LEARNING OPTIMAL ROBOT REWARDS FROM IMPERFECT TRAJECTORIES WITH QUASIMETRICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Video reward models for robot learning are typically trained to predict temporal progress, the relative position of each frame within a demonstration. However, this target measures the demonstrator's value rather than the optimal one. Pauses and detours therefore distort the target, and failed attempts yield no target at all, since the time to success is never observed. We propose **Quasimetric Reward Model (QRM)**, which targets the optimal goal-reaching value by learning from bounds rather than regression targets. Each observed video segment is a feasible path whose return provides a lower bound on the optimal value, even when the trajectory is inefficient or failed. A quasimetric model combines these bounds through the triangle inequality to tighten its value estimates. Such a value is defined for a single goal state, whereas a language instruction admits many successful outcomes and provides no goal image for a new task. QRM therefore distills the value of reaching the nearest successful training outcome into a compact model conditioned on visual history and language. Its dense reward complements the sparse success signal during reinforcement learning. On MetaWorld tasks unseen during reward model training, QRM improves its policy success rate with imperfect videos that degrade temporal-progress models, outperforming vision-language reward models $50\times$ its size. For fine-tuning a pretrained robot policy, QRM yields $1.8\times$ the success rate of the strongest baseline on held-out LIBERO-90 tasks and matches or exceeds every baseline on real-robot manipulation tasks learned from few demonstrations.

1 INTRODUCTION

In reinforcement learning, the reward specifies the task to the learner: it defines which behavior is optimal and governs how quickly that behavior is learned. Adapting a robot policy to a new task therefore requires a reward that is correct and informative. A task success signal is correct but uninformative before completion, whereas a hand-designed dense reward is informative but must be engineered for every task. Reward models learned from robot videos offer an alternative: trained once on a video corpus, they estimate progress on new tasks from images and a language instruction (Ma et al., 2023a; Zhang et al., 2025; Tan et al., 2026). Scaling these models calls for learning from broader video corpora, which contain hesitations, detours, recoveries, and failures alongside successful executions (Lee et al., 2026). A scalable reward model must benefit from this diversity.

Most video reward models are trained to predict *temporal progress*, the relative position of a frame within its demonstration, or the remaining time to completion (Yang et al., 2024; Zhang et al., 2025; Ayalew et al., 2025; Liu et al., 2026; Tan et al., 2026; Huang et al., 2026). Fitting a model to this target estimates the *demonstrator's* value, which matches the optimal value only when the demonstrator follows shortest paths. Yet videos inevitably contain imperfect demonstrations: pauses inflate remaining time, detours give revisited states conflicting targets, and failures even provide no completion time. Recent methods add supervision by rewinding videos (Zhang et al., 2025), comparing trajectories (Liang et al., 2026), or predicting annotated outcomes (Lee et al., 2026), but temporal targets still reflect the demonstration's quality.

We instead target the optimal goal-reaching value and supervise it with bounds rather than temporal progress targets. Every observed video segment is a feasible path between its endpoints, so its return *lower-bounds* the optimal value between them. The bound holds regardless of how the segment was produced. A detour yields a looser bound, and a failed attempt still bounds the value between

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

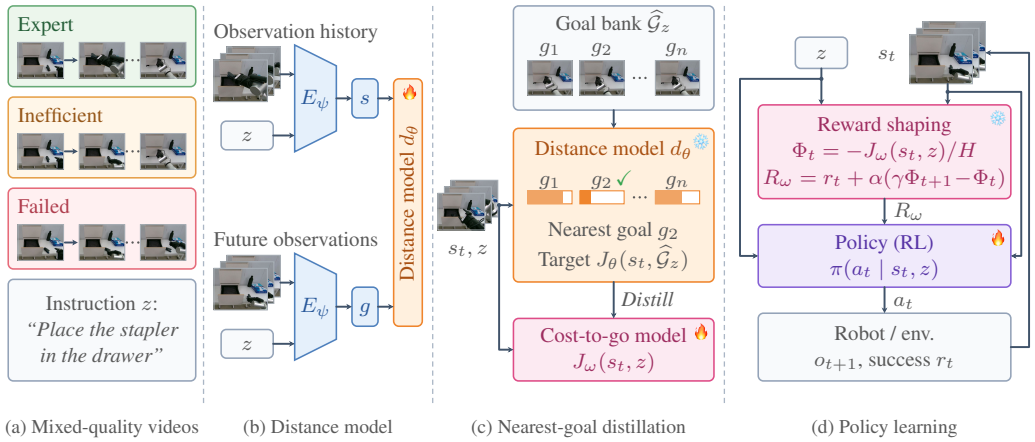


Figure 1: Overview of QRM. QRM learns directed goal reaching distances from observed paths and distills them into a reward model that uses visual history and language, without requiring a goal image during policy learning.

the states it visits. Optimal goal-reaching values have a geometric structure: the minimum cost of reaching one state from another obeys the triangle inequality but need not be symmetric, and is therefore a *quasimetric* (Wang & Isola, 2022b). Bounds from different segments can be combined through the triangle inequality, and additional paths of any quality can only tighten the estimate. QRL (Wang et al., 2023) and MAD (Steccanella et al., 2026) use this formulation for goal-conditioned control, learning the tightest value estimates consistent with the observed bounds.

Unlike goal-conditioned control, where the goal state is given as an input, video reward models receive the task only as a language instruction. We propose **Quasimetric Reward Model (QRM)**, which turns the quasimetric value into a language-conditioned reward in two stages. QRM first learns a quasimetric value function from segment bounds on mixed-quality videos. It then distills this value into a compact model conditioned on visual history and language, so that no goal image is needed. For each instruction, QRM collects the final states of its successful training videos in a goal bank, and the distillation target is the value of reaching the nearest of them. This target covers the many configurations that satisfy an instruction and is defined even for failed trajectories, whose own endpoints are not goals. During policy learning, the distilled model requires no goal image, goal retrieval, or vision-language model inference, and its dense reward, supplied through potential-based shaping (Ng et al., 1999), complements the sparse success signal.

Our contributions are as follows. (i) We propose to supervise video reward models with segment bounds on optimal goal-reaching values instead of temporal progress targets, which estimate the demonstrator's value rather than the optimal one. The bounds hold for every trajectory regardless of its quality, so additional videos of any quality can only tighten the estimate. (ii) We introduce QRM, which turns a goal-conditioned quasimetric value into a language-conditioned reward model by distilling the value of reaching the nearest successful training outcome. The target accommodates multiple valid outcomes and failed trajectories alike, and the distilled model needs no goal image during policy learning. (iii) We demonstrate that QRM benefits from inefficient and failed videos and transfers to unseen tasks. With 171M parameters, QRM yields an interquartile-mean success rate of 0.927 on unseen MetaWorld tasks, versus 0.853 for the strongest baseline, a vision-language reward model $50\times$ its size. When fine-tuning a pretrained policy, it achieves $1.8\times$ the best baseline's success on held-out LIBERO-90 tasks and matches or beats all baselines on three real-robot tasks.

2 Related Work

Temporal-progress reward models. Video reward models commonly regress temporal targets, including rank (Yang et al., 2024), normalized progress (Zhang et al., 2025; Ayalew et al., 2025), progress differences (Liu et al., 2026), or progress within annotated task stages (Kim et al., 2025;

Chen et al., 2026a; Tan et al., 2026). RynnValue (Huang et al., 2026) predicts language-conditioned temporal distance from timestamps and converts it into rewards through potential-based shaping (Ng et al., 1999), as do RoboDopamine (Tan et al., 2026) and QRM. QRM shares this interface but bounds optimal value using segment returns rather than temporal progress equality targets.

Rewards from imperfect and failed videos. A common strategy is to recover supervision from imperfect data through additional labels. RoboReward (Lee et al., 2026) annotates episodes with discrete outcome scores, while RoboMeter (Liang et al., 2026) combines expert progress targets with trajectory comparisons involving failed and suboptimal behavior. Earlier work uses human preferences and trajectory rankings (Christiano et al., 2017; Lee et al., 2021; Brown et al., 2019; 2020), or success and failure in online rollouts (Memarian et al., 2021; Tsao et al., 2026). Large vision-language models can also score progress without task-specific training (Rocamonde et al., 2024; Ma et al., 2025; Chen et al., 2026b), but require billion-parameter inference during reinforcement learning. QRM uses the temporal structure of videos and success flags for goal-bank construction, without graded outcome scores or trajectory comparisons. This can provide dense supervision even with failed episodes.

Value learning from action-free video. VIP (Ma et al., 2023b) frames action-free representation learning as the dual of goal-conditioned offline RL and uses negative embedding distance to a goal image as a value. LIV (Ma et al., 2023a) extends this to language goals. ICVF (Ghosh et al., 2023), V-PTR (Bhateja et al., 2024), and HILP (Park et al., 2024) learn value functions and temporal-distance-preserving representations using temporal-difference objectives for policy initialization or representation. VIP and LIV are closest to QRM because they produce rewards, but their objectives are regularized toward the data distribution, leaving values dependent on demonstration quality. QRM instead uses valid bounds from mixed-quality videos, models asymmetric costs, and distills the distance to the nearest of many successful outcomes.

Quasimetric distance learning. Optimal goal-reaching costs satisfy the triangle inequality without being symmetric, which motivates quasimetric models of goal-conditioned values (Pong* et al., 2018; Eysenbach et al., 2019; Wang & Isola, 2022b; Liu et al., 2023a; Wang & Isola, 2022a). QRL (Wang et al., 2023) maximizes distances subject to observed cost bounds, and MAD (Steccanella et al., 2026) learns from segment-length bounds in action-free trajectories of arbitrary quality. Related work extends quasimetric learning to stochastic settings and incorporates multistep returns (Myers et al., 2025; Zheng et al., 2026). These methods learn distances to a *given goal state* for control in the environment where the data were collected. QRM distills distance to the nearest successful training outcome into a language-conditioned reward model that needs no goal image during policy learning.

3 From Temporal Progress to Optimal Goal-Reaching Values

We formalize video reward learning as goal-conditioned reinforcement learning (GCRL). In this view, temporal-progress regression estimates the demonstrator's value, whereas the cost of every observed segment bounds the optimal value, and quasimetric structure combines these bounds. Such values presume a goal state that a language-specified task does not provide. QRM therefore distills the value of reaching the nearest successful training outcome into a language-conditioned reward model.

3.1 Cost-to-go and distances

For instruction z, consider the goal-conditioned MDP $\mathcal{M}_z = (\mathcal{S}, \mathcal{A}, P, c, \mathcal{G}_z)$ (Kaelbling, 1993; Schaul et al., 2015; Andrychowicz et al., 2017), with state space $\mathcal{S}$, action space $\mathcal{A}$, transition model P, and nonnegative transition cost $c(s_t, a_t, s_{t+1})$. The goal set $\mathcal{G}_z \subseteq \mathcal{S}$ contains states satisfying z. We assume deterministic dynamics and undiscounted returns throughout this section. A policy $\pi(a \mid s, \mathcal{G}_z)$ accumulates cost until it first reaches the goal set at $T_{\mathcal{G}_z} = \min\{t \geq 0 : s_t \in \mathcal{G}_z\}$. Its cost-to-go and the optimal cost-to-go are

$$J^\pi(s, \mathcal{G}_z) = \mathbb{E}_\pi \left[\sum_{t=0}^{T_{\mathcal{G}_z}-1} c(s_t, a_t, s_{t+1}) \,\middle|\, s_0 = s \right], \qquad J^\star(s, \mathcal{G}_z) = \inf_\pi J^\pi(s, \mathcal{G}_z), \tag{1}$$

with infinite cost if the goal is never reached and zero cost after arrival. Note that since $V^\pi = -J^\pi$ and $V^\star = -J^\star$, an upper bound on optimal cost is a lower bound on optimal value.

