function that ideally specifies not only what the robot should achieve, but also how to get there [2]. For robot learning, designing this signal by hand is laborious and brittle. Dense, shaped rewards require task-specific engineering [3, 4] and privileged state information such as object poses or goal distances, which must be estimated outside simulation and can introduce perception errors into the reward. Sparse success rewards are easier to specify but hard to optimize [5, 6]. In both cases, a reward designed for one task rarely transfers to another, so every new task reopens the same engineering problem.

Pre-trained vision-language models (VLMs) have recently shown broad visual and semantic understanding that transfers across domains with little task-specific tuning [7–10]. This success, together with progress in automated reward engineering [11], raises a natural question: can VLMs provide meaningful rewards and resolve the reward-definition bottleneck in robot RL?

*Correspondence to: p.karageorgis@uva.nl

40th Conference on Neural Information Processing Systems (NeurIPS 2026). Workshop: RoboPAD: Post-Training Adaptation of Robot Foundation Models.

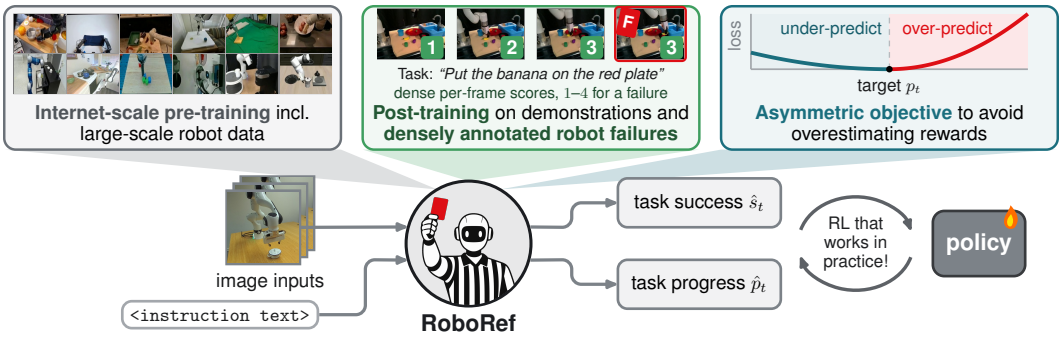


Figure 1: **RoboRef** builds on the internet-scale pre-training of Robometer [1] (left), adds post-training on more demonstrations and *failures with dense per-frame labels* (middle: a failed episode, "Put the banana on the red plate", with the scores our pipeline assigns), and trains with an *asymmetric loss* that damps under-estimation but not over-estimation (right). At inference, RoboRef predicts per-frame success $\hat{s}_t$ and progress $\hat{p}_t$, either of which can serve as the reward for RL.

Several recent works fine-tune VLMs into exactly such reward models [12, 13, 1, 14, 15]. All of these models report strong offline reward-quality metrics. However, whether the *released* models produce rewards that work inside an RL loop is rarely established reproducibly, since real-world setups are hard to compare across laboratories and the simulated evaluations that exist mostly adapt the reward to the target domain first (Sec. 2). Strong offline metrics alone do not guarantee that a reward model can train a policy.

Our experiments confirm this: VLM reward models that score well on offline reward prediction can fail completely to train a policy. We argue that current training strategies may serve offline metrics while hindering deployment in downstream RL, for two reasons.

First, failures are underrepresented in reward-model training. Robometer [1], among the strongest VLM reward models and trained on over a million robot trajectories, supervises its reward predictions only on successful trajectories. Other approaches train only on expert demonstrations [12] or manufacture negatives by modifying successful videos [13, 14]. None assembles a large corpus of failed robot behavior carrying dense, per-frame progress labels. Thus, failures are largely missing from the data that supervises these reward predictions, and rating failed behavior lies outside the models' training distributions. Yet failures are exactly what an optimizing policy produces, especially early in training, when the reward matters most.

Second, VLM reward models are trained with symmetric objectives: over- and under-prediction count as equal errors [1, 13–15]. However, under an optimizing policy, the two errors are not equally harmful [16]. A false positive plants a spurious goal that the policy will exploit, the failure mode known as *reward hacking*. A false negative, on the other hand, only slows learning. A symmetric objective is a sensible choice for offline accuracy, but not necessarily for RL.

We introduce **RoboRef**, a VLM reward model built for RL that addresses both gaps (Fig. 1). First, to bring failures into the training distribution, we train RoboRef on RBM-1M+, our re-curation of Robometer's training corpus RBM-1M, extended by simulated and real-world failures that carry progress labels for each frame. Second, we replace the usual symmetric loss with an asymmetric loss that penalizes over-prediction more heavily, making the model conservative and harder to exploit.

Our contributions are:

- **RoboRef, a conservative VLM reward model** initialized from Robometer and trained on annotated successes and failures with an asymmetric objective that penalizes over-prediction more.
- **RBM-1M+**, a re-curation of RBM-1M with 116,786 densely labeled failure trajectories, to our knowledge the **first training corpus at scale whose failures carry per-frame progress**.
- An **evaluation across offline metrics and downstream RL** on simulated benchmarks and two real-robot tasks, with every reward model queried zero-shot, showing that offline reward quality does not necessarily predict RL performance and that RoboRef performs best downstream.

Table 1: **Foundation reward models for robot learning**, as reported in their own papers. *Frozen RL*: in the RL setup its authors propose, the released model trains a policy without fine-tuning on the target task or a ground-truth reward. *Modified*: failures synthesized from successful demonstrations (e.g. relabeling, rewinding, image editing, trimming).

					Frozen RL	
Model	**Backbone**	**Data**	**Reward**	**Failures in training**	**Sim**	**Real**
Robometer	Qwen3-VL 4B	1M traj.	progress, success	● real (preference head only), modified	✗	✓
RoboReward	Qwen3-VL 4/8B	45k ep.	progress	● real (episode-level labels), modified	✗	✓
Robo-Dopamine	RoboBrain2.0 3/8B	100k traj.	progress	● modified	✗	✗
Robo-Dopamine 2.0	Qwen3-VL 8B	400k pairs	progress	● modified	✗	✗
LRM	Qwen3-VL 8B	24 sources	progress, success	● none	✓	✗
RoboRef (ours)	Qwen3-VL 4B	649k clips	progress, success	● real + sim (dense per-frame), modified	✓	✓

Code, model weights, and RBM-1M+ will be released at `https://roboref-reward.github.io`.

2 Related Work

Reward modeling. A long line of work learns the reward from data instead of writing it by hand. Inverse RL recovers a reward from expert demonstrations [17, 18], and later methods learn dense rewards from passive, action-free videos [19–21]. A second strand learns from human preferences over pairs of trajectories [22], an idea later applied to LLMs as RL from human feedback (RLHF) [23]. More recently, pre-trained foundation models have themselves become the source of feedback, either by scoring observations zero-shot [24, 25] or by supplying preferences [26, 27].

Vision-language reward models for robotics. Our most direct precursors fine-tune VLMs to score a video of a robot attempting a language-specified task. They differ in the rewards they predict and in how failures enter training (see Tab. 1). Importantly, evidence that these models train policies without task-specific fine-tuning or ground-truth reward is thin. Robometer [1], from which we initialize RoboRef, predicts per-frame progress and success. Real failures enter its training only through an auxiliary preference loss that ranks pairs of trajectories. RoboReward [13] scores entire episodes with a discrete reward from 1 to 5. It draws its negatives mostly from simple modifications of successes, such as clipping or relabeling. Both Robometer and RoboReward evaluate their released models in RL only on real robots, which are hard to compare across labs, and use separately trained in-domain models in simulation. Robo-Dopamine [12] predicts progress toward a goal image and trains only on expert demonstrations. For RL, it is adapted to each new task from a demonstration. Its successor Robo-Dopamine 2.0 [14] synthesizes failures by editing successes and is used in RL only alongside a ground-truth task reward. LRM [15] provides dense progress and completion rewards. It is the exception that demonstrates simulated RL with its frozen model, but in the real world it uses its reward only to filter rollouts. Trained on RBM-1M+, RoboRef is, to our knowledge, the first VLM reward model to supervise its reward predictions on failed attempts with dense per-frame progress labels. It also replaces the symmetric training objective these models share with one that penalizes over-prediction more.

Learning from failures. Most robot learning relies on successful data, and failure, when used at all, is something to detect or correct. RoboFAC [28] annotates failed trajectories with question-answer pairs for analysis and correction. FailSafe [29] generates failures with recovery actions. $\pi_{0.7}$ [30] conditions a policy on hand-labeled mistakes. All three capture whether and why an attempt failed, but a reward must score how far it progressed.

Reward hacking. Optimizing a policy against a learned reward invites reward hacking, where the policy exploits errors in the reward instead of solving the task [31, 32]. In RLHF, true performance degrades past a certain point even as the proxy reward keeps climbing [33]. For VLM reward models, false positives mislead an optimizing policy far more than false negatives [16], which motivates our asymmetric objective.

3 RoboRef

RoboRef is a foundation reward model for robot RL, built on the pre-trained Robometer [1] (Sec. 3.1). To teach it to judge the failures an optimizing policy produces, we train it on RBM-1M+, which adds dense per-frame progress labels to over 100k curated failure trajectories (Sec. 3.2). Its asymmetric objective penalizes reward over-estimation more heavily, making it harder for a policy to exploit (Sec. 3.3). Section 3.4 describes further model details and the training setup.

3.1 Preliminaries: Robometer

Robometer fine-tunes Qwen3-VL-4B-Instruct [34] to score a trajectory given as $T{=}8$ evenly sampled frames and a task description. A learned token follows each frame. Because the backbone is causally masked, the hidden state at the t-th token summarizes the trajectory up to frame t, so all per-frame predictions come from a single forward pass. Two heads read these states and predict the rewards a policy optimizes. The *progress* head predicts a distribution $\hat{p}_t$ over discretized progress bins on $[0, 1]$. The *success* head predicts the probability $\hat{s}_t$ that the task is complete. A third *preference* head judges which of two attempts at a task is better through a ranking loss. Training minimizes $\mathcal{L} = \mathcal{L}_{\text{prog}} + \mathcal{L}_{\text{succ}} + \mathcal{L}_{\text{pref}}$ on RBM-1M, roughly one million trajectories across twenty-one embodiments. The progress and success heads are supervised only on successful trajectories, where frame t receives progress t/T, and on modifications of them such as rewound clips. Real failures enter training solely through the preference head, as the trajectory comparison includes unsuccessful attempts. The progress and success losses penalize over- and under-prediction equally.

3.2 RBM-1M+: A failure-annotated dataset

The success head is easy to supervise on failures, since a failed attempt is incomplete at every frame. The progress head is not, since a failure does not advance steadily and its progress cannot be read off the frame index. We therefore score every failure frame on the discrete rubric $d \in \{1, \dots, 5\}$ that RoboReward-4B [13] uses to rank whole trajectories, mapped to progress $p = (d - 1)/4$. Since a genuine failure never completes the task, $d \leq 4$ and p never reaches 1. Three routes produce these scores depending on the data source.

(i) Simulator state. We *generate* failures in the simulated robot manipulation suites MetaWorld [3] and FailSafe [29]. Here, progress comes straight from privileged state. A scripted expert runs to a task-specific injection point, then switches to a noise mode that induces a particular failure, such as dropping a grasped object or stalling a mechanism. We keep the episode only if a state-based detector confirms it ended at the target score (examples in Fig. 7 and App. B).

(ii) A two-stage VLM–LLM pipeline. For failures we have only as video, i.e., the real-world clips already in RBM-1M or real-world failures we enrich the dataset with such as DROID [35] failures, a VLM describes each frame, naming the task-relevant objects, their state, and what the gripper is doing. An LLM, which never sees pixels, then walks the clip in order and returns a rubric level for each frame, reading its description together with all earlier descriptions and scores. Separating description from scoring proved substantially more reliable than asking a single model to watch the whole video and output a number, which led to frequent hallucinations and fabricated task progress (App. C.3). Figure 2 unrolls the pipeline over one example episode of DROID. More details are in Appendix C.

(iii) Monotone ramps. RoboReward [13] supplies partial executions, i.e., truncations of successful behavior. Their progress climbs steadily, so no model is needed to score them. We spread the clip's own score p evenly across its frames, rising from 0 at the first frame to p at the last.

The rubric scores of routes (i) and (ii) still form a staircase over 0, 0.25, 0.5 and 0.75, whereas successes keep Robometer's smooth t/T ramp. To prevent the model from separating failures from successes by shape alone, we interpolate each failure's progress linearly between the frames where its level changes, so that failures and successes share one continuous target on $[0, 1]$.

The resulting corpus holds 648,714 per-frame labeled clips, 116,786 of them failures (dataset breakdown in App. A). The failures Robometer already held become direct training signal for both reward heads for the first time. We generate 24,402 MetaWorld and 2,098 FailSafe failures ourselves and add 5,503 failed DROID episodes, a source RBM-1M carried only as expert demonstrations.

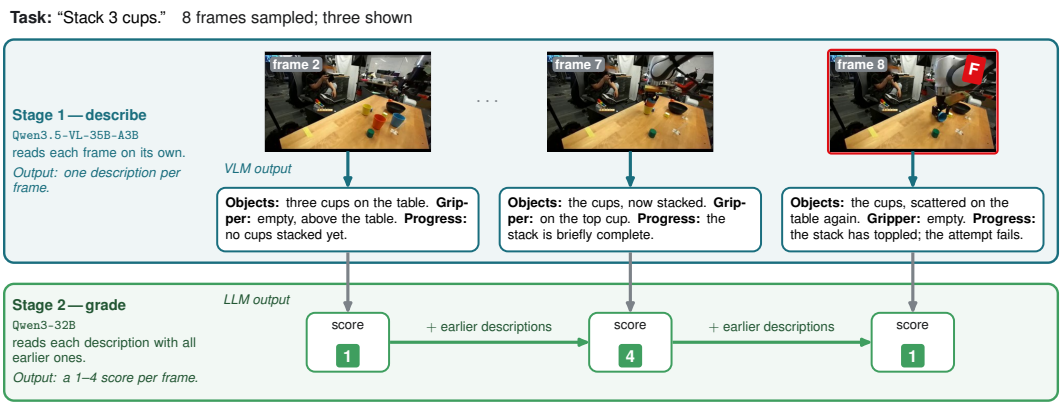


Figure 2: **Our two-stage labeling pipeline for real-world failures**, shown on three of eight sampled frames from a failed DROID attempt to stack three cups. A VLM describes each frame in isolation. An LLM grades each frame on a 1–4 progress rubric, reading its description together with all earlier ones. The score reaches 4 once the cups are stacked and falls back to 1 when the stack topples.