The optimal cost-to-go can be expressed by distances between states (Eysenbach et al., 2019; Wang et al., 2023). For a feasible trajectory $\tau = (s_0, a_0, \ldots, a_{T-1}, s_T)$, let $C(\tau) = \sum_{t=0}^{T-1} c(s_t, a_t, s_{t+1})$ be its total cost. For states $s, g \in \mathcal{S}$, let $\mathfrak{T}(s, g)$ be the feasible trajectories from s to g. We define the minimum path cost as $d^\star_{\text{env}}(s, g) \triangleq \inf_{\tau \in \mathfrak{T}(s,g)} C(\tau)$. Since reaching any state in $\mathcal{G}_z$ completes the task, the optimal cost-to-go is the smallest distance to a goal state (Appendix A.2.1),

$$J^\star(s, \mathcal{G}_z) = \inf_{g \in \mathcal{G}_z} d^\star_{\text{env}}(s, g). \tag{2}$$

3.2 Temporal progress measures the demonstrator, not the task

Temporal progress encodes a state's temporal position within a demonstration, rather than its optimal cost-to-go. For a successful demonstration ending at step T, a model $f(s_t, z)$ fits each frame t to normalized progress $p_t = t/T$ (Zhang et al., 2025; Ayalew et al., 2025) or its remaining time $T - t$ (Huang et al., 2026). With unit costs, $J^\star(s_t, \mathcal{G}_z) = T - t$ if and only if the remainder of the demonstration is a shortest path to $\mathcal{G}_z$.

In practice, however, imperfect demonstrations violate this condition. The remaining time $T - t$ is the cost incurred by demonstrator policy π_β from s_t. Least-squares regression predicts its conditional mean, recovering the demonstrator's cost-to-go rather than the optimal one,

$$f(s, z) = \mathbb{E}_{\pi_\beta}[\, T - t \mid s_t = s \,] = J^{\pi_\beta}(s, \mathcal{G}_z) \geq J^\star(s, \mathcal{G}_z), \tag{3}$$

with equality only if the demonstrator follows a shortest path from s (Appendix A.2.2). Normalized progress also depends on total demonstration length, assigning different targets to the same remaining time. Inefficient demonstrations can therefore distort both targets.

Every observed trajectory segment $\tau_{i:j}$ instead provides a bound for the distance between s_i and s_j because it is a feasible path,

$$d^\star_{\text{env}}(s_i, s_j) \leq C(\tau_{i:j}), \qquad 0 \leq i < j \leq T. \tag{4}$$

This bound holds for any policy that generates trajectories (Appendix A.2.3). Pauses, detours and even failed attempts can still provide bounds along their paths. We therefore give supervision using upper bounds obtained by segment costs rather than a temporal progress.

3.3 Combining bounds through quasimetric structure

Quasimetric structure can combine those loose segment bounds into tighter estimates of shortest-path costs. With nonnegative transition costs, $d^\star_{\text{env}}$ is a quasimetric (Wang & Isola, 2022b; Wang et al., 2023), satisfying $d^\star_{\text{env}}(s, s) = 0$ and the triangle inequality

$$d^\star_{\text{env}}(s, g) \leq d^\star_{\text{env}}(s, u) + d^\star_{\text{env}}(u, g) \qquad \text{for all } s, u, g \in \mathcal{S}, \tag{5}$$

but need not be symmetric (Appendix A.2.4). The triangle inequality combines bounds across trajectories. The costs of segments from s to u and from u to g together bound $d^\star_{\text{env}}(s, g)$, even if no observed trajectory connects s to g. Even an inefficient trajectory can contain efficient segments that provide tight bounds.

We therefore parameterize d_θ as a quasimetric to enforce the triangle inequality. As the zero function satisfies every upper bound, the bounds alone cannot determine the distance. Thus, we take the largest quasimetric consistent with them following quasimetric reinforcement learning (QRL) (Wang et al., 2023). Let $\hat{d}_\mathcal{D}$ denote the largest quasimetric satisfying Eq. (4) for every segment. Assuming these constraints are enforced exactly and the quasimetric class is unrestricted, $\hat{d}_\mathcal{D}(s, g)$ equals the cost of the shortest path from s to g formed by joining observed segments. Thus, adding trajectories provides more paths and can only tighten this estimate. For $\mathcal{D} \subseteq \mathcal{D}'$,

$$d^\star_{\text{env}}(s, g) \leq \hat{d}_{\mathcal{D}'}(s, g) \leq \hat{d}_\mathcal{D}(s, g). \tag{6}$$

The estimate equals $d^\star_{\text{env}}(s, g)$ when the joined segments can form a shortest path (Appendix A.2.5).

3.4 From goal-conditioned distances to a task reward

A learned distance $d_\theta(s, g)$ evaluates a single goal, whereas task cost-to-go $J^\star(s, \mathcal{G}_z)$ depends on the goal set through Eq. (2). This raises two difficulties. First, $\mathcal{G}_z$ is not given explicitly. We therefore cannot directly minimize the learned distance over all states that satisfy the instruction. Second, new tasks provide no goal image, and synthesizing one (Black et al., 2024) requires a costly model. QRM instead approximates $\mathcal{G}_z$ by successful training outcomes and distills the resulting estimate of $J^\star(s, \mathcal{G}_z)$ into a model that takes only s and z.

4 Quasimetric Reward Model

QRM is trained in two stages on videos paired with language instructions (Figure 1). The first stage learns a quasimetric between states from segment bounds. The second stage distills the distance to the nearest successful outcome into a language-conditioned cost-to-go model, which requires no goal image. Once trained, the cost-to-go model can serve a policy learner as a shaping potential alongside a sparse task reward. Appendix A.3 gives the practical implementation details.

4.1 Learning a quasimetric from observed paths

Inspired by QRL (Wang et al., 2023), we learn a quasimetric d_θ from the segment bounds in Eq. (4), which remain valid for inefficient and failed trajectories. Let ν sample trajectory segments $\tau_{i:j}$ spanning a few time steps $(i < j)$ for the constraints and μ sample state pairs (s, g) within the same trajectories for distance maximization. We solve

$$\begin{aligned} \max_{\theta} \quad & \mathbb{E}_{(s,g)\sim\mu}\left[\log(1 + d_\theta(s, g))\right] \\ \text{subject to} \quad & \mathbb{E}_{\tau_{i:j}\sim\nu}\left[\left[d_\theta(s_i, s_j) - C(\tau_{i:j})\right]_+^2\right] \le \epsilon_{\text{QRL}}^2, \end{aligned} \tag{7}$$

where $C(\tau_{i:j})$ is the observed segment cost, $[x]_+ = \max\{x, 0\}$, and $\epsilon_{\text{QRL}} \ge 0$ controls the allowed violation. The maximization selects the largest quasimetric consistent with the segment bounds of Eq. (4), since the one-sided constraints alone would admit the zero distance. We show that the learned distance composes segments from different trajectories into a shorter route and justify the choice of our practical trajectory sampler μ (Appendix A.4).

4.2 Nearest goal distillation into a language-conditioned reward model

Nearest goal distillation approximates $\mathcal{G}_z$ by successful training outcomes and trains a language-conditioned cost-to-go model to predict the distance to the nearest of them from the observation history and the instruction alone. Let

$$J_\theta(s, \mathcal{G}_z) = \inf_{g\in\mathcal{G}_z} d_\theta(s, g) \tag{8}$$

denote the learned counterpart of Eq. (2). The goal set $\mathcal{G}_z$ is unknown, but the final state of every successful training trajectory for instruction z lies in $\mathcal{G}_z$. For a query state whose trajectory starts from the observation o_0, we retrieve the final states of successful trajectories of the same task whose initial observations are closest to o_0 in the feature space of the visual encoder. The retrieved final states form the *goal bank* $\widehat{\mathcal{G}}_z(o_0)$. The distillation target $J_\theta(s, \widehat{\mathcal{G}}_z(o_0))$ is obtained by substituting the goal bank for $\mathcal{G}_z$ in Eq. (8). This supplies targets for states from failed trajectories whenever matching successful outcomes are available. Because the learned distances are unbounded, we use clipping at a horizon $H > 0$ to let the shaping potential of Section 4.3 be $[-1, 0]$, while preserving the quasimetric property (Appendix A.1.1).

The language-conditioned cost-to-go model $J_\omega(s, z) \in [0, H]$ is then trained to predict this target with the distance model frozen,

$$\mathcal{L}(\omega) = \mathbb{E}_{(s,z)\sim\eta}\left[\text{Huber}_\delta\left(J_\omega(s, z) - J_\theta\left(s, \widehat{\mathcal{G}}_z(o_0)\right)\right)\right] + \lambda_0\, \mathbb{E}_{(s,z)\sim\eta_{\text{succ}}}\left[J_\omega(s, z)^2\right], \tag{9}$$

where η samples training states with nonempty banks, η_{succ} samples successful training endpoints in the target environment, $\delta > 0$ is the Huber threshold, and $\lambda_0 \ge 0$ weights the second term. This term anchors successful endpoints at zero, consistent with $J^\star(s, \mathcal{G}_z) = 0$ for $s \in \mathcal{G}_z$. Once trained, the cost-to-go model J_ω requires neither a goal image nor the goal bank. Appendices A.3.4 and A.1 give training details and approximation error analysis.

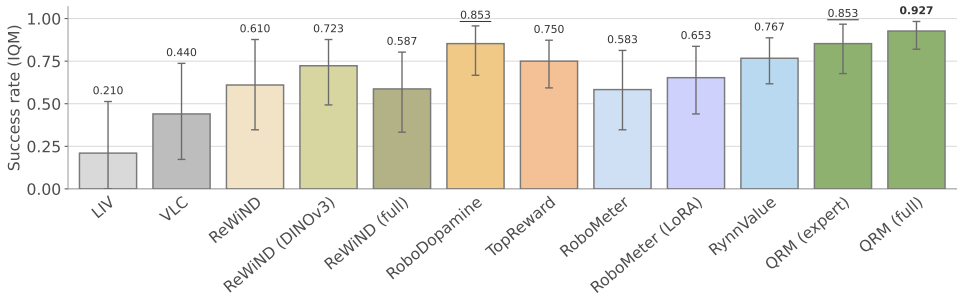


Figure 2: **Rewards for unseen MetaWorld tasks.** Online RL success rate after 100,000 steps. Bars show IQM over eight tasks and three policy seeds, with 95% confidence intervals.

4.3 Policy learning with the learned reward

The trained cost-to-go model J_ω provides a dense reward for reinforcement learning on a new task along with a sparse success signal. Its normalized potential $\Phi_\omega(s, z) = -J_\omega(s, z)/H \in [-1, 0]$ enters the reward via potential-based reward shaping (PBRS) (Ng et al., 1999). Let r_t^{success} be the sparse reward paid on the first success of an episode. We train the policy with the shaped reward

$$R_\omega(s_t, s_{t+1}, z) = r_t^{\text{success}} + \alpha \left[\gamma \widetilde{\Phi}_\omega(s_{t+1}, z) - \Phi_\omega(s_t, z)\right], \tag{10}$$

where $\alpha \geq 0$ scales the shaping term, γ is the discount factor of the policy learner, and $\widetilde{\Phi}_\omega$ equals Φ_ω except at terminal states, where it is zero. Episodes truncated by a time limit retain the potential of the next state. Under these conditions, the optimal policies of the sparse-success objective are invariant to potential-based shaping (Appendix A.1.4).