3.3 An asymmetric objective

Robometer's progress and success losses penalize over- and under-prediction equally, i.e., errors where the predicted reward lies above or below its target. Under RL, the two errors are not equally harmful. A reward over-prediction gives the policy more reward than it has actually accomplished, enabling reward hacking. Reward under-prediction only slows learning. We therefore keep both losses but scale the loss on under-predicted frames by a factor $\lambda < 1$. On the progress head, a frame is over-predicted when the predicted mean $\bar{p}_t = \sum_i \hat{p}_{t,i} c_i$ over the bin centers c_i exceeds the target p_t, and we weight its C51 cross-entropy [36] by w_t,

$$\mathcal{L}_{\text{prog}} = \frac{1}{T} \sum_{t=1}^{T} w_t \, \text{CE}\big(\text{Proj}(p_t),\, \hat{p}_t\big), \qquad w_t = \begin{cases} 1, & \bar{p}_t > p_t, \\ \lambda, & \bar{p}_t \le p_t, \end{cases} \tag{1}$$

where w_t is computed without gradient. On the success head over-prediction means calling an incomplete frame successful. Frames that do not depict success, with $s_t = 0$, keep full weight, and frames that show success, with $s_t = 1$, are damped by λ,

$$\mathcal{L}_{\text{succ}} = \frac{1}{|\mathcal{N}|} \sum_{t \in \mathcal{N}} \text{BCE}\big(0, \hat{s}_t\big) \;+\; \lambda \, \frac{1}{|\mathcal{P}|} \sum_{t \in \mathcal{P}} \text{BCE}\big(1, \hat{s}_t\big), \tag{2}$$

with $\mathcal{N} = \{t : s_t = 0\}$ and $\mathcal{P} = \{t : s_t = 1\}$. Missing a true success for a frame in $\mathcal{P}$ now costs less than falsely claiming success for a frame in $\mathcal{N}$. This encourages the model to only predict success when it is confident. We use $\lambda = 0.3$ on both heads. With $\lambda = 1$, the loss is symmetric.

3.4 Model details and training

RoboRef is initialized from Robometer-4B, and we fine-tune all parameters except the frozen vision encoder. We make two further changes to the model. First, we drop the preference head. Its main role in Robometer is to let failures without progress labels contribute to training. Since every failure in our corpus now carries dense labels, these failures supervise the reward heads directly, and our objective reduces to $\mathcal{L} = \mathcal{L}_{\text{prog}} + \mathcal{L}_{\text{succ}}$. Second, we sample T=16 frames per clip instead of eight, since judging a failure requires locating the frame where the attempt goes wrong. In an ablation, this helps on data sources whose failures are hard to score (Tab. 11). App. D gives more training details.

To isolate the effect of the asymmetric loss, we train a second model, Robometer-4B + Failures, identical to RoboRef except that it uses the symmetric loss ($\lambda = 1$).

4 Experiments

We compare **RoboRef** against four state-of-the-art VLM-based reward models, **Robometer-4B** [1], from which RoboRef is fine-tuned, **RoboReward-4B** [13], **Robo-Dopamine 2.0** [14], and **LRM-**

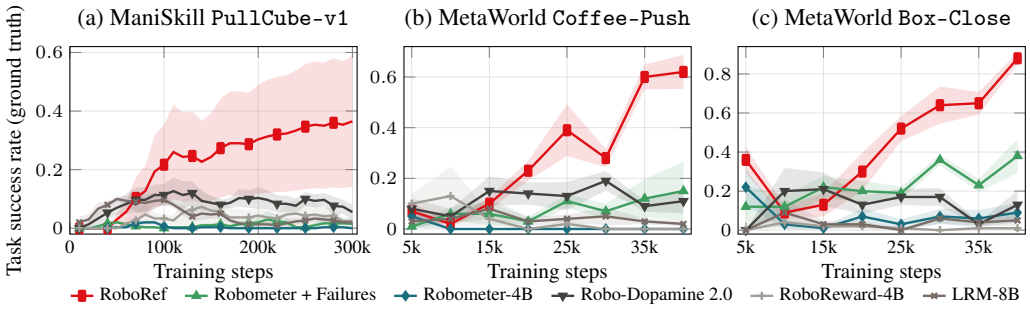


Figure 3: **Downstream RL on held-out simulated tasks.** Mean ground-truth success over five seeds, with the shaded area showing standard error. *(a)* ManiSkill `PullCube-v1`, SAC with dense rewards. *(b, c)* MetaWorld `Coffee-Push` and `Box-Close`, IBRL with sparse success rewards.

8B [15]. As a control, **Robometer-4B + Failures** (Sec. 3.4) is trained on the same data as RoboRef but with the symmetric loss, which separates our two contributions. Every reward model is frozen and queried zero-shot. We evaluate these models to answer four questions.

1. **Does RoboRef train better RL policies in simulation than existing models?** Yes, it trains the strongest policy on every task, including ones where no baseline learns at all (Sec. 4.1).
2. **What makes a VLM reward model succeed or fail under RL optimization?** False-positive rewards cause failures in RL, which RoboRef substantially reduces (Sec. 4.2).
3. **Do offline metrics predict RL success?** No, in our experiments the offline benchmark leaders train the weakest policy, and RoboRef, not built for offline metrics, trains the strongest (Sec. 4.3).
4. **Does RoboRef enable RL on a real robot?** Yes, it outperforms Robometer on two real-world manipulation tasks (Sec. 4.4).

4.1 Downstream RL in simulated tasks

Since simulated RL is reproducible across laboratories, we compare all reward models here first. We test both reward forms our models provide: a *dense per-step reward* from the progress head and a *sparse binary reward* from the success head. We evaluate on ManiSkill `PullCube-v1` and MetaWorld `Coffee-Push` and `Box-Close`. All three tasks are held out from RBM-1M+, although both simulators appear in it. We report ground-truth task success on separate evaluation episodes. Baselines run in the setups their authors intend, with their richest inputs and native reward forms. In particular, Robo-Dopamine 2.0 [14] receives a goal image and three camera views, while all other models read a single third-person view. App. E.3 gives the full protocol.

Dense reward. We train SAC [37] from scratch on ManiSkill `PullCube-v1`, with the progress head as a dense per-step reward. RoboRef clearly leads with a mean success of 0.37 (Fig. 3a). Apart from Robo-Dopamine 2.0 at 0.06, all other models, including Robometer-4B + Failures, end at or below 0.03. The wide bands reflect instability across seeds.

Sparse reward. In MetaWorld we use image-based IBRL [6], an RL algorithm for sparse rewards that initializes policy learning from demonstrations. On `Coffee-Push` and `Box-Close`, no baseline exceeds 0.13. Failure data alone raises Robometer-4B from 0.00 and 0.09 to 0.15 and 0.38. With the asymmetric loss on top, RoboRef reaches 0.62 and 0.88.

Across algorithms and reward types, only RoboRef trains a competent policy on these tasks. Robometer-4B + Failures, trained on the same data, improves on Robometer-4B but trails RoboRef, so our dataset alone does not explain the gap. We investigate what makes the difference next.

4.2 What makes a reward model work for RL?

To understand why most VLM reward models fail to train a policy, we examine the reward each model assigns to the states its own policy visits in ManiSkill. We log every reward prediction of the training runs in Fig. 3a together with the ground-truth success, about 1.5M steps per model.

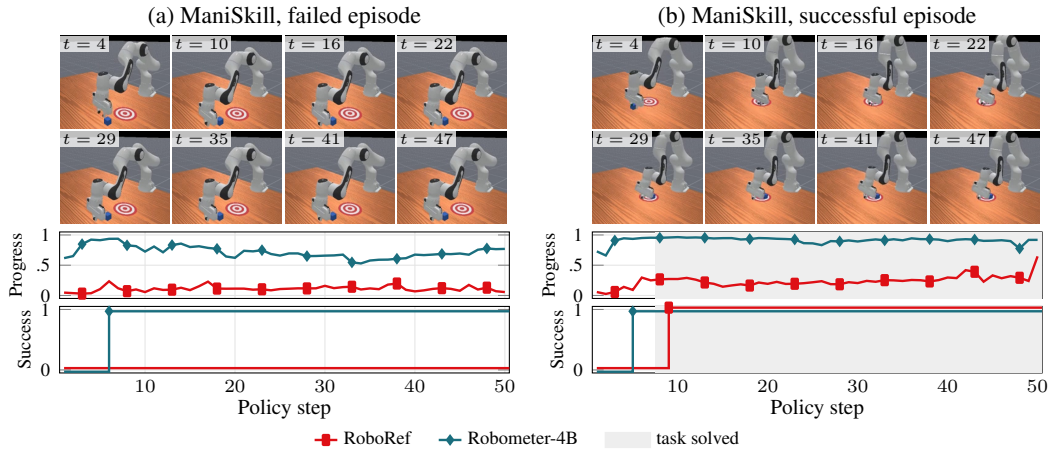


Figure 4: **Example reward predictions** of RoboRef and Robometer-4B on a failed (a) and a successful (b) ManiSkill `PullCube-v1` episode. Top: predicted progress $\in [0, 1]$. Bottom: success, held at 1 after the first detection. Gray: steps the simulator marks as solved.

RoboRef rarely rewards failure. Fig. 4 compares the reward predictions of RoboRef and Robometer-4B on the same two episodes. In the failed episode (Fig. 4a), RoboRef keeps progress near zero and never detects success. In the successful episode (Fig. 4b), its progress stays conservative but rises, and it detects success only once the task is solved. Robometer-4B's predictions look nearly identical in both episodes, with progress near 0.9 throughout and success detected within a few steps. Across all failed episodes, RoboRef predicts a mean reward of $\bar{r}_{\text{fail}} = 0.10$. In contrast, most baselines predict a high reward (Tab. 2) almost from the start of an episode (Fig. 12), which a policy can collect without ever solving the task. A low reward on failures is necessary (App. G.3), but not sufficient: RoboReward-4B assigns only 0.25 to failures, yet barely separates success from failure (AUROC 0.56, Tab. 2). RoboRef ranks successful steps above unsuccessful ones more reliably than any baseline (AUROC 0.92). App. G.2 gives further analyses.

Table 2: **On-policy rewards** on ManiSkill `PullCube-v1`, 1.5M steps per model. $\bar{r}_{\text{fail}}$: mean reward in failed episodes. AUROC: successful vs. unsuccessful steps.

Model	$\bar{r}_{\text{fail}} \downarrow$	AUROC↑
Robometer-4B	0.72	0.43
Robo-Dopamine 2.0	0.59	0.78
RoboReward-4B	0.25	0.56
LRM-8B	0.63	0.64
Constant	–	0.50
Robometer + Failures	0.24	0.66
RoboRef (ours)	**0.10**	**0.92**

Failure data and the asymmetric objective both contribute. Training Robometer-4B on RBM-1M+ cuts the reward on failed episodes from 0.72 to 0.24 and lifts AUROC from below chance (0.43) to 0.66. With identical data, the asymmetric objective lowers the failure reward further to 0.10 and raises AUROC to 0.92. Both ingredients lower the reward on failures and decrease the false positives a policy can exploit.

4.3 How does RoboRef perform on offline benchmarks?

Foundation reward models for robotics are usually validated the way perception models are, by prediction error on fixed datasets [13, 1]. In such benchmarks, models score robot trajectories, many of them successes or modified successes, and over- and under-prediction cost the same. Online RL is different. The reward is queried on the states the policy visits, which early in training are mostly failures. The policy seeks out states with high reward, so overestimated states are revisited and over-prediction is amplified, while under-prediction is not (Sec. 4.2). We therefore ask whether the two evaluations agree.

RoboRewardBench. RoboRewardBench [13] is an offline benchmark for foundation reward models in robotics. It labels more than 2k real-robot episodes on a five-level scale and measures prediction quality by Mean Absolute Error (MAE). RoboRef performs on par with most baselines but trails Robometer-4B and RoboReward-4B (Tab. 3). RoboReward-4B leads on every accuracy metric. Yet it trains no competent policy on any task we evaluated. RoboRef's mistakes are mostly under-predictions: it rates 1.3% of the episodes that fall short of the goal as near-complete or

Table 3: **Offline accuracy vs. RL success.** The two most accurate models on RoboRewardBench (RRB) train the two weakest policies. MAE rescaled to $[0, 1]$. *Can*: Robomimic `PickPlace-Can` (demonstrations as offline data). $\bar{r}_{\text{fail}}$: mean raw reward on RRB failures. RL: mean final success rate.

	Offline accuracy			Failures	RL success	
	RRB		*Can*	*RRB*	*Can*	*4 tasks*
Model	**MAE↓**	**Rank↑**	**MAE↓**	$\bar{r}_{\text{fail}}$↓	**Final↑**	**Mean↑**
Robometer-4B	0.97	0.80	0.08	0.50	0.00	0.02
Robo-Dopamine 2.0	1.34	0.60	0.11	0.22	0.33	0.16
RoboReward-4B	**0.41**	**0.86**	0.21	0.34	0.00	0.01
LRM-8B	1.06	0.70	0.14	0.29	0.20	0.07
Constant	1.24	0.50	–	–	–	–
Robometer + Failures	1.06	0.72	**0.06**	0.37	0.00	0.14
RoboRef (ours)	1.07	0.72	0.17	**0.18**	**0.64**	**0.63**

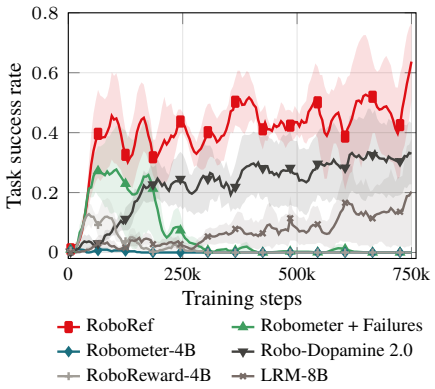


Figure 5: **RL on `PickPlace-Can`.** Mean ground-truth success of IBRL over three seeds, with standard error.

complete (score ≥ 4), where the other models rate 7–24%. RoboRef also predicts the lowest reward on non-successful episodes ($\bar{r}_{\text{fail}} = 0.18$). However, it misses 19% of true successes, twice the rate of Robometer-4B. App. G.1 gives a detailed breakdown.

Robomimic. Robomimic `PickPlace-Can` appears in no model's training data. We use the demonstrations from IBRL [6] as offline data and score each model's progress predictions against the frame order. RoboRef stands out neither by MAE (Tab. 3) nor by any other metric we compute (App. G.1). In RL with IBRL, however, it leads by a wide margin, reaching a final success rate of 0.64 against at most 0.33 for any baseline (Fig. 5). Robometer-4B, among the most accurate models offline, stays at zero task success. Comparing RoboRef with its control explains the gap. Robometer + Failures fits the labels best (0.06 MAE), but its policy first improves and then collapses. RoboRef, trained on the same data with the asymmetric loss, fits them worse (0.17 MAE) but trains the strongest policy.

Offline accuracy and RL. Lee et al. [13] found that offline accuracy predicts RL success for checkpoints of one model. Across six zero-shot models we find the opposite. Averaged across all four simulated tasks, RoboRef reaches a final success rate of 0.63, while RoboReward-4B, the most accurate model on RoboRewardBench, reaches 0.01. On RoboRewardBench, **offline accuracy and RL success rank the six models in nearly reverse order** (Tab. 3; Kendall $\tau = -0.87$, $p = 0.017$). No accuracy metric we compute shows a clear positive correlation (App. G.1). The offline quantity that tracks RL success most closely is the reward a model gives to non-successful episodes ($\bar{r}_{\text{fail}}$ in Tab. 3). The less of it, the better the policy ($\tau = 0.60$). This is the over-prediction that RoboRef is designed to suppress. Here, selecting by benchmark accuracy would pick the weakest reward for RL.

4.4 Real-world RL

We test whether RoboRef enables real-world RL on a Franka FR3 without a hand-designed reward. We train policies with RLPD [38], off-policy SAC bootstrapped from a few teleoperated demonstrations, using the RLInf implementation [39]. The reward at each step sums the model's progress and success predictions (Eq. 3). We compare against Robometer-4B, from which RoboRef is initialized. The two runs of each task differ only in which model produces the rewards, including for relabeling the demonstrations. App. F.1 describes the full setup.

Tasks. In `Pick-Zucchini`, the robot must pick up a zucchini lying on the table in front of it. In `Close-Pot-Lid`, it starts with a lid in its gripper and must place it on a pot. To keep training within a feasible wall-clock budget, the policy only moves the arm. Closing the gripper and lifting the zucchini, or opening it to release the lid, is scripted and happens upon episode termination, detected by the success head or a time-out (details in App. F.1). Despite the reduced action space, `Close-Pot-Lid` is challenging, as a lid released slightly misaligned or from too high falls off the pot. Ground-truth success is measured independently of the reward model (App. F.1).

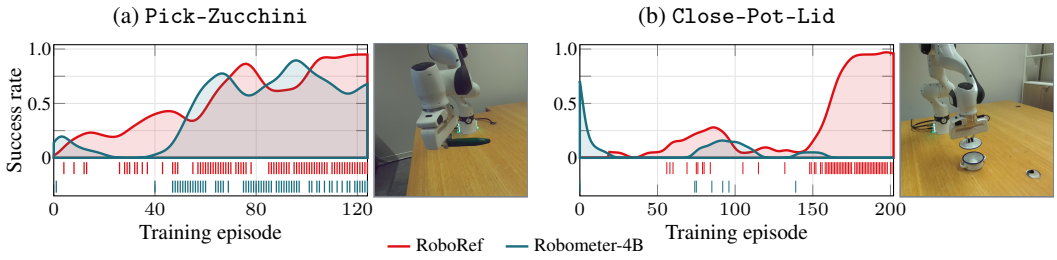


Figure 6: **Real-world RL on a Franka FR3.** Gaussian-smoothed ($\sigma = 3$) ground-truth training success rate with a 20-episode trailing average. Ticks below each plot mark solved episodes. *(a)* `Pick-Zucchini`. *(b)* `Close-Pot-Lid`. Evaluation success rates are in Tab. 4.

Results. On `Pick-Zucchini`, both models train a competent policy, with RoboRef slightly ahead at the end of training (Fig. 6a). The gap widens under frozen evaluation, where RoboRef succeeds in 80% of episodes against 53% for Robometer-4B (Tab. 4). On the more challenging `Close-Pot-Lid` task, only RoboRef learns to solve the task (Fig. 6b). Both runs start with occasional successes, but RoboRef improves on them and ends training with a high success rate. Qualitatively, we observe frequent false-positive success detections by Robometer-4B, which release the lid before it is seated (Fig. 14), the same failure mode we identified in simulation (Sec. 4.2). Under frozen evaluation, RoboRef seats the lid in 90% of episodes, while Robometer-4B succeeds in none (Tab. 4).

Table 4: **Frozen-policy evaluation**, success over 30 deterministic episodes.

Model	Zucchini	Pot lid
Robometer-4B	53%	0%
RoboRef (ours)	**80%**	**90%**

5 Conclusion

We introduced RoboRef, a foundation reward model for robot RL that (1) learns from RBM-1M+, our large-scale robotics dataset with densely annotated failures, and (2) is trained with an asymmetric objective that penalizes reward over-estimation. Queried frozen and zero-shot, RoboRef trains the strongest policy on every task we consider, across three RL algorithms, dense and sparse rewards, and simulated as well as real-world manipulation. To our knowledge, this makes RoboRef the state of the art among VLM reward models for robot RL.

Failure data and the asymmetric objective are both needed for RL: the former teaches the model to recognize failure, the latter biases its errors toward under-prediction. Both, however, are partially at odds with standard offline reward benchmarks, which contain many successful episodes and score predictions by symmetric accuracy. In our experiments, the models that fit offline labels best train the weakest policies. A reward model should therefore be judged by the policies it trains, not only by the accuracy of its predictions. We will release RoboRef, RBM-1M+, and our code to support this.

Limitations and future work. RoboRef's conservatism has a cost: it misses more true successes than its baselines, which can slow learning, although we did not observe this. The real-world failure labels in RBM-1M+ are imperfect and inherit noise from the VLM–LLM labeling pipeline (Fig. 8). RoboRef reads a single camera view and, like other VLM reward models, is sensitive to occlusion. In ManiSkill, for example, its predicted progress dips when the gripper occludes the cube (Fig. 4b, steps 12–20), which may contribute to the variance across seeds. Multiple camera views, as used by Robo-Dopamine 2.0, could make RoboRef more robust to occlusion. Finally, a frozen reward model cannot correct its errors in a new setting. Letting it adapt online, for instance through in-context learning [40] on a few rollouts, is another promising direction.

Acknowledgments and Disclosure of Funding

We thank Emily Perry and Thijmen Nijdam (University of Amsterdam) for their valuable help in conducting the real-world experiments. We also thank ELLIS for making this thesis collaboration between the University of Amsterdam and Toyota Motor Europe possible through the ELLIS MSc Honours Programme. This work used the Dutch national e-infrastructure with the support of the SURF Cooperative using grant no. EINF-17535 ("Foundation Reward Models for Robot Learning").

References

[1] Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S. Huang, Luke Zettlemoyer, Dieter Fox, Yu Xiang, Anqi Li, Andreea Bobu, Abhishek Gupta, Stephen Tu, Erdem Biyik, and Jesse Zhang. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. *CoRR*, abs/2603.02115, 2026. doi: 10.48550/ARXIV.2603.02115.

[2] Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*, pages 278–287. Morgan Kaufmann, 1999.

[3] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pages 1094–1100. PMLR, 2019.

[4] Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. ManiSkill3: GPU Parallelized Robotics Simulation and Rendering for Generalizable Embodied AI. *CoRR*, abs/2410.00425, 2024. doi: 10.48550/ARXIV.2410.00425.

[5] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[6] Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation Bootstrapped Reinforcement Learning. In Dana Kulic, Gentiane Venture, Kostas E. Bekris, and Enrique Coronado, editors, *Robotics: Science and Systems XX, Delft, The Netherlands, July 15-19, 2024*, 2024. doi: 10.15607/RSS.2024.XX.056.

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021.

[8] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-VL: A Frontier Large Vision-Language Model with Versatile Abilities. *CoRR*, abs/2308.12966, 2023. doi: 10.48550/ARXIV.2308.12966.

[9] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[10] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-E: An Embodied Multimodal Language Model. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 8469–8488. PMLR, 2023.

[11] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-Level Reward Design via Coding Large Language Models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[12] Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Yuheng Ji, Cheng Chi, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, Shaoxuan Xie, Guocai Yao, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Robo-Dopamine: General Process Reward Modeling for High-Precision Robotic Manipulation. *CoRR*, abs/2512.23703, 2025. doi: 10.48550/ARXIV.2512.23703.

[13] Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. RoboReward: General-Purpose Vision-Language Reward Models for Robotics. *CoRR*, abs/2601.00675, 2026. doi: 10.48550/ARXIV.2601.00675.

[14] Yijie Xu, Haopeng Jin, Run Zhou, Shengbang Liu, Sixiang Chen, Hongyang Cheng, Sicheng Hu, Peterson Co, Jinwen Luo, Huajie Tan, and Shanghang Zhang. Robo-Dopamine 2.0: History-Conditioned and OOD-Aware Process Reward Modeling for Robotic Manipulation. *CoRR*, abs/2608.15680, 2026. doi: 10.48550/ARXIV.2608.15680.

[15] 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. *CoRR*, abs/2603.16065, 2026. doi: 10.48550/ARXIV.2603.16065.

[16] Christian Gumbsch, Leonardo Barcellona, Lennard Schünemann, Platon Karageorgis, Andrii Zadaianchuk, Zehao Wang, Sergey Zakharov, Fabien Despinoy, Rahaf Aljundi, and Efstratios Gavves. From Demonstrations to Rewards: Test-Time Prompt Optimization for VLM Reward Models, 2026.

[17] Andrew Y. Ng and Stuart Russell. Algorithms for Inverse Reinforcement Learning. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 663–670. Morgan Kaufmann, 2000.

[18] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In Carla E. Brodley, editor, *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004. doi: 10.1145/1015330.1015430.

[19] 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.

[20] Alejandro Escontrela, Ademi Adeniji, Wilson Yan, Ajay Jain, Xue Bin Peng, Ken Goldberg, Youngwoon Lee, Danijar Hafner, and Pieter Abbeel. Video prediction models as rewards for reinforcement learning. *Advances in Neural Information Processing Systems*, 36:68760–68783, 2023.

[21] Yuyang Liu, Chuan Wen, Yihang Hu, Dinesh Jayaraman, and Yang Gao. Timerewarder: Learning dense reward from passive videos via frame-wise temporal distance. *CoRR*, abs/2509.26627, 2025. doi: 10.48550/ARXIV.2509.26627. URL https://doi.org/10.48550/arXiv.2509.26627.

[22] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep Reinforcement Learning from Human Preferences. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4299–4307, 2017.

[23] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[24] Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning. In *International Conference on Learning Representations*, 2024.

[25] Sumedh Anand Sontakke, Jesse Zhang, Séb Arnold, Karl Pertsch, Erdem Biyik, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. RoboCLIP: One demonstration is enough to learn robot policies. In *Advances in Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=DVlawv2rSI.

[26] Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson. RL-VLM-F: Reinforcement Learning from Vision Language Foundation Model Feedback. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Forty-First International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pages 51484–51501. PMLR / OpenReview.net, 2024.

[27] Cansu Sancaktar, Christian Gumbsch, Andrii Zadaianchuk, Pavel Kolev, and Georg Martius. SENSEI: Semantic exploration guided by foundation models to learn versatile world models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 52745–52777. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/sancaktar25a.html.

[28] Weifeng Lu, Minghao Ye, Zewei Ye, Ruihan Tao, Shuo Yang, and Bo Zhao. RoboFAC: A Comprehensive Framework for Robotic Failure Analysis and Correction. *CoRR*, abs/2505.12224, 2025. doi: 10.48550/ARXIV.2505.12224.

[29] 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. *CoRR*, abs/2510.01642, 2025. doi: 10.48550/ARXIV.2510.01642.

[30] Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, Vedant Choudhary, Foster Collins, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Maitrayee Dhaka, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachlan Groom, Haroun Habeeb, Hunter Hancock, Karol Hausman, Gashon Hussein, Victor Hwang, Brian Ichter, Connor Jacobsen, Szymon Jakubczak, Rowan Jen, Tim Jones, Gregg Kammerer, Ben Katz, Liyiming Ke, Mairbek Khadikov, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Brendon LeCount, Sergey Levine, Xinyu Li, Adrian Li-Bell, Vladislav Lialin, Zhonglin Liang, Wallace Lim, Yao Lu, Enyu Luo, Vishnu Mano, Nandan Marwaha, Aikys Mongush, Liam Murphy, Suraj Nair, Tyler Patterson, Karl Pertsch, Allen Z. Ren, Gavin Schelske, Charvi Sharma, Baifeng Shi, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Jiaming Tang, Jimmy Tanner, Shalom Tekeste, Marcel Torne, Kyle Vedder, Quan Vuong, Anna Walling, Haohuan Wang, Jason Wang, XuDong Wang, Chris Whalen, Samuel Whitmore, Blake Williams, Charles Xu, Sukwon Yoo, Lili Yu, Wuming Zhang, Zhuoyang Zhang, and Ury Zhilinsky. π-0.7: A Steerable Generalist Robotic Foundation Model with Emergent Capabilities. *CoRR*, abs/2604.15483, 2026. doi: 10.48550/ARXIV.2604.15483.

[31] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete Problems in AI Safety. *CoRR*, abs/1606.06565, 2016.

[32] Joar Max Viktor Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=yb3HOXO3lX2.

[33] Leo Gao, John Schulman, and Jacob Hilton. Scaling Laws for Reward Model Overoptimization. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10835–10866. PMLR, 2023.

[34] Qwen Team. Qwen3-VL Technical Report. *CoRR*, abs/2511.21631, 2025. doi: 10.48550/ARXIV.2511.21631.

[35] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R. Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Donovon Jackson, Charlotte Le, Yunshuang Li, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J. Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. DROID: A Large-Scale In-The-Wild Robot Manipulation Dataset. In Dana Kulic, Gentiane Venture, Kostas E. Bekris, and Enrique Coronado, editors, *Robotics: Science and Systems XX, Delft, The Netherlands, July 15-19, 2024*, 2024. doi: 10.15607/RSS.2024.XX.120.