5 Experiments

We evaluate whether QRM provides useful rewards for unseen tasks (Q1), improves large pretrained robot policies (Q2), and benefits from suboptimal videos (Q3). We then examine how its key components contribute to performance (Q4). Following the evaluation protocol of prior video reward models (Zhang et al., 2025; Tan et al., 2026; Liang et al., 2026; Huang et al., 2026), we assess each reward model by the success rate of the policies trained with it. For simulation environments, we report success rate using interquartile mean (IQM) and 95% confidence intervals following Agarwal et al. (2021); Zhang et al. (2025). Dataset, training, and evaluation details are in Appendix A.5, with per-task learning curves in Appendix A.13.

5.1 Q1. Does QRM provide useful rewards for unseen tasks?

Designing a reward for every new task is difficult to scale, which motivates reward models that transfer to unseen tasks without additional reward model training. We test this capability by evaluating QRM on eight MetaWorld (Yu et al., 2020) manipulation tasks held out from reward model training. QRM is trained on five successful demonstrations from each of 20 other tasks, mixed with RoboReward videos (Lee et al., 2026). QRM (expert) uses expert videos, while QRM (full) uses mixed-quality videos. We compare with LIV (Ma et al., 2023a), VLC (Alakuijala et al., 2025), ReWiND (Zhang et al., 2025), RoboDopamine (Tan et al., 2026), TOPReward (Chen et al., 2026b), RoboMeter (Liang et al., 2026), and RynnValue (Huang et al., 2026), including ReWiND variants matched to QRM's DINOv3 encoder (Siméoni et al., 2026) and training data. For each task, we train policies across 3 seeds with the same RLPD-based learner (Ball et al., 2023), varying only the reward model used to provide the learned reward. Reward models remain frozen during policy learning. Each policy receives 15,000 offline RL updates and 100,000 online environment steps.

Figure 2 compares policy success on tasks excluded from reward-model training. QRM (expert) matches RoboDopamine, the strongest baseline, at 0.853 IQM. QRM (full) exceeds this performance and achieves the highest IQM at 0.927. These results show that QRM provides useful rewards for new tasks without adapting the reward model to those tasks. The comparison with RoboDopamine

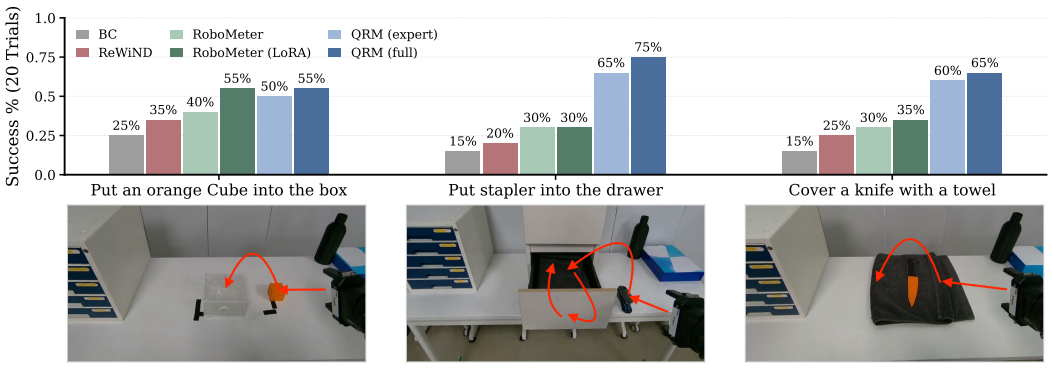


Figure 4: **Fine-tuning a pretrained policy on real-robot tasks.** Success rates over 20 evaluation trials per task. BC denotes the pretrained policy using behavioral cloning before online RL.

also highlights QRM's parameter efficiency. Both QRM variants use approximately $1/50$ as many parameters as RoboDopamine while matching or exceeding its performance. Comparing QRM (expert) with QRM (full) shows the benefit of adding suboptimal video data. For ReWiND, the corresponding comparison is between ReWiND (DINOv3) and ReWiND (full), which share QRM's visual backbone. QRM improves with the additional videos, whereas ReWiND performs worse. This suggests that QRM can use suboptimal videos to improve rewards for unseen tasks beyond what expert-only training provides.

5.2 Q2. Can QRM improve large pretrained robot policies?

For practical use with large pretrained robot policies, a reward model should guide their improvement on downstream tasks. We therefore evaluate QRM with GR00T N1.6-3B (NVIDIA et al., 2025) using DSRL (Wagenmaker et al., 2025) on LIBERO and real-robot manipulation tasks. In both settings, we first pretrain the policy through behavior cloning to obtain an initial success rate of approximately 10–30% before online RL. We compare QRM with DINOv3 adapted ReWiND (Zhang et al., 2025), RoboMeter (Liang et al., 2026), and LoRA finetuned RoboMeter. For each task, all comparisons use the same GR00T policy and DSRL training settings, varying only the reward model used to provide the learned reward. The reward model remains fixed during online learning. Appendix A.5.5 specifies the reward interfaces and training protocols.

LIBERO. We evaluate QRM on eight LIBERO-90 (Liu et al., 2023b) kitchen tasks. QRM and ReWiND are trained on five successful demonstrations from each of the other 82 tasks, mixed with either label-5 RoboReward videos (QRM expert, ReWiND) or all quality labels (QRM full). This study tests reward transfer to tasks excluded from QRM training with a pretrained vision-language-action (VLA) model. We evaluate on 50 initial states per task and report IQM from eight tasks with three random seeds per task, along with 95% confidence intervals.

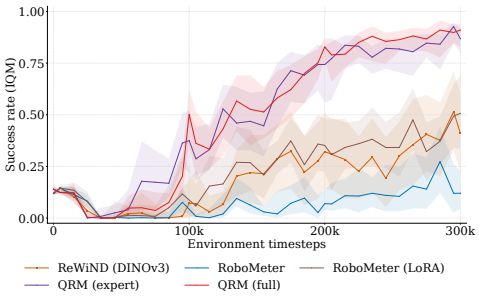
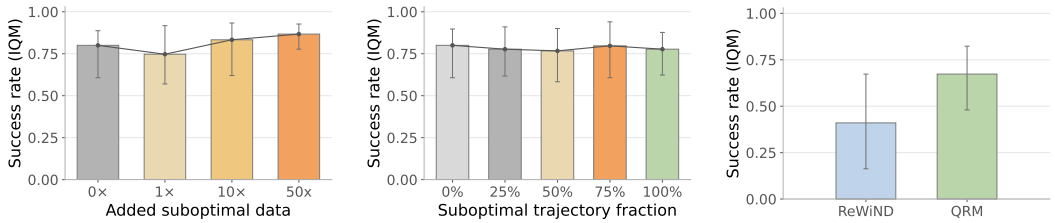


Figure 3: **LIBERO-90 fine-tuning.** Success IQM over eight tasks and three policy seeds, with 95% confidence intervals.

Figure 3 shows that QRM (full) and QRM (expert) achieve comparable performance, with QRM (full) reaching a slightly higher final IQM (0.910 vs. 0.868). Both variants outperform all evaluated baselines in final IQM while the strongest baseline, RoboMeter (LoRA), reaches 0.507. These results suggest that QRM provides useful guidance for adapting a pretrained policy to tasks unseen during reward-model training. The results also suggest that QRM maintains strong performance when suboptimal videos are added to reward-model training, supporting its ability to learn from mixed-quality data.



(a) Scaling with failed trajectories. (b) Optimal/suboptimal composition with fixed budget. (c) Successful but inefficient data.

Figure 5: **Effect of suboptimal training data on downstream policy learning.** We report final success IQM with 95% confidence intervals.

Real-robot manipulation. For real-robot experiments, we design the experiments to model the realistic setting of having few demonstration data. We collect approximately 10 minutes of teleoperation data for each task (30 demonstrations per task) to train the reward models and fine-tune the GR00T policy. We then introduce out-of-distribution settings for the same tasks and run online RL using the learned reward models to improve the policy's performance. We evaluate three tasks: placing an orange cube in a box, placing a stapler in a drawer and closing it and covering a knife with a towel. Resets and success judgments are provided by a human operator. We evaluate each trained policy on 20 separate trials and report success rates.

Figure 4 reports success rates of 55%, 75%, and 65% for QRM (full) on the cube, stapler, and towel tasks, respectively. It matches the strongest baseline on the cube task and improves success by 45 and 30 percentage points on the other two tasks. QRM (full) performs at least as well as QRM (expert) on all three tasks, which supports that adding suboptimal data does not degrade the performance and can improve performance.

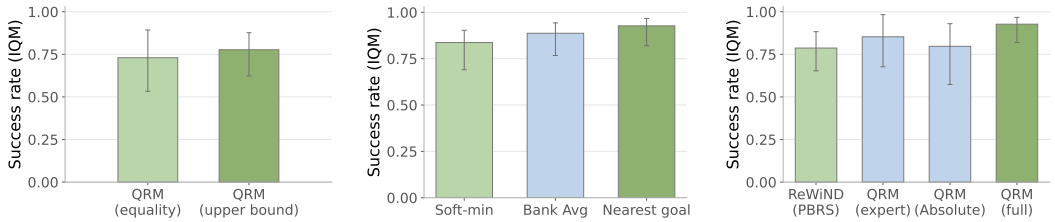
5.3 Q3. Does suboptimal data improve QRM?

Collecting expert demonstrations can be difficult on real robots, motivating reward learning from suboptimal trajectories. We test whether QRM can use inefficient and failed attempts as useful supervision through its path-cost bounds. We use MetaWorld videos with pauses, detours, recoveries, and failures. These studies use no RoboReward data and keep the downstream policy data and training procedure fixed. Appendix A.5.4 describes the trajectory construction.

We first test whether QRM can benefit from unsuccessful attempts without collecting more successful demonstrations. We keep 100 optimal trajectories fixed and add 100, 1,000, or 5,000 failed trajectories. Figure 5a illustrates the benefit of adding large pools of failed trajectories to reward-model training. With 5,000 more failed trajectories, QRM reaches 0.867 IQM compared with 0.800 using optimal demonstrations alone. This suggests that QRM can benefit from scaling reward model training with failed attempts, without collecting additional successful demonstrations. Since the goal bank remains fixed, the improvement comes from learning from the observed segments in failed videos, without adding new successful trajectories.

We next test whether QRM can learn useful rewards when optimal demonstrations are scarce or unavailable. We fix the reward-model training budget at 100 trajectories and vary the suboptimal fraction from 0% to 100% in 25% increments. The suboptimal pool contains successful but inefficient trajectories and failed attempts. Successful endpoints supply the goal bank, and downstream policy data and training remain fixed across conditions. Figure 5b shows that QRM retains 97.1% of the optimal-only IQM even when all optimal demonstrations are replaced. This suggests that QRM can provide useful rewards even without optimal demonstrations for reward-model training. Successful but inefficient trajectories and failed attempts can instead supply the supervision needed.

Finally, we test whether QRM learns more useful rewards than a temporal progress model (ReWiND) when every successful demonstration is inefficient. We train both models on the same 100 successful trajectories containing pauses, detours, or temporary regressions. Figure 5c shows that QRM achieves 0.673 IQM, compared with 0.410 for ReWiND. This gap suggests that path-cost bounds provide useful supervision even when inefficient behavior makes temporal progress a less reliable target.



(a) Upper bound vs. equality. (b) Goal distillation ablation. (c) PBRS ablation.

Figure 6: **Ablation studies.** Bars show final success IQM with 95% confidence intervals. **Left.** Upper bound versus temporal equality supervision. **Middle.** Goal aggregation. **Right.** Reward construction.

5.4 Q4. Which components of QRM are important?

We study how QRM's main design choices affect downstream policy learning. Figure 6 summarizes ablation results for learning distances from observed trajectory costs, using multiple successful outcomes for distillation, and converting predicted costs into rewards. We also test whether QRM's advantage over temporal progress learning holds when both models use the same reward shaping rule.

An inefficient segment can cost more than the shortest path between its endpoints. We first test whether its cost is more useful as an upper bound (UB) or an equality target (TE), using the 100% suboptimal data from the fixed-budget study in Q3, which contains successful but inefficient trajectories and failed attempts. Figure 6a shows that UB yields higher downstream policy success than TE, with 0.777 IQM compared with 0.730. This result supports allowing predicted costs to fall below those of inefficient demonstrations: pauses and detours can loosen a bound without forcing the model to match their additional cost. Appendix A.7 gives the objectives and shared training settings.

We next examine how multiple successful outcomes should define the distillation target by comparing Nearest goal distillation with bank averaging and a soft minimum. Nearest goal uses the minimum predicted distance to the retrieved successful outcomes, selecting the closest outcome. Bank averaging uses the mean of these distances, while the soft minimum provides a smooth approximation to their minimum. Figure 6b shows that Nearest goal achieves the highest success IQM at 0.927, compared with 0.887 for bank averaging and 0.837 for the soft minimum. These results support selecting the closest successful outcome to guide policy, consistent with completing a task by reaching any valid goal. Appendix A.9 defines the aggregation rules and reports confidence intervals.

Finally, we test whether QRM's advantage persists under the same reward shaping. We compare ReWiND (DINOv3) and QRM (expert) trained on the same data, applying PBRS to both models. Figure 6c shows that PBRS improves ReWiND's IQM from 0.723 to 0.787, but it remains below QRM (expert) at 0.853. This result supports the advantage of QRM's learned goal-reaching costs over temporal progress. To evaluate the choice of PBRS, we compare QRM (Absolute) and QRM (full), which use the same frozen model trained on the full dataset. QRM (Absolute) directly uses the normalized negative predicted cost-to-go as its dense reward, while QRM (full) applies PBRS to the same predictions. QRM (full) reaches 0.927 IQM, compared with 0.797 for QRM (Absolute). This result supports potential-based shaping over the direct use of predicted costs as rewards.

6 Conclusion

We introduce QRM, a video reward model that learns from bounds on goal-reaching costs, allowing inefficient and failed trajectories to provide supervision. By distilling the quasimetric distance to the nearest successful training outcome, QRM produces a language-conditioned reward that requires no goal image during policy learning. Experiments on MetaWorld show that QRM benefits from suboptimal videos and transfers to tasks held out from reward-model training. On LIBERO, both expert-only and mixed-quality reward training outperform all evaluated baselines. QRM also improves a pretrained policy on three real-robot manipulation tasks, matching or exceeding the evaluated baselines. These results support using observed paths as bounds for reward learning, making imperfect videos useful beyond imitating the demonstrator's timing.

AI USE STATEMENT

We used generative AI tools to draft sections of the paper, improve writing and presentation, and search for related literature. We also used these tools to support research ideation and execution, including feedback on experimental designs and code for analysis and visualization. AI tools assisted in developing mathematical arguments and proofs. The authors take responsibility for the final manuscript, including all claims, proofs, references, experimental results, and AI-assisted artifacts.

ETHICS STATEMENT

This work studies reward learning for robot manipulation using existing robot video datasets and teleoperated demonstrations. Experiments use simulated environments and a controlled physical robot setup, where a human operator provides resets and success judgments. Learned rewards may reflect biases in the training data or encourage unintended behavior outside the evaluated conditions. Deployment around people therefore requires additional safety validation and appropriate human oversight.

REPRODUCIBILITY STATEMENT

Sections 3 and 4 state the assumptions and learning objectives, with proofs in Appendices A.2 and A.1. Appendix A.3 specifies the architecture, pair sampling, goal-bank construction, optimization, and reward extraction. Appendix A.5 describes the datasets, task splits, baselines, training budgets, random seeds, and evaluation procedures. Training algorithms and per-task learning curves are provided in Appendices A.6 and A.13.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 29304–29320. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/f514cec81cb148559cf475e7426eed5e-Paper.pdf.

Minttu Alakuijala, Reginald McLean, Isaac Woungang, Nariman Farsad, Samuel Kaski, Pekka Marttinen, and Kai Yuan. Video-language critic: Transferable reward functions for language-conditioned robotics. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=jJOVpnNrEp.

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. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf.

Tewodros W Ayalew, Xiao Zhang, Kevin Yuanbo Wu, Tianchong Jiang, Michael Maire, and Matthew R Walter. Progressor: A perceptually guided reward estimator with self-supervised online refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10297–10306, 2025.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 1577–1594. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/ball23a.html.

Chethan Bhateja, Derek Guo, Dibya Ghosh, Anikait Singh, Manan Tomar, Quan Vuong, Yevgen Chebotar, Sergey Levine, and Aviral Kumar. Robotic offline rl from internet videos via value-function learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16977–16984, 2024. doi: 10.1109/ICRA57147.2024.10611575.

Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pre-trained image-editing diffusion models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 33431–33452, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/8ea50bf458f6070548b11babbe0bf89b-Paper-Conference.pdf.

Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 783–792. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/brown19a.html.

Daniel S. Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (eds.), *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 330–359. PMLR, 30 Oct–01 Nov 2020. URL https://proceedings.mlr.press/v100/brown20a.html.

Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Fred Shentu, and Philipp Wu. Sarm: Stage-aware reward modeling for long horizon robot manipulation. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 17091–17116, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/1c92edb990a05f2269f0cc3afbb4c952-Paper-Conference.pdf.

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

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf.

Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/5c48ff18e0a47baaf81d8b8ea51eec92-Paper.pdf.

Dibya Ghosh, Chethan Anand Bhateja, and Sergey Levine. Reinforcement learning from passive data via latent intentions. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11321–11339. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/ghosh23a.html.

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 Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Dongchi Huang, Hongyin Zhang, Bohan Hou, Siteng Huang, Zhian Su, Hang Guo, Tong Lu, Zhaofeng Xu, Jiahao Tang, Jianfei Yang, Donglin Wang, Peixi Peng, Mingxiu Chen, Deli Zhao, and Xin Li. RynnValue: Scaling robotic value foundation models with temporal distance. *arXiv preprint arXiv:2608.09853*, 2026.

Leslie Pack Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence*, 1993. URL https://api.semanticscholar.org/CorpusID:5538688.

Changyeon Kim, Minho Heo, Doohyun Lee, Honglak Lee, Jinwoo Shin, Joseph Lim, and Kimin Lee. Subtask-aware visual reward learning from segmented demonstrations. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 15142–15165, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/277628cff838927d869cd1f671328ce0-Paper-Conference.pdf.

Kimin Lee, Laura M Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6152–6163. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/lee21i.html.

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

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. In *Robotics: Science and Systems*, 2026.

Bo Liu, Yihao Feng, Qiang Liu, and Peter Stone. Metric residual network for sample efficient goal-conditioned reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 8799–8806, 2023a.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023b. doi: 10.52202/075280-1939. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8c3c666820ea055a77726d66fc7d447f-Paper-Datasets_and_Benchmarks.pdf.

Yuyang Liu, Chuan Wen, Yihang Hu, Dinesh Jayaraman, and Yang Gao. Timerewarder: Learning dense reward from passive videos via frame-wise temporal distance. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=XztRm216YS.

Yecheng Jason Ma, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. LIV: Language-image representations and rewards for robotic control. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 23301–23320. PMLR, 2023a.

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. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=YJ7o2wetJ2.

Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, Osbert Bastani, Dinesh Jayaraman, Wenhao Yu, Tingnan Zhang, Dorsa Sadigh, and Fei Xia. Vision language models are in-context value learners. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 33984–34009, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/54854cf15a24fff9f5134a8641136fe4-Paper-Conference.pdf.

Farzan Memarian, Wonjoon Goo, Rudolf Lioutikov, Scott Niekum, and Ufuk Topcu. Self-supervised online reward shaping in sparse-reward environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2369–2375, 2021. doi: 10.1109/IROS51168.2021.9636020.

Vivek Myers, Bill Zheng, Benjamin Eysenbach, and Sergey Levine. Offline goal-conditioned reinforcement learning with quasimetric representations. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 19654–19679. Curran Associates, Inc., 2025. doi: 10.52202/085713-0661. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/1c5956164472d6d8123d574aa75cd063-Paper-Conference.pdf.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pp. 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1558606122.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. In *ArXiv Preprint*, March 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt. Featured Certification.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with Hilbert representations. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 39737–39761. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/park24g.html.

Vitchyr Pong*, Shixiang Gu*, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep RL for model-based control. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Skw0n-W0Z.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

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*, volume 2024, pp. 28446–28463, 2024.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schaul15.html.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seung Eun Yi, Michael Ramamonjisoa, Francisco Massa, Daniel HAZIZA, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal,

Herve Jegou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=2NlGyqNjns. Featured Certification.

Lorenzo Steccanella, Joshua Benjamin Evans, Özgür Şimşek, and Anders Jonsson. Learning the minimum action distance. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ZnOnnJJghs.

Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Cheng Chi, Yuheng Ji, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, Shaoxuan Xie, Guocai Yao, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. General process reward modeling for robotic reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22412–22422, June 2026.

Raymond Tsao, Andrew Wagenmaker, and Sergey Levine. Learning process rewards via success visitation matching for efficient rl. *arXiv preprint arXiv:2606.23640*, 2026.

Andrew Wagenmaker, Yunchu Zhang, Mitsuhiko Nakamoto, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 258–282. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/wagenmaker25a.html.

Tongzhou Wang and Phillip Isola. Improved representation of asymmetrical distances with interval quasimetric embeddings. In *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*, 2022a. URL https://openreview.net/forum?id=KRiST_rzkGl.

Tongzhou Wang and Phillip Isola. On the learning and learnability of quasimetrics. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=y0VvIg25yk.

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 36411–36430. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/wang23al.html.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 5776–5788. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Daniel Yang, Davin Tjia, Jacob Berg, Dima Damen, Pulkit Agrawal, and Abhishek Gupta. Rank2reward: Learning shaped reward functions from passive video. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2806–2813, 2024. doi: 10.1109/ICRA57147.2024.10610873.

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 *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100. PMLR, 2020.

Jiahui Zhang, Yusen Luo, Abrar Anwar, Sumedh Anand Sontakke, Joseph J Lim, Jesse Thomason, Erdem Biyik, and Jesse Zhang. Rewind: Language-guided rewards teach robot policies without new demonstrations. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 460–488. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/zhang25a.html.