[36] Marc G. Bellemare, Will Dabney, and Rémi Munos. A Distributional Perspective on Reinforcement Learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 449–458. PMLR, 2017.

[37] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1856–1865. PMLR, 2018.

[38] Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning (ICML)*, 2023.

[39] 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. In *20th USENIX Symposium on Operating Systems Design and Implementation (OSDI 26)*, pages 829–846, 2026.

[40] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[41] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alexander Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta,

Andrew E. Wang, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Paul Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guanzhi Wang, Hao Su, Haoshu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I. Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey T. Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi Jim Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J. Joshi, Niko Sünderhauf, Ning Liu, Norman Di Palo, Nur Muhammad (Mahi) Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R. Sanketi, Patrick Tree Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante-Gomez, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham D. Sonawani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Liangwei Xu, Xuanlin Li, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, and Zipeng Lin. Open X-Embodiment: Robotic Learning Datasets and RT-X Models : Open X-Embodiment Collaboration. In *IEEE International Conference on Robotics and Automation, ICRA 2024, Yokohama, Japan, May 13-17, 2024*, pages 6892–6903. IEEE, 2024. doi: 10.1109/ICRA57147.2024.10611477.

[42] Haoshu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. RH20T: A Comprehensive Robotic Dataset for Learning Diverse Skills in One-Shot. In *IEEE International Conference on Robotics and Automation, ICRA 2024, Yokohama, Japan, May 13-17, 2024*, pages 653–660. IEEE, 2024. doi: 10.1109/ICRA57147.2024.10611615.

[43] Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, and Ranjay Krishna. MolmoAct: Action Reasoning Models that can Reason in Space. *CoRR*, abs/2508.07917, 2025. doi: 10.48550/ARXIV.2508.07917.

[44] Yinpei Dai, Jayjun Lee, Nima Fazeli, and Joyce Chai. RACER: Rich Language-Guided Failure Recovery Policies for Imitation Learning. In *IEEE International Conference on Robotics and Automation, ICRA 2025, Atlanta, GA, USA, May 19-23, 2025*, pages 15657–15664. IEEE, 2025. doi: 10.1109/ICRA55743.2025.11127799.

[45] Pranav Atreya, Karl Pertsch, Tony Lee, Moo Jin Kim, Arhan Jain, Artur Kuramshin, Clemens Eppner, Cyrus Neary, Edward S. Hu, Fabio Ramos, Jonathan Tremblay, Kanav Arora, Kirsty Ellis, Luca Macesanu, Matthew Leonard, Meedeum Cho, Ozgur Aslan, Shivin Dass, Jie Wang, Xingfang Yuan, Xuning Yang, Abhishek Gupta, Dinesh Jayaraman, Glen Berseth, Kostas Daniilidis, Roberto Martin Martin, Youngwoon Lee, Percy Liang, Chelsea Finn, and Sergey Levine. RoboArena: Distributed Real-World Evaluation of Generalist Robot Policies. *CoRR*, abs/2506.18123, 2025. doi: 10.48550/ARXIV.2506.18123.

[46] Zhiyuan Zhou, Pranav Atreya, Abraham Lee, Homer Rich Walke, Oier Mees, and Sergey Levine. Autonomous Improvement of Instruction Following Skills via Foundation Models. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, volume 270 of *Proceedings of Machine Learning Research*, pages 4805–4825. PMLR, 2024.

[47] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[48] Zhiyuan Zhou, Pranav Atreya, You Liang Tan, Karl Pertsch, and Sergey Levine. AutoEval: Autonomous Evaluation of Generalist Robot Manipulation Policies in the Real World. *CoRR*, abs/2503.24278, 2025. doi: 10.48550/ARXIV.2503.24278.

[49] Qwen Team. Qwen3.5-Omni Technical Report. *CoRR*, abs/2604.15804, 2026. doi: 10.48550/ARXIV.2604.15804.

[50] NVIDIA. World Simulation with Video Foundation Models for Physical AI. *CoRR*, abs/2511.00062, 2025. doi: 10.48550/ARXIV.2511.00062.

[51] GLM. GLM-5: From Vibe Coding to Agentic Engineering. *CoRR*, abs/2602.15763, 2026. doi: 10.48550/ARXIV.2602.15763.

[52] Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, Guanzhou Chen, Zichen Ding, Changyao Tian, Zhenyu Wu, JingJing Xie, Zehao Li, Bowen Yang, Yuchen Duan, Xuehui Wang, Zhi Hou, Haoran Hao, Tianyi Zhang, Songze Li, Xiangyu Zhao, Haodong Duan, Nianchen Deng, Bin Fu, Yinan He, Yi Wang, Conghui He, Botian Shi, Junjun He, Yingtong Xiong, Han Lv, Lijun Wu, Wenqi Shao, Kaipeng Zhang, Huipeng Deng, Biqing Qi, Jiaye Ge, Qipeng Guo, Wenwei Zhang, Songyang Zhang, Maosong Cao, Junyao Lin, Kexian Tang, Jianfei Gao, Haian Huang, Yuzhe Gu, Chengqi Lyu, Huanze Tang, Rui Wang, Haijun Lv, Wanli Ouyang, Limin Wang, Min Dou, Xizhou Zhu, Tong Lu, Dahua Lin, Jifeng Dai, Weijie Su, Bowen Zhou, Kai Chen, Yu Qiao, Wenhai Wang, and Gen Luo. InternVL3.5: Advancing Open-Source Multimodal Models in Versatility, Reasoning, and Efficiency. *CoRR*, abs/2508.18265, 2025. doi: 10.48550/ARXIV.2508.18265.

[53] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[54] Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. GELLO: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

A RBM-1M+

This appendix expands Section 3.2. Table 5 gives the composition of the training corpus at the level the main text needs, Table 6 breaks the same corpus down to one row per upstream source, and Table 7 says where every dense per-frame failure label comes from. Every count describes the corpus of 648,714 rows, where one row is one episode rendered from a single camera view. Every successful trajectory, whatever its source, uses Robometer's t/T heuristic, so all 531,928 successes share one labeling route and the 116,786 failures are where the work is.

Table 5: Composition of the training corpus. One row is one clip from one camera view. *Labels*: source of the dense per-frame failure targets. All successes use the t/T heuristic.

Source	Labels	Failures	Successes
Robometer expert demos	–	0	490,798
Robometer mixed archives	VLM+LLM	68,923	26,029
MetaWorld (ours)	simulator	24,402	1,850
FailSafe (ours)	simulator	2,098	75
DROID (ours)	VLM+LLM	5,503	5,863
RoboReward partials (ours)	ramp to score	15,860	7,313
Total		**116,786**	**531,928**

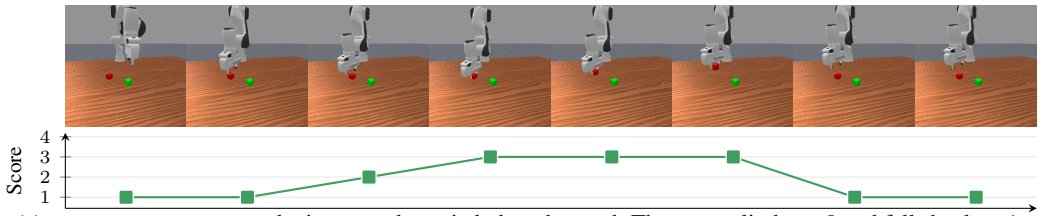
Table 6: Per-source composition of the training corpus. Counts are clips, one per camera view. Small sources are grouped into "other" rows. Totals match Table 5.

Source	Failures	Successes	Total
Expert demonstrations retained from RBM-1M			
DROID [35]	0	149,804	149,804
RT-1 (Fractal) [41]	0	87,204	87,204
Bridge V2 [41]	0	83,023	83,023
Language Table [41]	0	50,000	50,000
BC-Z [41]	0	39,344	39,344
RoboSet [41]	0	36,480	36,480
RH20T [42]	0	15,744	15,744
MolmoAct [43]	0	15,324	15,324
Other Open X-Embodiment [41]	0	13,875	13,875
Mixed-expertise archives retained from RBM-1M			
RACER [44]	29,211	7,131	36,342
RoboArena [45]	17,510	2,635	20,145
SOAR [46]	11,999	4,797	16,796
LIBERO [47]	6,241	5,659	11,900
AutoEval [48]	3,721	4,946	8,667
Other retained archives	241	861	1,102
Added or replaced by us			
MetaWorld [3] (added, sim.)	24,402	1,850	26,252
RoboReward partials [13] (replaced)	15,860	7,313	23,173
DROID failures (added)	5,503	5,863	11,366
FailSafe [29] (replaced, sim.)	2,098	75	2,173
Total	**116,786**	**531,928**	**648,714**

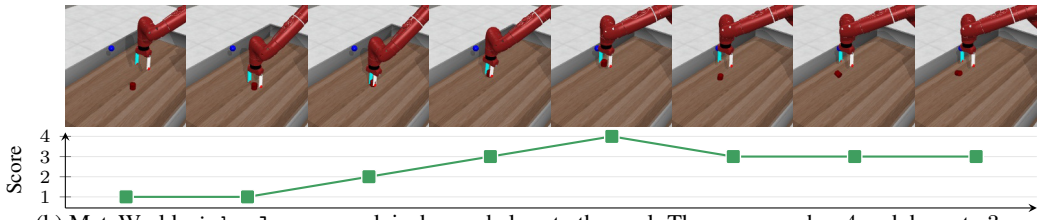
The retained expert demonstrations are the largest part and come almost entirely from Open X-Embodiment sources [41] carried over from RBM-1M, with DROID, RT-1 and Bridge the three biggest. The retained mixed-expertise archives are the smaller corpora that actually carry failures, chief among them RACER, RoboArena and SOAR. Below them sit the sources we add or replace ourselves.

Table 7: Origin of the dense per-frame failure labels. All 531,928 successes use the t/T heuristic and are omitted.

Label route	Sources	Failures
Simulator ground truth	MetaWorld, FailSafe	26,500
VLM+LLM pipeline	DROID, RACER, RoboArena, SOAR, LIBERO, AutoEval, other retained archives	74,426
Ramp to clip score	RoboReward partials	15,860
Total		**116,786**



(a) `FailSafe-stack`: a cube is grasped, carried, then dropped. The score climbs to 3 and falls back to 1.



(b) MetaWorld `pick-place`: a puck is dropped close to the goal. The score reaches 4 and drops to 3.

Figure 7: Dense, non-monotonic failure labels from simulation. Each strip shows eight keyframes of a failed attempt, and the curve beneath plots the per-frame progress score read from the simulator. In both cases the score rises as the arm makes progress and then regresses when the object is dropped.

B Simulated Failures

This appendix describes the failures we generate in simulation for RBM-1M+ (Section 3.2). Figure 7 shows two of them in full, eight keyframes of a failed attempt above the per-frame progress score read from the simulator. In both, the score climbs while the arm makes progress and falls back when the object is dropped, the non-monotonic shape that a t/T label cannot express. Appendix B.1 describes how these failures are generated, and Appendix B.2 shows a larger sample.

B.1 Tasks and failure modes

Both simulated benchmarks generate failures by injecting a fault into a scripted policy, as described in Section 3.2. The kind of failure that comes out depends on the task family and the target score. Table 8 lists the nine noise modes we use in MetaWorld [3] and where each applies, and Table 9 lists the tasks by family.

FailSafe [29] adds three ManiSkill [4] tasks, `FailPickCube` (lift a cube to a goal), `FailPushCube` (push a cube to a goal), and `FailStackCube` (stack a red cube on a green one), scored on the same one-to-five scale. Each task is driven by a scripted multi-stage planner, reach, grasp, lift, align, place, and a fault is injected at a chosen stage. The stage sets the score, so a fault early in the plan yields a low score and one at the last stage a near-complete score-4 failure. The injected fault fits the stage, a grasp that is then released or frozen, a push that stalls or veers off, or a cube carried and dropped just short of the target. Stacking is the only FailSafe task with a meaningful score 3, a cube grasped and held or carried but not yet placed.

Table 8: The nine MetaWorld noise modes and the family and score at which each is applied. Scores 1 and 2 use the same modes across every family; the higher scores are family-specific.

Mode	Behavior	Applied at
random	fully random actions	score 1 (all families)
wrong-direction	a strong bias away from the goal	score 1 (all); score 3 (grasp-and-move)
freeze	the arm holds its position	scores 2–4 (all families)
drop	the gripper is forced open mid-transport	score 3 (grasp-and-move)
late-drop	the gripper is forced open near the goal	score 4 (grasp-and-move)
misplace	the expert action with a small offset near the goal	score 4 (grasp-and-move, push-and-sweep)
lift-away	the arm lifts off and loses contact	score 3 (push-and-sweep)
push-sideways	the arm pushes perpendicular to the goal	scores 3–4 (push-and-sweep)
partial-actuation	the expert drives the mechanism most of the way, then stalls	score 4 (mechanism, press)

Table 9: The 46 MetaWorld tasks by family, with the scores each family covers. Mechanism and press tasks skip score 3, whose single actuation has no meaningful partial state.

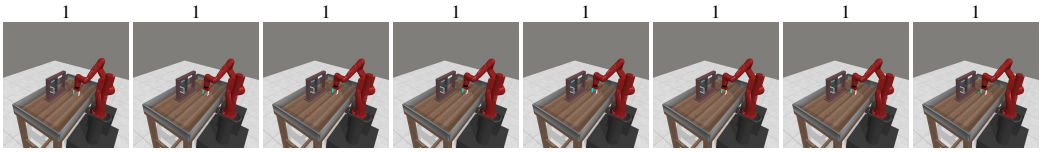
Family	Scores	Tasks
Grasp-and-move	1–4	assembly, basketball, bin-picking, box-close, coffee-pull, hammer, hand-insert, pick-out-of-hole, pick-place, pick-place-wall, shelf-place, stick-pull, stick-push
Push-and-sweep	1–4	coffee-push, plate-slide, plate-slide-side, plate-slide-back, plate-slide-back-side, push, push-wall, push-back, soccer, sweep, sweep-into
Mechanism	1, 2, 4	dial-turn, disassemble, door-open, door-close, door-lock, door-unlock, drawer-open, drawer-close, faucet-open, faucet-close, handle-press, handle-press-side, handle-pull, handle-pull-side, lever-pull, window-open, window-close
Press	1, 2, 4	button-press, button-press-wall, button-press-topdown, button-press-topdown-wall, coffee-button

B.2 Examples

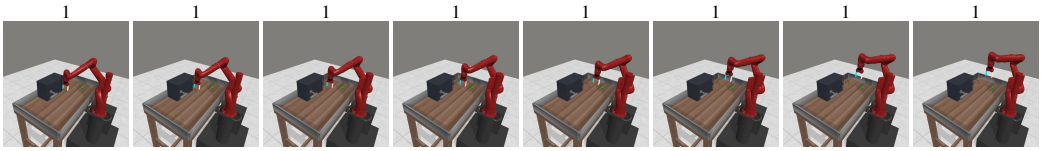
This section shows a sample of the failure trajectories we generate in simulation. Each example is a short filmstrip of a single failed attempt, and the number above each frame is its ground-truth progress score, read directly from the simulator state. The scores run from 1 (no progress) to 5 (success), so a failed attempt never exceeds 4.