Bill Zheng, Vivek Myers, Benjamin Eysenbach, and Sergey Levine. Scaling goal-conditioned reinforcement learning with multistep quasimetric distances. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 146974–146992, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/eddc0fab5a42f8d8de6eb5566cd9f1d3-Paper-Conference.pdf.

A Appendix

A.1 Analysis of Nearest Goal Distillation

We analyze how the finite goal bank and distillation affect QRM's estimate of remaining cost. Fix an instruction z seen during reward model training and a query context $\xi = (o_0, z)$. All states and goal candidates in this approximation analysis are encoded with this context.

A.1.1 Clipped distances

Proposition 1 (Clipping preserves the quasimetric)**.** *For any quasimetric d_θ and $H > 0$, the clipped distance $\bar{d}_\theta(s, g) = \min\{d_\theta(s, g), H\}$ used to form the distillation target in Section 4.2 is also a quasimetric.*

Proof. Let $\bar{d}_\theta(x, y) = \min\{d_\theta(x, y), H\}$. Nonnegativity and $d_\theta(x, x) = 0$ are preserved immediately. The triangle inequality follows from

$$\begin{aligned} \bar{d}_\theta(x, z) &\le \min\{d_\theta(x, y) + d_\theta(y, z), H\} \\ &\le \bar{d}_\theta(x, y) + \bar{d}_\theta(y, z). \end{aligned} \tag{11}$$

Thus, clipping preserves the quasimetric properties. □

A.1.2 Finite goal bank error

Using Eq. (8), write $J^\star_\theta(s, z) = J_\theta(s, \mathcal{G}_z(\xi))$ for the score over all valid goals in this context and $J_{\text{bank}}(s, z) = J_\theta(s, \widehat{\mathcal{G}}_z(\xi))$ for the retrieved-bank score.

Assumption 1 (Valid goal bank and nearest outcome)**.** *The retrieved bank $\widehat{\mathcal{G}}_z(\xi)$ is finite and nonempty, and its candidates are successful outcomes valid in the query's setting, so $\widehat{\mathcal{G}}_z(\xi) \subseteq \mathcal{G}_z(\xi)$. For every query state s with context ξ, some $g^\star_s \in \mathcal{G}_z(\xi)$ attains the ideal score $J^\star_\theta(s, z) = \bar{d}_\theta(s, g^\star_s)$.*

Retrieval provides a nonempty bank whenever matching candidates are available. We assume that retrieval finds valid goals in this analysis.

Theorem 1 (Finite goal bank error)**.** *Under Assumption 1, define the bank coverage error as $\epsilon_{\text{bank}}(s, z) = \min_{g_n \in \widehat{\mathcal{G}}_z(\xi)} \bar{d}_\theta(g^\star_s, g_n)$. Then the nearest goal score satisfies*

$$0 \le J_{\text{bank}}(s, z) - J^\star_\theta(s, z) \le \epsilon_{\text{bank}}(s, z). \tag{12}$$

Proof of Theorem 1. Under Assumption 1, every bank goal is valid. The bank is a subset of the set used by the ideal infimum, so $J_{\text{bank}}(s, z) \ge J^\star_\theta(s, z)$. This proves the lower bound.

Let $\widetilde{g}_s \in \arg\min_{g_n \in \widehat{\mathcal{G}}_z(\xi)} \bar{d}_\theta(g^\star_s, g_n)$. The triangle inequality gives

$$\begin{aligned} J_{\text{bank}}(s, z) &\le \bar{d}_\theta(s, \widetilde{g}_s) \\ &\le \bar{d}_\theta(s, g^\star_s) + \bar{d}_\theta(g^\star_s, \widetilde{g}_s) \\ &= J^\star_\theta(s, z) + \epsilon_{\text{bank}}(s, z). \end{aligned} \tag{13}$$

Subtracting $J^\star_\theta(s, z)$ proves the upper bound. □

This bound links score accuracy to coverage of successful outcomes. For a fixed distance model and query context, adding valid outcomes to the evaluated set cannot increase the bank error.

The coverage term is at most H and equals H when $d_\theta(g^\star_s, g_n) \ge H$ for every retrieved goal. Thus, a bank can contain valid outcomes while giving a loose bound if its goals are far from the nearest outcome in the directed distance.

A.1.3 ERROR DECOMPOSITION

We next separate the remaining approximation errors. Let the ideal environment score be $J^{\star}_{\mathrm{env},H}(s,z) = \inf_{g\in\mathcal{G}_z(\xi)} \min\{d^{\star}_{\mathrm{env}}(s,g), H\}$. Define the geometry error as $\epsilon_{\mathrm{geom}}(s,z) = |J^{\star}_{\theta}(s,z) - J^{\star}_{\mathrm{env},H}(s,z)|$ and the distillation error as $\epsilon_{\mathrm{distill}}(s,z) = |J_{\omega}(s,z) - J_{\mathrm{bank}}(s,z)|$. Together with bank coverage, define $\epsilon_{\mathrm{total}}(s,z) = \epsilon_{\mathrm{geom}}(s,z) + \epsilon_{\mathrm{bank}}(s,z) + \epsilon_{\mathrm{distill}}(s,z)$. Under the conditions of Theorem 1,

$$|J_{\omega}(s,z) - J^{\star}_{\mathrm{env},H}(s,z)| \le \epsilon_{\mathrm{total}}(s,z). \tag{14}$$

Proof of Eq. (14). Adding and subtracting $J^{\star}_{\theta}(s,z)$ and $J_{\mathrm{bank}}(s,z)$ gives

$$\begin{aligned}\left|J_{\omega}(s,z) - J^{\star}_{\mathrm{env},H}(s,z)\right| \le{}& |J_{\omega}(s,z) - J_{\mathrm{bank}}(s,z)| \\ &+ |J_{\mathrm{bank}}(s,z) - J^{\star}_{\theta}(s,z)| \\ &+ \left|J^{\star}_{\theta}(s,z) - J^{\star}_{\mathrm{env},H}(s,z)\right|.\end{aligned} \tag{15}$$

The first and third terms are the distillation and geometry errors. By Theorem 1, the middle term is at most $\epsilon_{\mathrm{bank}}(s,z)$. This proves Eq. (14). □

Accurate distances, good coverage of successful outcomes, and faithful distillation together can yield an accurate estimate of remaining cost. Whether training achieves this accuracy requires empirical evaluation.

A.1.4 POLICY INVARIANCE UNDER POTENTIAL BASED SHAPING

The following result applies standard potential based shaping (Ng et al., 1999) independently of the approximation analysis.

Corollary 1 (Policy invariance under potential based shaping)**.** *For any instruction z, fix a start state s and a potential Φ. Consider finite episodes with discount $0 < \gamma \le 1$ and zero potential at every true terminal state. Let V^{π}_{Φ} be the expected discounted return using Φ in Eq. (10), and let V^{π}_{base} use the same sparse success reward and discount without shaping. Then every policy π satisfies*

$$V^{\pi}_{\Phi}(s,z) = V^{\pi}_{\mathrm{base}}(s,z) - \alpha\Phi(s,z). \tag{16}$$

Consequently, for any two policies π and π',

$$V^{\pi}_{\Phi} - V^{\pi'}_{\Phi} = V^{\pi}_{\mathrm{base}} - V^{\pi'}_{\mathrm{base}}, \tag{17}$$

where all returns are evaluated at (s,z). The shaped and sparse rewards therefore rank policies in the same order and have the same optimal policies.

Proof of Corollary 1. For every trajectory ending after T transitions, the discounted shaping terms cancel across time,

$$\sum_{t=0}^{T-1} \gamma^t \left[\gamma\Phi(s_{t+1},z) - \Phi(s_t,z)\right] = -\Phi(s_0,z) + \gamma^T\Phi(s_T,z) = -\Phi(s_0,z), \tag{18}$$

where the last equality uses the zero terminal potential. Taking expectations gives Eq. (16). For fixed (s,z), the shaped return differs from the sparse return by a constant independent of the policy. Subtracting the returns of two policies proves Eq. (17). □

For continuing trajectories with $0 < \gamma < 1$ and bounded Φ, the same identity follows because $\gamma^T\Phi(s_T,z) \to 0$. A time-limit truncation does not end the underlying task. Retaining its next-state potential, as in Eq. (10), preserves the identity when the learner includes the continuation value rather than treating the truncation as a true terminal.

This result requires no assumptions on distance accuracy, goal-bank coverage, or distillation. It applies to seen and unseen instructions under the same terminal conditions.

We evaluate whether QRM improves learning within a fixed interaction budget in Section 5.

A.2 Proofs for Section 3

A.2.1 Goal-set cost and distance

Proof of Eq. (2). Under deterministic dynamics every rollout of a policy is a feasible trajectory. If it first reaches $\mathcal{G}_z$ at a state g, its cost up to that step is at least $d^\star_{\text{env}}(s, g)$, so $J^\pi(s, \mathcal{G}_z) \geq \inf_{g \in \mathcal{G}_z} d^\star_{\text{env}}(s, g)$ for every π. Conversely, take any $g \in \mathcal{G}_z$ and any feasible trajectory from s to g. Removing its loops does not increase its cost, and a policy that follows the resulting path reaches $\mathcal{G}_z$ no later than at g, at no larger cost because costs are nonnegative. Taking infima over trajectories and over g gives the identity without requiring a minimizing path or goal to exist. If no goal is reachable, both sides are infinite. □

A.2.2 Temporal regression as policy evaluation

Proposition 2 (Temporal regression evaluates the demonstrator)**.** *Under the assumptions of Section 3.1, let demonstrations of z be generated by a stationary Markov policy π_β that reaches $\mathcal{G}_z$ with probability one, each ending when it first reaches $\mathcal{G}_z$, at step T. Assume finite expected episode length and $\mathbb{E}[\sum_{t=0}^{T-1} C(\tau_{t:T})^2] < \infty$.*

(i) Over unrestricted functions $f(s, z)$, the minimizer of $\mathbb{E}_{\tau \sim \pi_\beta}[\sum_{t=0}^{T-1} (f(s_t, z) - C(\tau_{t:T}))^2]$ equals the demonstrator's cost-to-go almost everywhere on visited states, $f(s, z) = J^{\pi_\beta}(s, \mathcal{G}_z)$. With unit costs the regression target is the remaining time $C(\tau_{t:T}) = T - t$, giving Eq. (3).

(ii) $J^{\pi_\beta}(s, \mathcal{G}_z) \geq J^\star(s, \mathcal{G}_z)$, with equality at a visited state s if and only if the demonstrator follows a minimum-cost path from s to $\mathcal{G}_z$ with probability one.

Proof. (i) The squared loss decomposes over visited states, and for each state s it is minimized by the conditional mean of the target, $f(s, z) = \mathbb{E}_{\pi_\beta}[C(\tau_{t:T}) \mid s_t = s]$. Fix a time t. Because π_β and the dynamics are Markov, the law of the suffix $\tau_{t:T}$ given $s_t = s$ is that of a fresh rollout of π_β from s, whatever happened before t. Since π_β reaches $\mathcal{G}_z$ with probability one and costs accumulate without discounting until $\mathcal{G}_z$ is first reached, $\mathbb{E}_{\pi_\beta}[C(\tau_{t:T}) \mid s_t = s] = J^{\pi_\beta}(s, \mathcal{G}_z)$ for every t, and hence also when averaged over the times at which s is visited.