We group the examples by the score the attempt reaches, so each block illustrates what a given level of partial progress looks like across different tasks and different ways of failing. The failure modes the generator can use depend on how far the attempt gets. Early failures look like random flailing or motion in the wrong direction, mid-progress failures hold an object and then freeze or push it the wrong way, and near-complete failures grasp or place the object and then drop or misplace it at the last moment. The scores follow the behavior, including downward when an object is dropped, which is the property a monotone t/T label cannot capture.

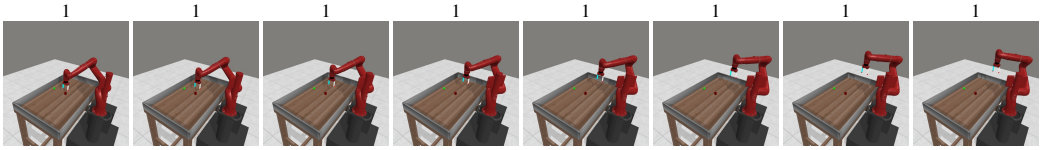
B.2.1 Score 1: no meaningful progress



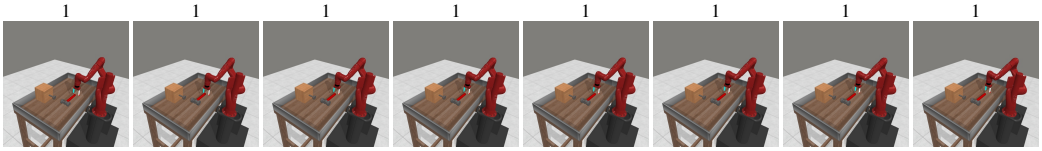
Window-close, random actions.



Door-open, wanders off, never reaches the door.

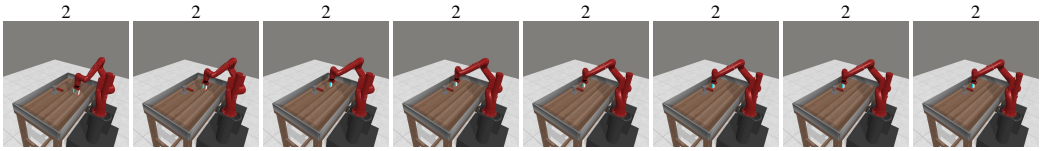


Push, moves the wrong way.

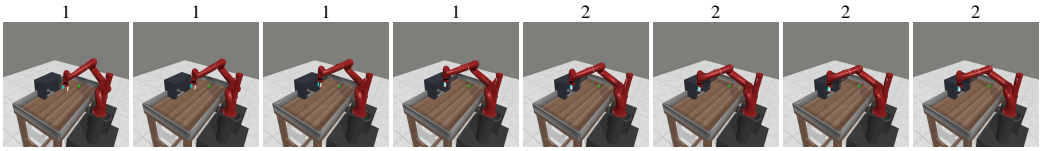


Hammer, random actions.

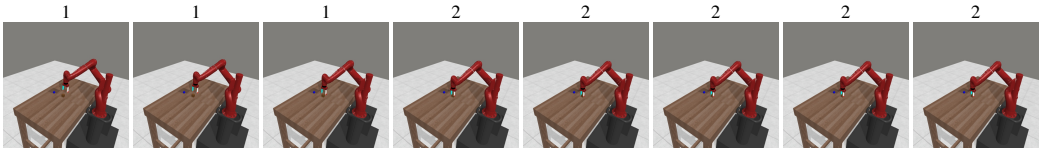
B.2.2 Score 2: object approached but not engaged



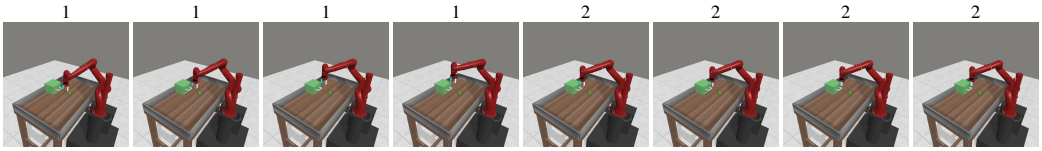
Faucet-open, reaches then freezes.



Door-open, reaches then freezes.

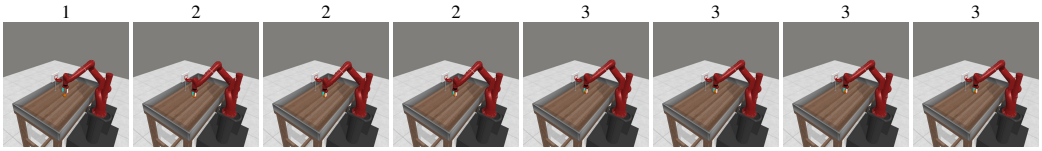


Sweep-into, touches then freezes.

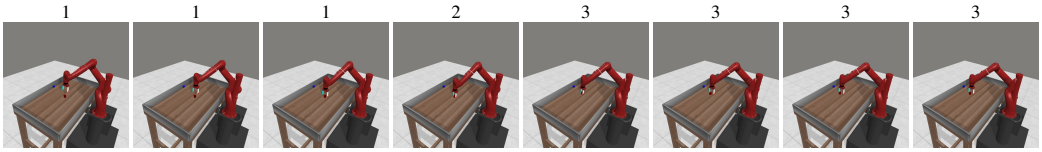


Drawer-open, reaches the handle but never grips it.

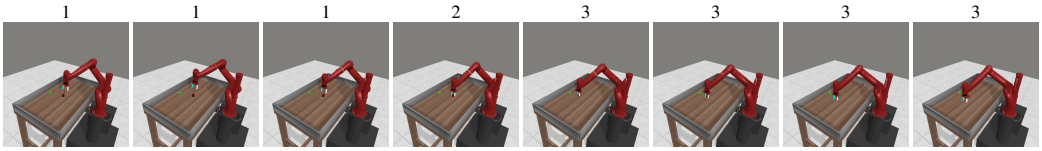
B.2.3 Score 3: object engaged, partial progress



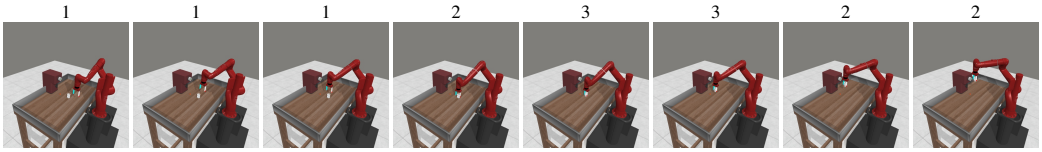
Basketball, grasps the ball but never lifts it to the hoop.



Pick-place, picks up the puck then drops it.

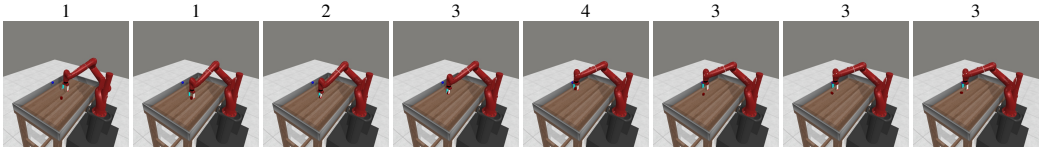


Push the puck to the goal, then slides it sideways instead.

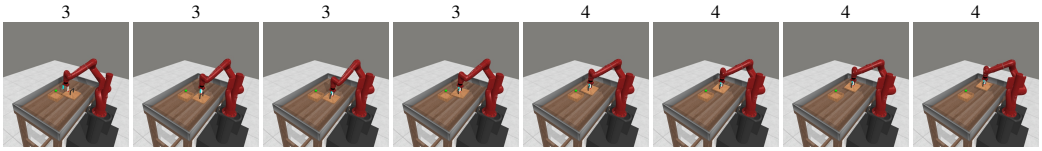


Coffee-push, pushes partway then lifts away.

B.2.4 Score 4: near-complete but ultimately fails



Pick-place, grasps the puck and carries it close to the goal but ultimately drops it.



Box-close, nearly closes then misplaces.

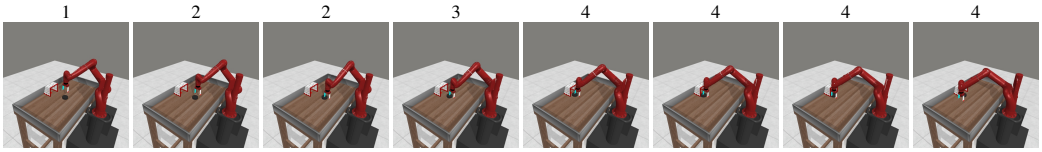
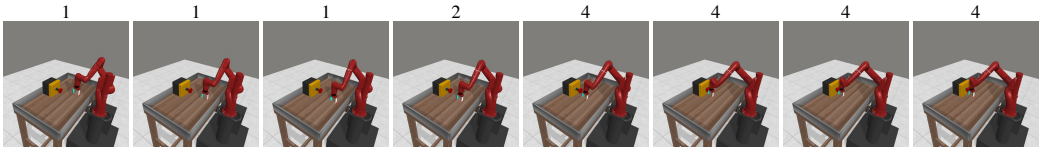
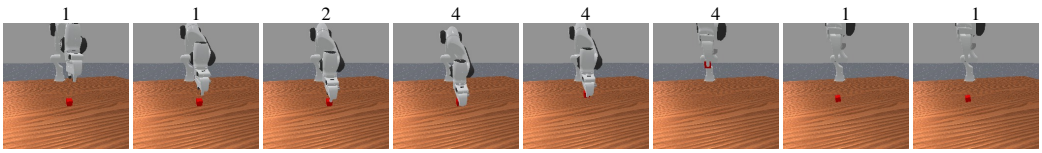


Plate-slide, almost there then off-line.

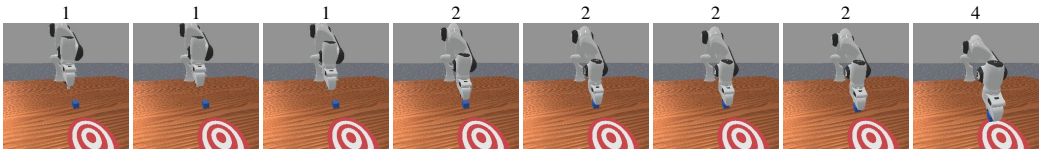


Button-press, almost presses then stops.

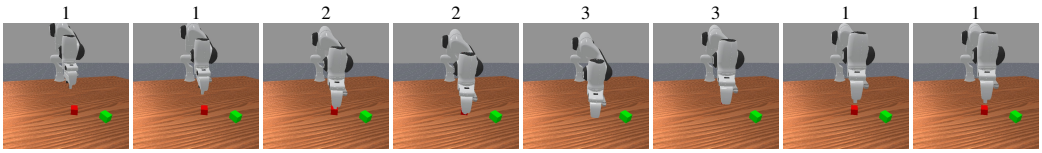
B.2.5 FailSafe (ManiSkill)



Pick, grasps and lifts then releases.



Push, pushes toward the goal then stalls.



Stack, carries the cube then drops it.

C VLM Labeling of Real-World Failures

These are the prompts behind the two-stage pipeline of Section 3.2, with the vision-language model prompt in teal and the language model prompts in amber. The vision-language model sees the frame prompt once per frame. For multi-object tasks the language model first runs the decomposition prompt to turn the instruction into ordered sub-steps, then runs the grading prompt once per frame, accumulating the earlier frame descriptions and its own previous score. Placeholders in braces are filled in at run time.

C.1 Vision-language model

Frame description • Qwen3.5-VL-35B-A3B

```
Task: {task}

Look at this image and answer each question in 1 sentence. Do not guess intentions or say
 what the robot is "about to do".

1. List the main objects relevant to the task and their state (up to 5 objects). Count
multiples. (e.g., "3 cups upright on table, 1 cup in gripper", "cloth folded in half on
table", "lid off, next to blender")
2. What is the gripper doing RIGHT NOW? (e.g., "gripping the red cup", "empty and open",
"closed around the cloth edge", "not visible")
3. Compared to the task goal, what has been accomplished and what has not? (e.g., "1 of 4
 cups stacked, 3 remain", "cloth lifted but not folded", "towel is on the rack")
```

For tasks that involve several objects, questions 1 and 3 are replaced with counting variants, while question 2 stays the same.

Frame description, multi-object variant (questions 1 and 3)

```
1. List ALL relevant objects and their current state. If there are multiple instances of
the same object, count them explicitly -- both those already processed and those
remaining. (e.g., "3 cups stacked, 2 still on table", "1 nail polish in purse, 3
remaining on table")
3. How many objects have been successfully moved/processed toward the goal vs how many
remain? Give a count if possible.
```

C.2 Language model

The language model does two jobs, decomposing a multi-step task into ordered sub-steps and then grading each frame.