(ii) Every demonstrated suffix from s is a feasible trajectory from s to its endpoint $s_T \in \mathcal{G}_z$, so Eq. (4) and Eq. (2) give $C(\tau_{t:T}) \geq d^\star_{\text{env}}(s, s_T) \geq J^\star(s, \mathcal{G}_z)$ for every sample. Taking the conditional expectation and using part (i) yields $J^{\pi_\beta}(s, \mathcal{G}_z) \geq J^\star(s, \mathcal{G}_z)$. A random variable bounded below by a constant has mean equal to that constant if and only if it equals the constant with probability one, that is, if and only if almost every demonstrated suffix from s attains the minimum cost of reaching $\mathcal{G}_z$. □

If π_β depends on history, the minimizer in (i) is still the conditional mean $\mathbb{E}[C(\tau_{t:T}) \mid s_t = s]$, now an average of the remaining cost of the demonstrator over the histories that lead to s, and the lower bound still holds because it applies to every suffix. For normalized progress, the minimizer is instead $\mathbb{E}[t/T \mid s_t = s]$. It also reflects demonstration timing, but when T varies it is not a fixed rescaling of J^{π_β}.

A.2.3 Observed segments bound the optimal distance

Proof of Eq. (4). Every observed segment $\tau_{i:j}$ belongs to $\mathfrak{T}(s_i, s_j)$. Its cost is therefore at least the infimum over this set, so $d^\star_{\text{env}}(s_i, s_j) \leq C(\tau_{i:j})$. Equality holds exactly when the segment has minimum cost between its endpoints. This argument does not require the trajectory to succeed or the generating policy to be optimal. □

A.2.4 Shortest-path costs form a quasimetric

Proof of Eq. (5). The empty trajectory belongs to $\mathfrak{T}(s, s)$ and has zero cost. Since all transition costs are nonnegative, no trajectory can have negative cost. Therefore $d^\star_{\text{env}}(s, s) = 0$.

We next prove the triangle inequality. If either $d^\star_{\text{env}}(s, u)$ or $d^\star_{\text{env}}(u, g)$ is infinite, the inequality holds immediately. Otherwise, for any $\epsilon > 0$, the definition of the infimum gives trajectories $\tau_1 \in \mathfrak{T}(s, u)$

and $\tau_2 \in \mathfrak{T}(u, g)$ such that

$$C(\tau_1) \le d^\star_{\text{env}}(s, u) + \frac{\epsilon}{2}, \qquad C(\tau_2) \le d^\star_{\text{env}}(u, g) + \frac{\epsilon}{2}. \tag{19}$$

Joining τ_1 and τ_2 gives a feasible trajectory from s to g. Hence

$$d^\star_{\text{env}}(s, g) \le C(\tau_1) + C(\tau_2) \le d^\star_{\text{env}}(s, u) + d^\star_{\text{env}}(u, g) + \epsilon. \tag{20}$$

Because this holds for every positive ϵ, the triangle inequality follows. The argument does not imply symmetry, and directed dynamics can have different costs in the two directions. □

A.2.5 Combining segment bounds and adding data

Proof of Eq. (6). Let $q_{\mathcal{D}}(s, g)$ be the infimum of the total cost of all finite chains of observed segments that connect s to g. The empty chain gives $q_{\mathcal{D}}(s, s) = 0$, and the value is $+\infty$ if no chain exists. We allow this infinite value in the unrestricted quasimetric class.

All segment costs are nonnegative. Concatenating chains proves the triangle inequality by the same infimum argument as in Appendix A.2.4, so $q_{\mathcal{D}}$ is a quasimetric. Each observed segment is itself a chain, so $q_{\mathcal{D}}$ satisfies its upper bound. Conversely, any quasimetric d satisfying all segment bounds obeys, along every chain $s = x_0, \dots, x_m = g$ with segment costs C_k,

$$d(s, g) \le \sum_{k=1}^{m} d(x_{k-1}, x_k) \le \sum_{k=1}^{m} C_k.$$

Taking the infimum over chains gives $d(s, g) \le q_{\mathcal{D}}(s, g)$. Thus $q_{\mathcal{D}}$ is the largest consistent quasimetric $\hat{d}_{\mathcal{D}}$.

Every chain is a feasible environmental path, so $d^\star_{\text{env}}(s, g) \le q_{\mathcal{D}}(s, g)$. Adding trajectories adds possible chains and cannot increase their infimum. For $\mathcal{D} \subseteq \mathcal{D}'$, this gives

$$d^\star_{\text{env}}(s, g) \le q_{\mathcal{D}'}(s, g) \le q_{\mathcal{D}}(s, g),$$

which is Eq. (6). If segments in $\mathcal{D}'$ form an environmental shortest path, that chain also gives $q_{\mathcal{D}'}(s, g) \le d^\star_{\text{env}}(s, g)$, establishing equality. □

This result concerns exact segment constraints and the unrestricted class described in Section 3.3. The sampled, relaxed training objective is given in Eq. (7).

A.3 Practical implementation

We describe the QRM architecture and training procedure used in MetaWorld, LIBERO, and on the real robot. Environment-specific datasets and policy learning settings are given in Appendix A.5.

A.3.1 Contextual state representation

Inspired by the temporal architecture of ReWiND (Zhang et al., 2025), we encode images with frozen DINOv3 ViT-B/16 features (768 dimensions) (Siméoni et al., 2026) and instructions with frozen MiniLM-L12 features (384 dimensions) (Wang et al., 2020; Reimers & Gurevych, 2019). Linear projections map both modalities to width 512. A causal Transformer processes the instruction, initial observation, and up to 16 recent observations, producing $s_t = E_\psi(o_{\max(0,t-15):t}, \xi)$ with context $\xi = (o_0, z)$. It has four layers, eight attention heads, feed-forward width 2,048, ReLU activations, and dropout 0.1.

A two-block residual projector with LayerNorm maps s_t to 2,048 coordinates. The IQE-maxmean distance head (Wang & Isola, 2022a) uses 64 components of dimension 32. The scalar cost-to-go model is a three-block residual MLP of width 512 with output $J_\omega = H\,\mathrm{sigmoid}(f_\omega)$, where $H = 48$. Distances share the same task context. The visual and language backbones remain frozen throughout training.

We include the task information to the state embedding as task context determines which visual differences matter. For example, under "close the drawer," observations differing only in a stapler's position can be equally far from completion. Under "place the stapler in the drawer," that difference matters. The encoder can therefore suppress task-irrelevant variation. Segment costs remain task-independent, and the quasimetric enforces triangle inequalities for every fixed context.

A.3.2 STAGE 1 SAMPLING AND OPTIMIZATION

Each update samples target-environment trajectories with probability 0.2 and RoboReward trajectories with probability 0.8, uniformly within each domain. MetaWorld-only studies sample their selected corpus. Two batches of 128 pairs share the sampled trajectories. For the local batch B_c, we sample a valid horizon $h \in \{1, 2, 4, 8\}$ and start index uniformly. For the push batch B_p, we independently sample two distinct indices over the full trajectory and sort them into forward order. Let d_i^c and d_j^p denote the corresponding distances, and C_i the observed local path cost. We define

$$v_{B_c} = \frac{1}{|B_c|} \sum_{i \in B_c} [d_i^c - C_i]_+^2, \qquad r_{B_c} = v_{B_c} - 0.25^2, \tag{21}$$

and optimize

$$\mathcal{L}_{B_c, B_p} = -\frac{1}{|B_p|} \sum_{j \in B_p} \log(1 + d_j^p) + \lambda r_{B_c} + \frac{1}{2}[r_{B_c}]_+^2. \tag{22}$$

The shared multiplier is initialized at $\lambda = 1$ and updated by

$$\lambda \leftarrow \max\{0, \lambda + 0.01 r_{B_c}\}. \tag{23}$$

Stage 1 distances are not clipped. H bounds only the distillation target.

We use AdamW with geometry-head learning rate 3×10^{-4}, Transformer learning rate 10^{-5}, weight decay 10^{-4}, and gradient-norm limit 1. Training consists of 1,000 geometry-head warmup updates, 10,000 adaptation updates, and 1,000 polishing updates. Adaptation also updates the final Transformer block. Appendix A.4 evaluates the within-trajectory push sampler.

A.3.3 GOAL BANK CONSTRUCTION

The bank contains successful endpoints from the target environment and RoboReward label-5 trajectories. label 1–4 RoboReward trajectories contribute training pairs but do not supply goals. For each query, we retrieve up to 16 endpoints with the same normalized task identity, ranked by cosine similarity between initial-image features. No additional camera, scene, or embodiment filters are applied.

We encode each candidate's recent observation history under the query's context $\xi = (o_0, z)$ as

$$g_n = E_\psi\left(o^{(n)}_{\max(0, T_n - 15):T_n}, \xi\right). \tag{24}$$

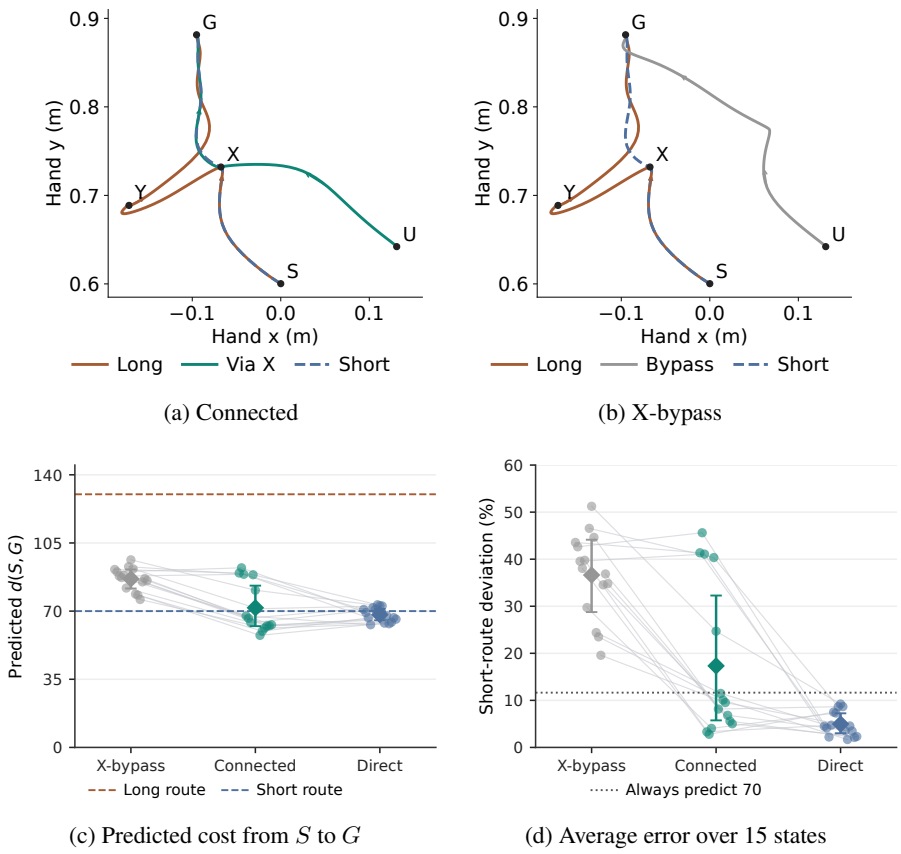
The frozen distance model uses the minimum directed distance to these candidates, clipped at $H = 48$, as $J_{\text{bank}}(s, z)$. Queries without a matching candidate are excluded from distillation but may train Stage 1. Interpreting cross-trajectory distances as reaching costs assumes that candidate histories represent compatible outcomes in the query's setting. Appendix A.8 reports coverage and retrieval examples.