Task decomposition • Qwen3-32B

```
Break the following robot manipulation task into 3-6 ordered sub-steps that a robot must
complete sequentially. Each sub-step should be a concrete, observable action.

Task: {task}

Rules:
- List exactly the physical sub-steps in order.
- Each sub-step must be independently verifiable from a camera image.
- Do NOT include "start" or "finish" as sub-steps.
- Keep each sub-step to one sentence.

Reply with a numbered list and nothing else. Example for "stack the red cup on the blue
cup":
1. Move gripper toward the red cup.
2. Grasp the red cup.
3. Lift the red cup off the table.
4. Move the red cup above the blue cup.
5. Lower the red cup onto the blue cup.
6. Release the red cup.
```

Cumulative grading • Qwen3-32B

```
You are grading a robot's cumulative progress toward completing a task. You will see
descriptions of keyframes from a failed episode.

Task: {task}

This is frame {frame_num} of {total_frames}.{prev_score_text}

{previous_context}Current frame (frame {frame_num}, {current_timestamp}s):
{current_description}

Score the robot's cumulative progress AS OF THIS FRAME from 1 to 4:
1 - No progress: The robot never interacted with the correct task-relevant object. Moving
, hovering, or grasping the WRONG object counts as 1. If the task is "stack cups" but the
 robot only touches a bottle, that is 1.
2 - Early progress: The robot approached or contacted the correct object but did not
advance beyond that (e.g., moved toward the cup but didn't grasp, or grasped it but
immediately dropped it).
3 - Partial progress: The robot completed at least one meaningful sub-step of the task (e
.g., grasped the correct object AND moved it toward the goal, but didn't finish).
4 - Near completion: The robot completed MOST sub-steps of the task but failed at the
very end (e.g., grasped + moved + nearly placed but dropped; lid was almost on but slid
off).
```

```
How to handle ambiguity:
- Frame 1 MUST be scored 1. The robot cannot have completed any meaningful action by the
very first keyframe. If the description suggests otherwise, it is a VLM hallucination --
score 1.
- Frame 2 can be scored at most 2. If the description suggests score 3 or 4, it is almost
 certainly a VLM hallucination -- score 1 or 2.
- Your score should be >= your previous score UNLESS there is clear evidence that
progress was REVERSED.
- Do NOT lower the score just because the current description is less detailed than a
previous one.
- If in doubt, keep your previous score.

All episodes are failures -- the task was NOT completed, so a score of 5 is not possible.

Reply with exactly one line: ANSWER: <score> because <one sentence reason>
```

For a multi-step task the same grading template is used, with two changes. First, the decomposed sub-steps, produced by the decomposition prompt above, are inserted just after the task line.

Grading, multi-step: sub-steps inserted after the task line

```
The task consists of these ordered sub-steps:
{substeps}
```

Second, the four-level rubric is replaced by one defined in terms of the fraction of sub-steps completed.

Grading, multi-step: rubric

```
This is a MULTI-STEP task. Score based on what fraction of the sub-steps above have been
completed:
1 - No progress: The robot never interacted with the correct task-relevant object(s).
2 - Early progress: The robot completed only the very first sub-step(s) -- e.g.,
approached and grasped the correct object but made no further advancement. Grasping alone
 is a 2, not a 3.
3 - Partial progress: The robot completed roughly ONE THIRD of the sub-steps and made
meaningful advancement beyond just grasping. For a 6-step task, completing ~2 steps earns
 a 3.
4 - Near completion: The robot completed roughly TWO THIRDS or more of the sub-steps.
Only the last step(s) remain. For a 6-step task, completing 4+ steps earns a 4.
```

The per-frame context, the ambiguity rules, and the answer format are unchanged.

C.3 Choosing the labeling models

We label real-world failures in two stages because a single vision-language model proved unreliable at the job. Our first attempt scored full DROID videos directly with RoboReward-8B [13], and its score followed how many frames it saw more than what they showed. General-purpose VLMs given a whole clip and its task fail in the opposite direction. They read the task description, assume the robot carried it out, and report progress that never happened. A clip-level scorer also returns one number per video, while our training targets need one per frame. Splitting the work narrows what each model must do. The VLM only has to say what a single frame shows, and a language model that never sees pixels turns those descriptions into scores. Even so, the pipeline still hallucinates, and Figure 8 shows a case where it does. To choose the two models behind the pipeline of Section 3.2, we built a small hand-labeled benchmark. We drew fifty DROID failure clips, picked to be as varied as we could make them, sampled eight frames from each, and annotated every frame by hand with its ground-truth rubric level. Preliminary runs pointed to the vision stage as the bottleneck, while the grading language model, Qwen3-32B, was already reliable in prior work and in our own use. We therefore fixed the grader to Qwen3-32B and varied the vision-language model that writes the frame descriptions.

We score each candidate by how closely the pipeline's per-frame labels match the manual ones, and report two numbers. The first is exact-match accuracy. The second is a within-one accuracy, which also counts a prediction correct when it lands within a single level of the label. The tolerance reflects what these labels are for. The downstream signal needs to recognize a failure and roughly how far it got, so a one-level slip, a three read as a two or a four, does not change the picture. Table 10 reports both over the four hundred hand-labeled frames.

Table 10: Candidate describers for the labeling pipeline, with Qwen3-32B as the fixed grader. Per-frame accuracy against 50 hand-labeled DROID failure clips. *Exact*: correct level. *Within one*: at most one level off. We use Qwen3.5-VL.

Describer	Exact (%)↑	Within one (%)↑
Qwen3.5-VL-35B-A3B [49]	81.6	**91.2**
Cosmos-Predict 2.5 [50]	80.2	88.0
GLM-4.7 [51]	71.1	85.0
InternVL3.5 [52]	69.3	79.8
Claude Opus 4.6 *(proprietary)*	**84.2**	89.8

Two models stood out, Qwen3.5-VL-35B-A3B [49] and Cosmos-Predict 2.5, a model specialized for robot data. GLM-4.7 and InternVL3.5 were clearly weaker on both measures. Claude Opus 4.6 posted the best exact-match figure, but not by a margin that justified paying for a closed model at the scale we label, and on the within-one measure it fell behind Qwen3.5-VL. Between the two front-runners we chose Qwen3.5-VL.

Furthermore, Qwen3.5-VL is a fast mixture-of-experts model that handles language as readily as vision and in our experiments it runs at least five times faster than Qwen3, so we tried using it for the grading stage as well and dropping Qwen3-32B. The grades came out noticeably worse, so we kept the pairing described in Section 3.2, a Qwen3.5-VL describer and a Qwen3-32B grader.

However, there are still noticeable hallucinations and we show an example in Figure 8 for transparency. The VLM invents lego blocks that are not present and keeps changing their colors, a white block in the gripper at one frame and a black one at the next. Reading these shifting descriptions, the grader infers that objects are being handled and moved and that the manipulation must be progressing, so it raises the score to near-complete even though the attempt only scatters the tower. This is the clearest evidence that the bottleneck sits on the vision side of the pipeline. The grader reasons soundly over what it is told, so once the descriptions drift its scores drift with them. We accept a small amount of this residual label noise on the assumption that it washes out over a large corpus.

D Training Details

Frame budget. RoboRef reads sixteen frames per clip where Robometer reads eight (Section 3.4). Table 11 compares the two budgets on the asymmetric fine-tune.

The two-stage pipeline scores eight frames per clip, since sixteen proved too costly, and the scores are lifted to the sixteen frames our models read by the conservative interpolation of Section 3.2. Failures are 18.0% of the corpus, so we oversample them by a factor of 1.89 during training, which lifts their share of a batch to about 30%. Two sources are weighted further. MetaWorld successes get a factor of thirteen, since that split is almost all failures and its few successes would otherwise barely appear, and DROID failures by three. FailSafe gets no extra weight, since its clips come from only three tasks.

We fine-tune with AdamW [53], weight decay 0.01, a cosine schedule with 10% warmup, gradients clipped to norm 10, eight trajectories per GPU, a learning rate of 10^{-5}, and $5{,}000$ steps. The vision encoder stays frozen and training is sharded across eight H100 GPUs with FSDP in bfloat16.

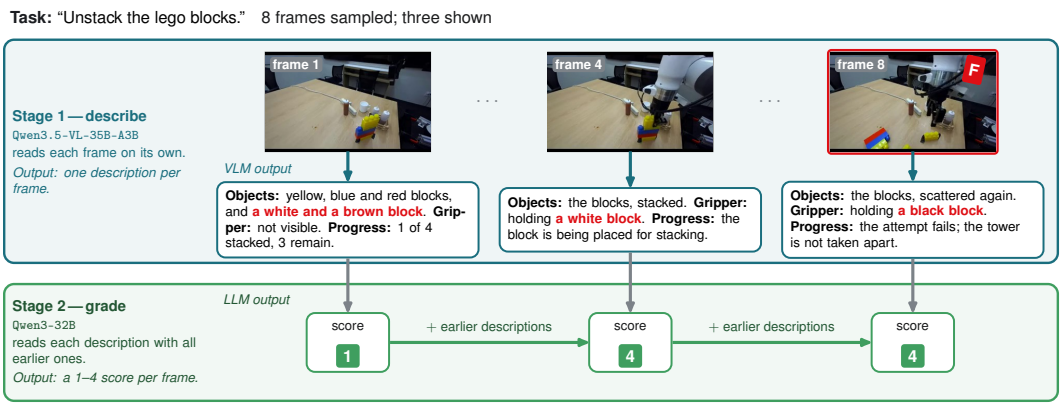


Figure 8: A hallucination that survives the two-stage pipeline of Section 3.2, shown in the format of Figure 2, on a DROID clip whose task is to unstack a lego tower. The real blocks are yellow, blue and red with a second yellow block on top. The white and brown objects behind them are a cup and a bottle. The VLM's fabrications, in red, report objects that are not there (a white and a brown block), then misread the grasped yellow block as white, and at the end see a black block in an empty gripper. Because each description is stated with confidence, and the reported blocks on the gripper keep changing, the grader is pulled to a full score of four from the fourth frame on, whereas the true progress is closer to two. The arm lifts the top block but then scatters the tower instead of unstacking it. Frames, descriptions, and scores are real outputs.

Table 11: Eight vs. sixteen frames per clip for RoboRef after 5,000 training steps, on the in-distribution evaluation splits. *Kendall's τ*: correlation with true frame order. *Rank accuracy*: fraction of success–failure pairs ordered correctly. Higher is better for both.

	Kendall's τ			Rank accuracy		
Split	**8 frames**	**16 frames**	**Δ**	**8 frames**	**16 frames**	**Δ**
MetaWorld	0.620	**0.750**	+0.129	0.791	**0.861**	+0.069
DROID	0.314	**0.331**	+0.017	0.557	**0.622**	+0.065
FailSafe	**0.768**	0.752	−0.016	**0.884**	0.876	−0.008
Robometer	0.740	**0.763**	+0.022	**0.881**	0.867	−0.014
Mean	0.611	**0.649**	+0.038	0.778	**0.806**	+0.028

E Evaluation Details

E.1 Evaluation splits

The frame-budget ablation (Tab. 11) uses four in-distribution splits that hold out clips from the training sources, MetaWorld, DROID, FailSafe, and Robometer, so each model is scored on the distribution it trained on. Table 12 gives their sizes. The clips are held-out episodes of training tasks, with one exception. Four MetaWorld tasks, `assembly`, `box-close`, `coffee-push`, and `stick-pull`, are dropped from every model's training set and appear only at evaluation, so that split also probes unseen tasks.

Table 12: Sizes of the four in-distribution offline evaluation splits.

Split	Domain	Failures	Successes	Total
MetaWorld	simulated	468	104	572
DROID	real-world	229	734	963
FailSafe	simulated	481	74	555
Robometer	mixed	1,387	2,478	3,865
Total		**2,565**	**3,390**	**5,955**

Two details qualify these splits. The clips are held-out *episodes* of training tasks, with one exception. Four MetaWorld tasks, `assembly`, `box-close`, `coffee-push`, and `stick-pull`, are dropped from every model's training set and appear only at evaluation, so that split also probes unseen tasks. The out-of-distribution split is Robometer's held-out-robot set, 571 real-world clips across six robot arms no model saw in training, labeled successful (233), suboptimal (172), or failure (166).

E.2 RoboRewardBench protocol

RoboRewardBench [13] pairs a rollout video with its task description and asks for a single end-of-episode score in $\{1, \ldots, 5\}$, where 1 marks no goal-relevant change and 5 perfect completion. It holds 2,831 human-verified episodes drawn from Open X-Embodiment [41] and RoboArena [45]. We run every model under a single protocol, so all numbers in Table 14 come from our own experiments. Each video is sampled at one frame per second, always including the final frame.

Querying each model. RoboReward-4B returns a score on the benchmark's five levels directly. RoboRef, Robometer-4B + Failures, and Robometer-4B are read at the final frame of their progress head, which returns a continuous value on $[0, 1]$ that we cut into five equal levels. Robo-Dopamine 2.0 and LRM-8B score the change between two states, so we give them each episode's first and final frame and read the result on the same five levels. LRM-8B receives the inputs it expects. Robo-Dopamine 2.0 compares a state against a goal image of the completed task, which the benchmark does not supply. We substitute the first frame, so its numbers here understate it.

Rescaling. Cutting a continuous output into five equal levels assumes the model uses the full range. RoboRef does not. Its conservative objective keeps its progress below 0.55 on this data, so it can never output a 4 or a 5, although a third of the episodes carry those labels. We therefore also report each metric after min–max rescaling every model's outputs on the benchmark to $[0, 1]$ before cutting, under one rule applied to every row. This leaves RoboReward-4B and LRM-8B unchanged, since both already span the full range, and it makes Robo-Dopamine 2.0 worse. Rescaled, RoboRef's MAE improves from 1.35 to 1.07, which passes the constant predictor.

Metrics. MAE is the group-wise mean absolute error against the 1–5 label, and exact match is the fraction of episodes assigned the correct level. Rank accuracy is the fraction of episode pairs with different labels that a model orders correctly. It is 0.50 at chance and unaffected by rescaling, so we treat it as the fairest comparison across models whose outputs live on different scales. The constant predictor answers 3 everywhere.

E.3 Reinforcement learning protocol

Evaluation. Every curve reports ground-truth task success, measured in separate evaluation episodes in which the reward model is never queried. Curves show the mean over seeds with a one standard error band. Each run uses a single A100.

Dense reward (ManiSkill). We train SAC [37] from scratch, with no demonstrations and no pretraining, for 300k steps over five seeds. The progress head supplies a dense per-step reward, and no model terminates the episode, so no success threshold or minimum episode length is needed. RoboReward-4B supports only sparse rewards and runs under the sparse protocol below.