A.3.4 STAGE 2 SUPERVISION

We retain the same domain mixture and sample query trajectories with matching goals uniformly within each domain, then sample query times uniformly within each trajectory. Let $\mathcal{Q}_B$ contain queries with matching goals and $\mathcal{S}_B$ the simulator-verified successful queries in the minibatch. With the encoder and distance model frozen, we train only the cost-to-go model using

$$\begin{aligned}\mathcal{L}_{\text{distill},B} = {} & \text{mean}_{i \in \mathcal{Q}_B}\left[\text{Huber}_1(J_\omega(s_i, z_i) - \text{sg}[J_{\text{bank}}(s_i, z_i)])\right] \\ & + \text{mean}_{i \in \mathcal{S}_B} J_\omega(s_i, z_i)^2.\end{aligned} \tag{25}$$

Empty averages contribute zero. RoboReward endpoints supply goals but do not receive the zero-boundary penalty. AdamW uses 10,000 updates, batch size 128, learning rate 3×10^{-4} with cosine decay, weight decay 10^{-4}, and gradient-norm limit 1.



(a) Connected

(b) X-bypass

(c) Predicted cost from S to G

(d) Average error over 15 states

Figure 7: **Learning a shorter route from inefficient trajectories.** Connected training paths (a), X-bypass training paths (b), start-to-goal predictions (c), and average short-route errors (d). Small points show individual models, lines connect results for the same scene and training seed, diamonds show means, and error bars show 95% bootstrap intervals.

A.3.5 Reward extraction

For MetaWorld, we apply potential-based shaping (Ng et al., 1999) with potential $\Phi_t = -J_\omega(s_t, z)/48$, discount $\gamma = 0.99$, and shaping coefficient $\alpha = 1$, giving

$$r_t = 200\,\mathbf{1}[\text{success}] + \alpha\left[\gamma\widetilde{\Phi}_{t+1} - \Phi_t\right]. \tag{26}$$

The absolute-QRM uses

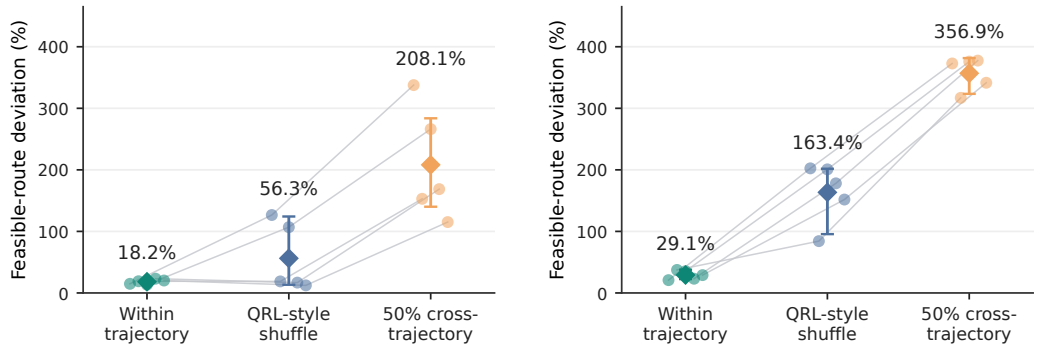
$$r_t^{\text{absolute}} = r_t^{\text{success}} + \widetilde{\Phi}_{t+1} \tag{27}$$

The DINOv3 ReWiND PBRS comparison substitutes $\Phi_t = f_{\text{ReWiND}}(s_{\le t}, z) - 1$ with the same shaping scale, and success bonus. Chunk-level rewards for GR00T policy learning are specified in Appendix A.5.5.

A.4 Justification of within-trajectory sampling

The maximization term in Eq. (7) pushes sampled states and encourages larger distances between them. QRL samples these states independently (Wang et al., 2023), whereas we sample forward pairs within a trajectory, where the recorded path provides a cost bound. We test whether our sampler can learn shorter routes by combining segments from different trajectories and compare it with samplers that also push pairs across trajectories.

Experimental setup. We use five scenes of the MetaWorld (Yu et al., 2020) reaching task `reach-v2`, with three training seeds per scene. Figure 7 shows the robot hand's paths from



(a) Connected trajectories

(b) Disconnected trajectories

Figure 8: **Comparison of samplers.** Circles show scene means over three seeds, lines connect the same scene, diamonds show overall means, and error bars show 95% intervals. Deviation measures relative error in the remaining cost of the short scripted route. Lower values are better. Both panels use the same scale.

starting positions S and U toward goal G, with a meeting point X and a detour through Y. We measure route cost by the number of actions. The long route $S \to X \to Y \to G$ takes 130 actions, while a scripted controller designed to approximate a shortest path follows $S \to X \to G$ in 70 actions. We evaluate predictions against this shorter route's recorded cost.

All conditions train on the long route and use within-trajectory sampler with the Stage 1 settings in Appendix A.3.2. Figure 7a shows the *connected* condition, which adds $U \to X \to G$. The model can combine S-to-X from one trajectory with X-to-G from the other. Figure 7b shows the *X-bypass* condition, which replaces the added trajectory that does not pass through X, keeping its start, goal, and action count fixed. This tests whether meeting near X helps the model learn the shorter route. For comparison, the *direct* condition trains on both the 130-action long route $S \to X \to Y \to G$ and the complete 70-action short route $S \to X \to G$. This condition provides a reference in which the complete short route is included in training, while connected and X-bypass training exclude it.

Distance predictions. Figure 7c compares predicted costs from S to G. The dashed *Short route* and *Long route* lines mark the recorded costs of 70 and 130 actions. Predictions closer to 70 better match the shorter route.

Figure 7d evaluates the remaining cost to goal g from the first 15 states of the long trajectory. For each state s, we compare the predicted distance with the remaining action count along the short route, $C_{\text{short}}(s, g)$, using relative error

$$100 \, \frac{|d_\theta(s, g) - C_{\text{short}}(s, g)|}{C_{\text{short}}(s, g)}.$$

The plotted *Short-route deviation* averages these errors, with lower values indicating better predictions. The dotted *Always predict 70* line assigns 70 to every state. It incurs error because the remaining cost decreases as the hand approaches the goal. All conditions use the same queries, goal observations, and context. We compute 95% bootstrap intervals by resampling scenes and seeds, matched across conditions.

Connected training predicts 71.7 from S to G, close to the short-route cost of 70, compared with 86.6 for X-bypass and 68.2 for direct supervision. Relative error falls from 36.6% for X-bypass to 17.3% for connected training, a reduction of 19.3 percentage points with a 95% interval of $[7.9, 30.5]$. Direct supervision reaches 5.0% error. On short training segments, predicted distances remain about 0.84 times the recorded cost in both connected and X-bypass conditions, suggesting that the improvement is not simply due to smaller predictions everywhere. These results support learning a shorter route by combining observed segments without seeing the complete short trajectory.

Comparing samplers. We next compare samplers using trajectories $S \to X \to Y$ and $U \to X \to G$ in the connected condition. Here the first trajectory ends at Y, so no training trajectory contains both S and G. The disconnected condition replaces the second trajectory with a path around X. We use the same five scenes, three seeds, and relative-error measure over the first 15 states, with the 70-action short route used only for evaluation.

Within-trajectory sampling uses forward pairs from one trajectory. Mixed sampling replaces half of the later states with states from another trajectory. QRL-style sampling shuffles recorded transitions, pairing the start of one with the end of another, without requiring the same trajectory or forward time order. Only the sampler changes. The model, objective, local cost constraints, batch size, and training budget are shared, and the evaluated state–goal pairs are excluded from sampling.

Figure 8 shows mean relative errors of 18.2%, 56.3%, and 208.1% for within-trajectory, QRL-style, and mixed sampling on connected data. On disconnected data, they are 29.1%, 163.4%, and 356.9%. Within-trajectory sampling has the lowest mean relative error in both conditions.

Distance at the meeting point. To understand the sampler results, we measure the learned distance between the two observations near X in the connected condition. Although the hand positions differ by less than 1 mm, the observations come from different trajectories. We measure from the observation on $S \to X \to Y$ to the one on $U \to X \to G$, using the first trajectory's initial image and the shared instruction. Each observation retains its own history.

Table 1: **Learned distance at the meeting point.** Raw Stage 1 distances averaged over five scenes and three seeds, with 95% bootstrap intervals.

Sampler	Mean distance	95% CI
Within-trajectory	1.31	[0.92, 1.67]
QRL-style shuffle	172.04	[139.84, 211.80]
50% cross-trajectory	189.59	[145.67, 236.89]

Within-trajectory sampling keeps this distance small, whereas both samplers allowing cross-trajectory push assign much larger distances (Table 1). The triangle inequality combines the two segments' distances with this additional distance at X to bound the distance from S to G. A smaller distance at X gives a tighter bound when the segment distances are fixed. This helps explain why within-trajectory sampling better estimates the shorter route.

Together, these results support within-trajectory push under the fixed training budget. It keeps distance maximization tied to observed path-cost bounds while allowing generalization across training trajectories.

A.5 Experimental setup

A.5.1 MetaWorld data and baselines

The main reward-training dataset contains 100 successful MetaWorld (Yu et al., 2020) trajectories from 20 tasks, with five trajectories per task. QRM (full) adds 54,099 RoboReward (Lee et al., 2026) trajectories with labels 1 through 5, while QRM (expert) adds the 9,924 label-5 trajectories. RoboReward is used only for reward-model training, including its provided splits. The MetaWorld and RoboReward sampling probabilities are 0.2 and 0.8, respectively, as specified in Appendix A.3.2. These probabilities describe minibatch sampling rather than dataset proportions.

The eight MetaWorld evaluation tasks are button-press-wall, door-lock, faucet-close, handle-press-side, window-close, reach-wall, coffee-button, and sweep-into. They are held out from reward-model training, and a separate policy is trained for each task. The reward model remains frozen during downstream learning. The MetaWorld-only data studies use the same training and evaluation task split.

We compare against ReWiND (Zhang et al., 2025), LIV (Ma et al., 2023a), VLC (Alakuijala et al., 2025), RoboDopamine (Tan et al., 2026), TOPReward (Chen et al., 2026b), RoboMeter (Liang et al., 2026), and RynnValue (Huang et al., 2026). Original ReWiND uses DINOv2 (Oquab et al., 2024). The DINOv3 variant matches QRM's visual backbone, and ReWiND (full) additionally uses all

RoboReward labels. RoboMeter (LoRA) is fine-tuned on the same MetaWorld and RoboReward sources. The remaining external models use their released checkpoints. The separate DINOv3 ReWiND PBRS comparison uses the reward conversion in Appendix A.3.5.

A.5.2 METAWORLD POLICY LEARNING

Policies use visual observations and proprioception. We perform 15,000 offline updates before online learning with an RLPD-based learner (Ball et al., 2023) and advantage-weighted policy extraction. Policy and critic MLPs have widths 768, 512, and 256, without LayerNorm. The critic ensemble contains five networks, with two sampled for target computation. Critic targets omit the entropy term. Online replay batches contain equal proportions of offline and online transitions. The downstream offline dataset is held fixed across reward-model data studies. Table 2 gives the shared training settings.

Table 2: Shared MetaWorld policy-learning settings.