Sparse reward (MetaWorld, Robomimic). We use image-based IBRL [6], which starts from a behavior-cloning policy pretrained on demonstrations and then optimizes a sparse success reward. Every frame the policy produces is scored, and the moment the success score crosses the model's threshold the episode is declared solved and ends with the reward paid at that step. A single false detection therefore both ends the episode early and pays a reward the policy did not earn. MetaWorld runs use five seeds and Robomimic runs three seeds due to high-computational demands.

Threshold calibration. Turning a continuous success score into this fire-or-not decision requires a threshold. The models emit their scores on different scales, so a single shared cut would favor some models and starve others. We therefore calibrate one threshold per model, with the same procedure for every model, ours and the baselines alike. On a small set of oracle rollouts, collected

once and kept out of training, we select the operating point that recovers the most true successes under a strict false-positive budget. Because the procedure is shared, the calibration cannot be what separates the models, and whatever gap remains belongs to the reward itself. The calibration does read ground-truth labels, a cost for an otherwise zero-shot deployment, but it is a convenience rather than a requirement. Across tasks and simulators, RoboRef's threshold fell between 0.75 and 0.85, and a badly set threshold is visible in the training curves. Too low a threshold produces false detections, early terminations, and a flat success curve. Too high a threshold is never crossed by any episode, successful or not, so the policy receives almost no signal.

Minimum episode length. IBRL rests on off-policy actor-critic updates, which are susceptible to an early-training exploit. The policy learns to provoke a spurious success in the first few steps of an episode, before the task is physically solvable, collects the terminal reward, and resets. We observed this with every reward model, ours and the baselines alike. We therefore ignore any detection before a per-task minimum episode length, set from the length of a demonstration of that task in the set IBRL is pretrained on. This encodes a simple physical prior, that nothing is solved faster than an expert solves it, so an earlier detection is spurious by construction. Every reward model is held to the same minimum. We tried values between 0.8 and 1.0 times the demonstration length and use the best one for each task.

Baseline configurations. RoboReward-4B [13], Robo-Dopamine 2.0 [14], and LRM-8B [15] each run in the configuration their authors intend, with their richest inputs and native reward forms, which favors them. In all three simulators, Robo-Dopamine 2.0 receives a goal image of the completed task and three camera views, while every other model reads one third-person view. The released Robometer-4B is evaluated with its progress head, the better of its two heads in our experiments.

F Real-World Experiments

This appendix expands Section 4.4. Section F.1 gives the setup shared by all real-world tasks and the details of each task, and Section F.2 shows example rollouts.

F.1 Setup

Robot and cameras. We use a Franka FR3 and a ZED 2i camera that publishes 1280×720 frames at 10 Hz. The reward model sees a 300 px square crop centered at pixel $(615, 505)$, resized bicubically to 480×480. The policy sees its own view, a square center crop resized to 128×128. Each clip is sampled to exactly 16 frames for RoboRef and 8 for Robometer-4B, with `linspace` indices over the episode so far. When the episode is shorter we enforce padding by repeating the latest frame.

Reward. Both reward models expose a progress head and a success head, and every step is scored on the *prefix clip* $c_{0:t}$, the frames $0 \dots t$ of the episode so far. The reward is the continuous form

$$r_t \;=\; \mathrm{progress}(c_{0:t}) \;-\; 1 \;+\; p_{\mathrm{success}}(c_{0:t}). \tag{3}$$

The same formula scores the frames recorded after a release, so a release is paid for by what the scene looks like once the object is out of the gripper.

Release protocol. The policy releases once the success head stays above the model's release trigger for 3 consecutive steps, extended to 5 if the arm is still moving substantially during that window, for a steadier release; there is no fixed height gate, since an object dropped from a height can land anywhere and height alone cannot tell a hovering gripper from a seated lid. The arm then retreats, and the outcome is scored on the 10 frames recorded afterward, so occlusion from the gripper cannot influence the reward. The two models use different release thresholds because their success heads are calibrated differently (Table 13); each is evaluated with its own trained threshold, not a shared one. We use 0.75 for RoboRef, and use 0.6 for Robometer-4B which is what the authors reported.

Demonstrations and training. Demonstrations are teleoperated with a GELLO leader arm [54] at 10 Hz and 720p. Frames are subsampled at stride 4 (0.4 s per kept step), and actions are the *realized* pose displacement between kept frames, divided by the action scale $(0.02\ \mathrm{m}, 0.1\ \mathrm{rad})$ and mapped

into the end-effector frame. Transitions that cannot be represented within $[-1, 1]$ are omitted rather than clipped.

Demonstration rewards are not hand-labeled. Each demonstration is scored by the same reward server, with the same prefix-clip logic and Eq. 3, so each run trains on demonstrations relabeled by *its own* reward model. Both runs use RLPD [38], i.e. SAC with a demonstration ratio of 0.5, $\gamma = 0.96$ and mean Q aggregation, with one reward-model call per policy step (`action_repeat` $= 1$).

Evaluation. Success is always the operator's label. For the frozen evaluation, the policy is fixed and learning is off, resets are randomized exactly as in training, and an episode that never releases counts as a failure. Checkpoints are written every 25 global steps and are not selected on performance; we use the last checkpoint for both models.

Pick up the zucchini. The robot must grasp a zucchini lying on the table and lift it clear. The policy's 6-DoF actions move the arm only, with the fingers held open through the 50-step approach; the grasp itself is scripted, closing and then lifting straight up to a fixed height, unless there are 3 consecutive successes detected by the success head of the models. We collected 19 demonstrations for this task, and both models train for 125 episodes.

Close the pot lid. The robot starts holding the lid in a closed gripper and must seat it on the pot and let go. An episode runs at most 50 policy steps. Each reset drives the arm to a pose drawn uniformly from ± 5 cm in x and y about a measured start pose, and orientation is not randomized. Between episodes the operator puts the lid back in the gripper by hand while the arm holds still. For this task we collected 25 demonstrations. Both runs train for 200 episodes. We evaluate RoboRef at `global_step_125` (episode 185) and Robometer-4B at `global_step_100` (episode 188), so the two have seen comparable robot experience. In an attempt to improve the baseline, a second Robometer-4B checkpoint (`global_step_50`, covering its best training window) was evaluated, but it scored 0/30 as well.

Table 13: Per-model settings in the real-world experiments. Both models see the clip of the episode so far, sampled with `linspace` indices, and each run's demonstrations are relabeled by its own reward model.

Task	Reward model	frames	padding	release trigger	success flag
Pick up the zucchini	RoboRef	16	linspace	0.75	0.75
	Robometer-4B	8	linspace	0.60	0.60
Close the pot lid	RoboRef	16	linspace	0.75	0.75
	Robometer-4B	8	linspace	0.60	0.60

F.2 Gallery

Each filmstrip shows one successful RoboRef rollout from the frozen evaluation, from the randomized start to the release, then the retreat and one of the outcome frames the reward model scores. Frames are the full camera view, not the model's crop.

G Extended Results

This appendix collects results that support the main text: the full offline results (Appendix G.1), the reward each model pays on its own policy's states in ManiSkill (Appendix G.2), an intervention showing that false positives cause the RL failure (Appendix G.3), and reward traces on the real robot (Appendix G.4).

G.1 Offline evaluation

Tables 14 and 15 give the full offline results behind Tab. 3. Beyond the accuracy metrics, both decompose each model's error by direction and report, for every metric, Kendall's τ between the ordering of the six models it induces and their RL success. Table 16 adds the success head for the three models that have one.

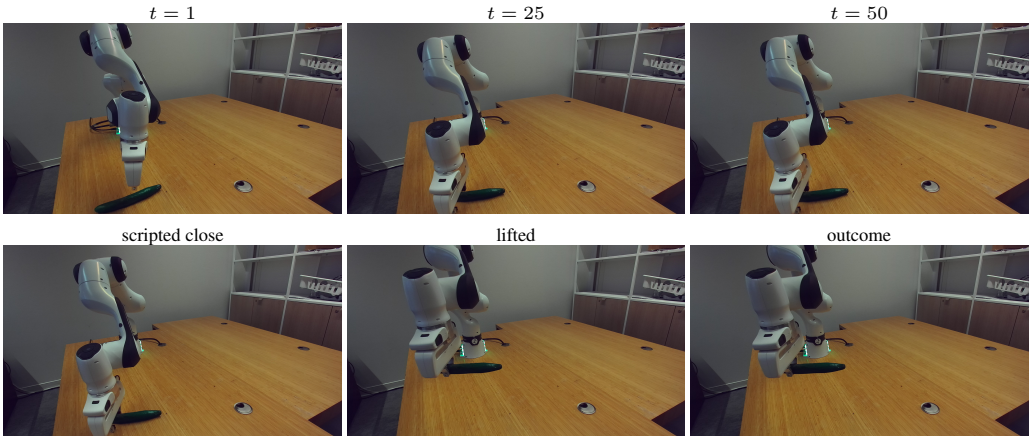


Figure 9: **Pick up the zucchini**, RoboRef, training episode 102. The success head did not cross its release trigger during the 50-step approach (max 0.05), so the scripted grasp closes the fingers and lifts, and the model scores the outcome. On this example, it averaged 0.781 above its 0.75 threshold.

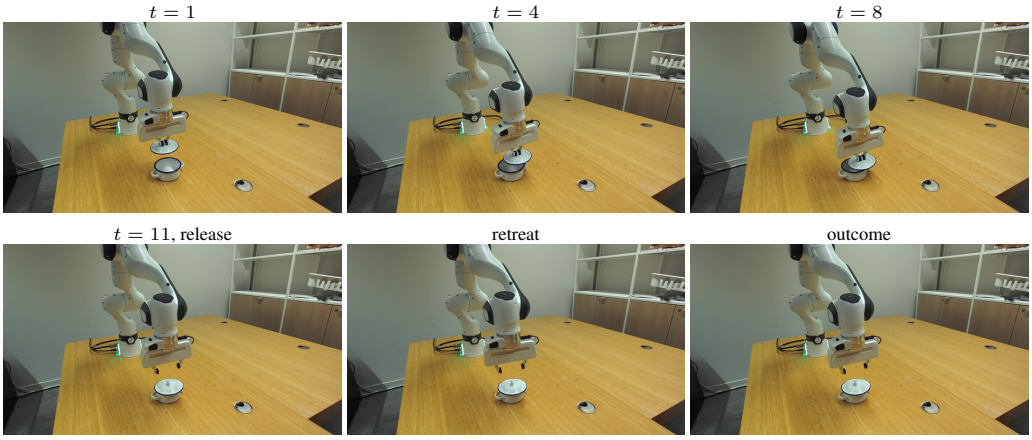


Figure 10: **Close the pot lid**, RoboRef, evaluation episode 19. Eleven policy steps take the robot from the randomized start to the release. The success head averaged 0.787 over the outcome frames, and the operator labeled the lid seated.

Where RoboRef's error lies. On RoboRewardBench, RoboRef assigns the top level to 0.1% of non-successful episodes and a level of four or higher to 1.3% of episodes labeled below four. The baselines do so on 5–13% and 7–24%. In exchange RoboRef misses 19% of true successes, twice the rate of Robometer-4B, and its under-prediction is the largest of all models. Our control separates the two ingredients as in Sec. 4.2. Robometer + Failures already matches RoboRef on top-level false positives, so the failure data removes those. The asymmetric loss is what lowers the reward on non-successes across the range, from 22.9% to 1.3% on the four-or-higher rate and from 0.37 to 0.18 on the mean raw score.

Robomimic. Here the split should be read with the label in mind. The target is the frame index, so a model that scores a lifted can at frame $T/2$ above 0.5 counts as over-predicting. RoboRef does so more than any baseline, and the model that reproduces the ramp most faithfully is the control, which was trained on that ramp and trains no policy. The split is also small, 160 frames from ten demonstrations, and the level-based rates in Tab. 15 rest on 20 success frames. On the success head (Tab. 16), all three models separate success frames from the rest perfectly. They differ only in calibration. The control's head never crosses 0.5 on a success frame, RoboRef's does so on 85% of them, and Robometer-4B's on all. In RL this is absorbed by the per-model threshold calibration of App. E.3.

Table 14: **RoboRewardBench, full results.** Progress head for all models except RoboReward-4B, which returns levels directly. MAE_r after rescaling outputs to $[0, 1]$ (App. E.2). Over and Under are mean over- and under-prediction of rescaled progress. FP_5 is the fraction of non-successes scored 5, $FP_{\geq 4}$ the fraction of episodes below level 4 scored 4 or higher, FN the fraction of successes scored below 5. Reward on failures is the mean score on non-successes, raw and rescaled. The last rows give Kendall's τ between each metric's model ranking and RL success (area under the training curve, four tasks), oriented so $+1$ means agreement, with an exact permutation p.

	Accuracy				Error direction					Reward on failures	
Model	**MAE↓**	**MAE_r↓**	**Rank↑**	**AUROC↑**	**Over↓**	**Under↓**	**FP_5↓**	**$FP_{\geq 4}$↓**	**FN↓**	**raw↓**	**resc.↓**
Robometer-4B	0.961	0.968	0.803	0.859	0.491	**0.163**	0.047	0.198	0.092	0.497	0.507
Robo-Dopamine 2.0	1.242	1.339	0.597	0.601	0.571	0.209	0.133	0.240	0.117	0.223	0.611
RoboReward-4B	**0.405**	**0.405**	**0.856**	**0.905**	**0.166**	0.196	0.061	0.069	**0.050**	0.339	0.339
LRM-8B	1.056	1.056	0.700	0.721	0.167	0.454	0.070	0.096	0.129	0.292	**0.292**
Robometer + Failures	1.065	1.063	0.723	0.760	0.437	0.283	**0.001**	0.229	0.186	0.374	0.489
RoboRef (ours)	1.350	1.068	0.718	0.774	0.249	0.463	**0.001**	**0.013**	0.186	**0.177**	0.320
Kendall τ vs. RL	-0.87	-0.73	-0.47	-0.33	-0.20	-0.60	-0.07	-0.07	-0.60	$+0.73$	$+0.07$
p	0.017	0.056	0.272	0.469	0.719	0.136	1.000	1.000	0.136	0.056	1.000

Table 15: **Robomimic `PickPlace-Can`, progress head.** Ten IBRL demonstrations of sixteen frames, target t/T, success = last level (20 frames). Columns as in Tab. 14, with MAE on progress in $[0, 1]$ and rates over frames.

	Accuracy				Error direction					Reward on failures	
Model	**MAE↓**	**MAE_r↓**	**Rank↑**	**AUROC↑**	**Over↓**	**Under↓**	**FP_5↓**	**$FP_{\geq 4}$↓**	**FN↓**	**raw↓**	**resc.↓**
Robometer-4B	**0.101**	0.083	0.958	**1.000**	0.037	0.212	0.013	**0.000**	**0.000**	0.386	0.372
Robo-Dopamine 2.0	0.112	0.112	**0.990**	0.990	**0.000**	0.438	**0.000**	**0.000**	0.062	0.326	0.326
RoboReward-4B	0.347	0.214	0.796	0.772	0.019	0.625	0.006	**0.000**	0.106	**0.134**	**0.268**
LRM-8B	0.162	0.140	0.924	0.929	0.131	0.481	0.131	**0.000**	**0.000**	0.276	0.306
Robometer + Failures	0.180	**0.064**	0.983	0.944	0.237	**0.006**	0.087	0.075	0.006	0.291	0.497
RoboRef (ours)	0.328	0.167	0.857	0.872	0.512	0.113	0.044	0.275	0.075	0.164	0.570
Kendall τ vs. RL	-0.20	-0.07	$+0.20$	-0.07	-0.33	$+0.33$	$+0.07$	-0.78	-0.43	$+0.07$	-0.47
p	0.719	1.000	0.719	1.000	0.469	0.469	1.000	0.200	0.356	1.000	0.272

Fig. 11 shows the same disconnect on this split: neither offline MAE nor rank accuracy on the expert demonstrations correlates with Robomimic RL success ($\tau = -0.07$ and $\tau = -0.20$, both not significant), the same pattern as on RoboRewardBench (Tab. 3); RoboReward-4B's coarse discrete 1–5 output limits its rank accuracy here in particular.

Agreement with RL. Sec. 4.3 correlates the columns of Tab. 3: rescaled RoboRewardBench MAE against final RL success averaged over the four simulated tasks. Of the 15 model pairs, 14 are ordered oppositely by the two ($\tau = -0.87$, $p = 0.017$). The τ rows of Tabs. 14 and 15 instead measure RL success as the area under the training curve, which also credits how fast a policy learns. The conclusions agree: rescaled MAE gives $\tau = -0.73$ ($p = 0.056$), and the raw reward on non-successful episodes gives $\tau = +0.73$ ($p = 0.056$), against $+0.60$ ($p = 0.136$) with final success. The τ rows use the same convention throughout. RL success is the area under the training curve averaged over the four simulated tasks, each metric is oriented so that $+1$ means its better-is direction agrees with RL, and p is an exact permutation test. Every accuracy metric on RoboRewardBench is negative, from -0.33 for AUROC to -0.87 for the benchmark's own MAE, and on Robomimic none exceeds $|\tau| = 0.2$. The quantity that leans most clearly toward RL is the mean raw reward on non-successful episodes, $\tau = 0.73$, the offline counterpart of the failure reward in Fig. 12b. It holds on the raw scale only, which is the scale a policy receives, and disappears after min–max rescaling. With six models and many metrics no single row is conclusive on its own. The finding is the sign pattern. Metrics that reward fitting the labels order the models against RL, and the one that measures the reward given on failures orders them with it.

Table 16: **Robomimic `PickPlace-Can`, success head.** Same frames as Tab. 15, for the three models that deploy Robometer's success head. FP and FN threshold the head at 0.5. Mean $\hat{s}$ is the mean success probability on non-success frames and on frames at levels 1–2.

						Mean $\hat{s}$	
Model	**AUROC↑**	**TPR@FPR5↑**	**FP@0.5↓**	**FN@0.5↓**	**ECE↓**	**non-succ.↓**	**level 1–2↓**
Robometer-4B	**1.000**	**1.000**	**0.000**	**0.000**	**0.047**	0.043	0.005
Robometer + Failures	**1.000**	**1.000**	**0.000**	1.000	0.080	**0.009**	**0.000**
RoboRef (ours)	**1.000**	**1.000**	**0.000**	0.150	0.059	0.025	**0.000**

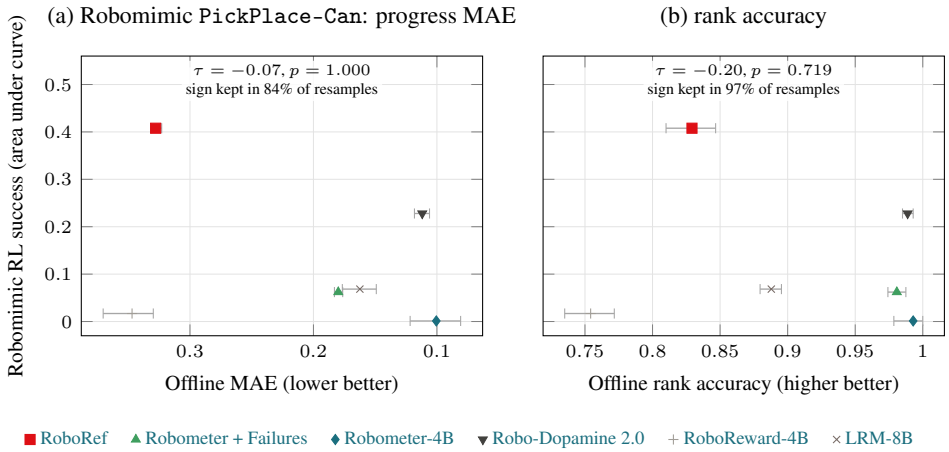


Figure 11: **Offline accuracy does not predict RL on Robomimic either.** Each model's per-frame progress on the ten `PickPlace-Can` expert demonstrations IBRL trains from (agentview and wrist views, 16 frames, re-rendered from the saved simulator states at 224 px; Robometer-4B read at the same 8 of 16 frames as its RL run), by *(a)* MAE and *(b)* rank accuracy against true progress t/T, versus Robomimic RL success as area under the training curve. Error bars are 95% bootstrap intervals over the ten demonstrations.

G.2 Reward hacking analysis

All numbers in Table 17 come from the training episodes of the dense-reward runs in Section 4.1. Models are scored with their progress head, except RoboReward-4B, which runs on the five levels of its success head. No model terminates episodes in this regime, so each score is logged as returned and the FP threshold is applied after training, for measurement only. We restrict FP to never-solved episodes, since a progress model should score highly just before a real success. Solve rates and step counts differ across rows because each reward trains a different policy, and both metrics are computed within each model's own episodes. The baselines report scores on a coarse grid (LRM-8B to one decimal place, Robo-Dopamine 2.0 as small fractions, RoboReward-4B on five levels), so their scores contain many ties. AUROC assigns tied scores their average rank.

Figure 12 shows the same training logs as curves. Panel (a) is the mean predicted reward over the course of never-solved episodes, (b) its mean over the episode, and (c) the AUROC column of Table 17.

G.3 False positives cause the RL failure

Section 4.2 and the reward-hacking analysis of Appendix G.2 show that the rewards which fail to train a policy pay success-level reward in failed episodes. That evidence is correlational. Here we test the causal claim directly. We take Robometer-4B, whose dense-reward runs on ManiSkill `PullCube-v1` never learn the task (Fig. 3a), remove its false positives and nothing else, and train again.

Intervention. We rerun SAC with the recipe of Fig. 3a (Appendix E.3) and change only the reward. At every step the simulator reports as not solved, a Robometer-4B reward at or above its success

Table 17: On-policy reward quality on ManiSkill `PullCube-v1`, pooled over five seeds. *Solved*: share of steps the simulator marks solved. *AUROC*: probability that a successful step scores above an unsuccessful one. *FP*: share of steps in failed episodes that reach the 5th percentile of scores at successful steps.

Model	Steps	Solved	AUROC↑	FP↓
Robometer-4B	1,499,380	0.007	0.429	0.893
Robo-Dopamine 2.0	1,582,388	0.046	0.777	0.639
RoboReward-4B	1,469,792	0.027	0.563	0.956
LRM-8B	1,499,380	0.029	0.640	0.850
Constant	–	–	0.500	1.000
Robometer + Failures	1,499,380	0.007	0.660	0.835
RoboRef (ours)	1,499,380	0.131	**0.919**	**0.393**

threshold ($\tau = 0.654$, the threshold behind its FP rate in Appendix G.2) is clipped to just below τ. Rewards at solved steps and all rewards below τ are left as the model returned them, so the policy still learns from Robometer-4B's own progress estimates, with its ordering of unsolved states intact and only the success-level band removed. The mask uses the simulator's success signal and is therefore an oracle, a diagnostic rather than a method. It fires on 40–46% of all training steps across the three seeds, which is how much of Robometer-4B's reward is paid at success level for states that are not successes. As an upper bound we also train with the environment's ground-truth dense reward.

Result. Figure 13 shows the three rewards side by side. Unchanged, Robometer-4B ends at 0.00 success on all five seeds, and its mean over seeds never exceeds 0.04 at any evaluation. With its false positives masked, the same model reaches 0.92 ± 0.02 (three seeds), close to the ground-truth reward's 0.94 ± 0.02 (five seeds). It learns more slowly, passing 0.5 success at 80k steps against 50k for the ground-truth reward. Robometer-4B's progress signal below the threshold is informative enough to train a competent policy. What prevents learning is the success-level reward it pays in states that are not successes, which is the error RoboRef's asymmetric objective targets.

G.4 Reward traces on the real robot

Figure 14 is the real-robot counterpart of Figure 4: a failed and a successful close-lid episode from the Robometer-4B run of Section 4.4, which saved every frame its reward model saw. Both models are re-run offline on these frames with the online protocol of Appendix F.1, and on these two episodes the recomputed scores reproduce the scores logged online to within 0.05. In the failed episode (a), Robometer-4B detects success during the approach and the lid is released off the seat, while RoboRef never detects success. In the successful episode (b), both models detect success, RoboRef at the step the lid is released.

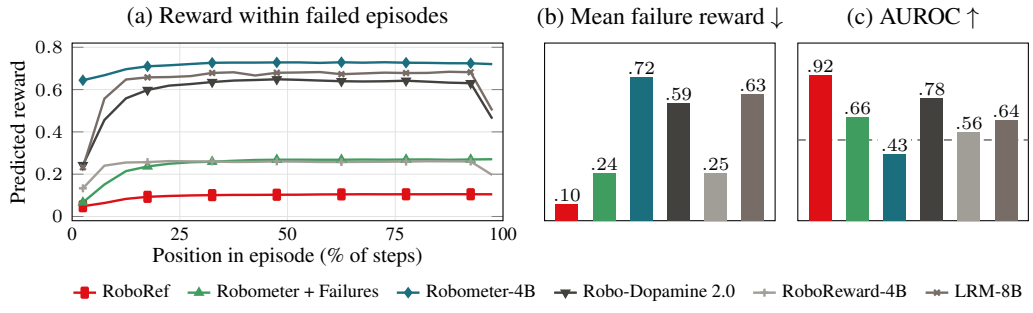


Figure 12: **Reward predictions on ManiSkill.** *(a)* Mean predicted reward over the course of never-solved training episodes (1,500 per model) and *(b)* its mean over the episode. *(c)* AUROC of successful vs. unsuccessful steps over all training steps, dashed line at chance. Pooled over five seeds.

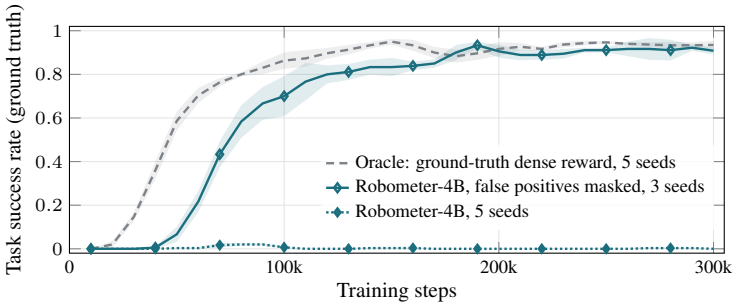


Figure 13: **Masking false positives rescues RL.** SAC on ManiSkill `PullCube-v1` with the dense-reward recipe of Fig. 3a. Masking clips Robometer-4B's reward to just below its success threshold at every step the simulator reports as unsolved, and changes nothing else. Curves are smoothed and shaded as in Fig. 3, mean over seeds with a one standard error band.

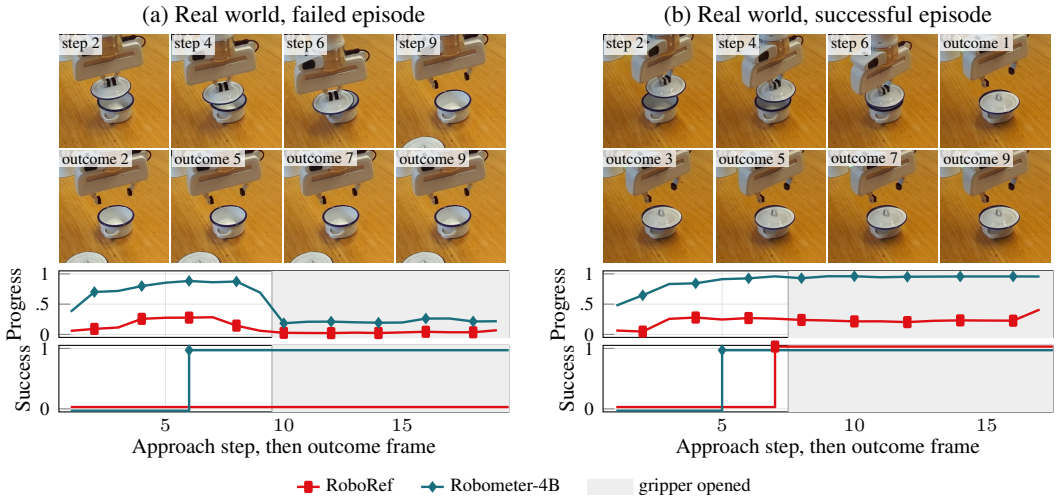


Figure 14: **Example reward predictions on the real robot** of RoboRef and Robometer-4B on a failed (a) and a successful (b) close-lid episode, the counterpart of Figure 4. Top: predicted progress $\in [0, 1]$. Bottom: success, held at 1 once the success probability has exceeded the model's release threshold (RoboRef 0.75, Robometer-4B 0.6) for three consecutive steps. Gray: frames after the gripper opens.