Setting	Value
Offline updates	15,000
Online environment steps	100,000
Learning rate	10^{-4}
Discount	0.99
Action chunk length	10
Online replay fraction	0.5
Offline critic-update ratio	1
Online critic-update ratio	4
Online gradient steps	1
Online warm-start steps	4,000
Episode horizon	128
Added action noise	0

A.5.3 EVALUATION AND UNCERTAINTY

QRM uses reward-training seed 0, and the DINOv3 ReWiND PBRS comparison uses reward-training seed 42. Each frozen reward model is evaluated with policy seeds 42, 43, and 44 on the eight held-out tasks, giving 24 task-policy outcomes per condition. Each evaluation consists of 25 episodes with a deterministic policy. Final success is measured at 100,000 online environment steps. Learning curves begin after offline training.

Following Agarwal et al. (2021), we report the IQM over the 24 task-policy success rates, averaging the middle 50% of values. The final mean is the arithmetic average over all 24 outcomes. The 95% confidence intervals use 50,000 task-stratified bootstrap draws. Tasks remain fixed, and the three policy seeds are resampled with replacement independently within each task. We recompute the IQM for each draw and use the 2.5th and 97.5th percentiles as interval endpoints. These intervals describe policy-seed variation conditional on the selected tasks and trained reward model.

A.5.4 SUBOPTIMAL TRAJECTORY CONSTRUCTION

All data studies in Section 5.3 use only MetaWorld. The optimal corpus contains 100 scripted trajectories, five per training task. Successful but inefficient trajectories perturb scripted control with pauses after initial movement, strong action-noise bursts, or excursions away from the goal, followed by recovery to simulator-reported success. Accepted trajectories finish later than the corresponding unperturbed execution.

Failed trajectories use corrupted scripted control or exploratory control. They are filtered to contain no simulator success at any recorded step, and their endpoints do not enter the goal bank. The mixed suboptimal pool contains 36% successful but inefficient trajectories, 32% failed corrupted-control trajectories, and 32% exploratory trajectories.

Scaling failed trajectories. The scaling study keeps the same 100 optimal trajectories and adds 0, 100, 1,000, or 5,000 failed trajectories, corresponding to $0\times$, $1\times$, $10\times$, and $50\times$ the optimal dataset

Table 3: **SAC settings for noise-space policy learning.** The discount is applied per executed control step.

Setting	LIBERO	RB-Y1
Actor learning rate	10^{-4}	10^{-4}
Critic / temperature learning rate	3×10^{-4}	3×10^{-4}
Minibatch size	256	256
Critics / target reduction	10 / mean	10 / mean
Target-network update coefficient	0.005	0.005
Target entropy	-64	-64
Control-step discount	0.999	0.999
Updates per collected chunk	20	20 (cap 2,000 per episode)
Learning threshold	500 chunks	64 chunks
Policy seeds	0, 1, 2	0
QRM shaping coefficient α	1	100
Episode horizon	400 steps	640 steps (task 1 and 3) / 800 steps (task 2)
Training budget	300,000 environment steps	10,000 environment steps

size. Each added pool contains equal numbers of failed corrupted-control and failed exploratory trajectories. Successful but inefficient trajectories are excluded from these additions, so the goal bank continues to use only the fixed optimal corpus. Table 12 reports the final results.

Fixed-budget composition. We keep 100 trajectories and replace 0%, 25%, 50%, 75%, or 100% of the optimal demonstrations with trajectories from the mixed suboptimal pool. Successful endpoints in each mixture supply its goal bank. At 100% suboptimal data, these come from successful but inefficient trajectories. The budget fixes trajectory count rather than frame count. Table 13 reports the final results.

Successful but inefficient data. QRM and DINOv3 ReWiND each train on the same 100 successful but inefficient trajectories, five per training task. No ordinary optimal trajectories or RoboReward videos are included. QRM uses their successful endpoints as its goal bank. Table 14 reports the final results.

All QRM data studies use the shared training settings in Appendix A.3, with each corpus supplying its own training pairs and successful goal candidates. The downstream policy's offline data remain fixed.

Figures 16, 17, and 18 show the corresponding learning curves.

A.5.5 Pretrained policy learning

We use DSRL (Wagenmaker et al., 2025) with GR00T N1.6-3B (NVIDIA et al., 2025) in LIBERO and on the RB-Y1. A tanh-Gaussian soft actor-critic (SAC) actor (Haarnoja et al., 2018) selects a 128-dimensional latent vector, repeated to initialize the frozen flow-matching action head. GR00T decodes a 16-step action chunk using four flow-integration steps. The SAC learner is updated online while GR00T and the reward model remain frozen.

The SAC input is a 64×64 camera image and an eight-dimensional proprioceptive state. Actor and critic networks use four convolutional layers with 32 channels, a 50-dimensional visual bottleneck, and a three-layer MLP of width 128. LIBERO uses group normalization and spatial softmax, while the robot uses a plain CNN. Table 3 summarizes shared hyperparameters and environment-specific budgets.

Reward construction. For a chunk of $L_k \leq 16$ executed actions, QRM supplies

$$r_k = 64\,\mathbf{1}[\text{success}] + \alpha\left(\gamma^{L_k}\widetilde{\Phi}_{k+1} - \Phi_k\right), \qquad \Phi_k = -J_\omega(s_{t_k}, z)/48, \tag{28}$$

where $\gamma = 0.999$, with $\alpha = 1$ in LIBERO and $\alpha = 100$ on the robot. The critic uses the same chunk discount γ^{L_k}. Success zeroes the next-state potential and disables bootstrapping. Truncations retain both. The success signal comes from simulator state in LIBERO and a human operator on the robot.

Table 4: **LIBERO tasks excluded from QRM training.** These tasks remain available in the separate policy-adaptation corpus.

Scene	Instruction
Kitchen 1	Put the black bowl on the plate
Kitchen 1	Put the black bowl on top of the cabinet
Kitchen 2	Open the top drawer of the cabinet
Kitchen 2	Put the black bowl at the front on the plate
Kitchen 2	Stack the black bowl at the front on the black bowl in the middle
Kitchen 3	Put the frying pan on the stove
Kitchen 3	Put the moka pot on the stove
Kitchen 3	Turn on the stove

A.5.6 LIBERO EXPERIMENTS

Tasks and reward training. We train QRM on 410 successful demonstrations from 82 LIBERO-90 tasks (Liu et al., 2023b), with five demonstrations per task, and reserve the eight tasks in Table 4 for downstream learning. The RoboReward subsets are the same as in Appendix A.5.1. LIBERO and RoboReward trajectories are sampled with probabilities 0.2 and 0.8. The architecture and two-stage training follow Appendix A.3, with LIBERO replacing MetaWorld in the training mixture and successful goal bank.

Policy and training. The GR00T prior is adapted on 3,921 LIBERO-90 demonstrations for 5,000 behavior-cloning updates. It observes external and wrist-camera images, the instruction, and end-effector position, axis-angle orientation, and two gripper joint positions, and predicts seven-dimensional robot actions. Behavior cloning updates the final four language-model blocks and action modules using AdamW with learning rate 10^{-4}, batch size 64, and weight decay 10^{-5}. The cosine learning-rate schedule spans 20,000 updates with 1,000 warmup updates.

Online replay starts empty. Collection uses Gaussian latent noise until the learning threshold, then samples the SAC actor. Each task is trained for 300,000 environment steps with three policy seeds. Episodes terminate on success or after 400 steps. Evaluation uses all 50 official initial states, also used during training, and the simulator's binary success predicate. We report IQM over the 24 task–seed success rates and 95% intervals from 50,000 bootstrap draws that resample seeds within each fixed task. These intervals measure policy-seed variability for the trained reward models and evaluation tasks. QRM uses Eq. (28) with $\alpha = 1$.

A.5.7 REAL-ROBOT EXPERIMENTS



(a) RB-Y1 setup. (b) Cube in box. (c) Stapler in drawer. (d) Cover knife.

Figure 9: **Real-robot experimental setting.** The RB-Y1 uses its right arm and gripper with a head-mounted RGB camera. Tasks require placing an orange cube in a box, placing a stapler in a drawer and closing it, and covering a knife with a towel.

Robot platform. We use the Rainbow Robotics RB-Y1[1], a wheeled dual-arm mobile manipulator (Figure 9). The robot has two seven-joint arms with a six-joint torso that connects the arms to a two-wheel differential-drive base, and a two-joint head. All experiments use the right arm and its

[1] https://rainbow-robotics.com/en/products/rb-y1/

gripper, with the base, torso, head, and left arm held fixed. Visual observations come from an Intel RealSense D405 mounted on the head bracket.

Tasks and data collection. The three tasks are placing an orange cube in a box, placing a stapler in a drawer and closing it, and covering a knife with a towel. We collect 30 teleoperated demonstrations per task, about 10 minutes of data per task. The policy observes the current camera image, instruction, seven joint positions, and gripper state. Each action contains seven joint-position targets and a gripper target. We execute 16-action chunks open loop at 20 Hz.

Policy adaptation. A separate GR00T policy is adapted per task for 20,000 behavior-cloning updates, training the projector and action head while freezing the language model and vision encoder. We use learning rate 10^{-4}, batch size 32, weight decay 10^{-5}, and a 5% warmup fraction.

Reward models. All models use head-camera observations and instructions. QRM and ReWiND (Zhang et al., 2025) are trained on the 90 collected demonstrations along with their respective RoboReward subsets. We also evaluated the pretrained RoboMeter (Liang et al., 2026) and its LoRA variant adapted to the same 90 demonstrations. LoRA uses rank 32, scaling 64, and dropout 0.05, with the progress, preference, and success heads also trained. Adaptation uses 1,000 updates, batch size 32, learning rate 2×10^{-5}, weight decay 0.01, and a 10% warmup fraction. ReWiND uses 16 subsampled history frames, while RoboMeter and its LoRA variant use at most eight. Their reward for each action chunk is the predicted progress after the robot finishes the chunk and settles, All methods receive an additional bonus of 64 once at success. QRM uses Eq. (28) with $\alpha = 100$.

Online learning and evaluation. Each condition uses one policy seed and a nominal budget of 10,000 steps (625 action chunks). Updates start after 1,024 steps and run after each episode, with settings in Table 3. Images are augmented with random crops and color jitter. The operator assigns success or failure. Trials interrupted by robot hardware faults are excluded. We evaluate each trained policy for 20 trials and report success rates with Wilson 95% intervals in Figure 4. These intervals measure evaluation-trial uncertainty for a single trained policy.

A.5.8 Parameter efficiency

Table 5 reports model sizes for Q1. We count all reward-model parameters used at inference, including frozen encoders, and exclude the policy and components used only during training. Both QRM variants have 171M parameters, including the DINOv3 and MiniLM encoders, contextual state encoder, and distilled cost-to-go model.

Table 5: **Reward-model size and downstream performance in Q1.** Counts include frozen encoders. M and B denote millions and billions. Success IQMs are from Figure 2, with confidence intervals in Table 9.

Reward model	Parameters	Success IQM
RoboDopamine (Tan et al., 2026)	8.77B	0.853
TOPReward (Chen et al., 2026b)	8.77B	0.750
RoboMeter (Liang et al., 2026)	4.45B	0.583
RoboMeter (LoRA)	4.52B	0.653
QRM (expert)	171M	0.853
QRM (full)	171M	**0.927**

RoboMeter and RoboMeter (LoRA) have approximately 26.0 and 26.4 times as many parameters as QRM, while RoboDopamine and TOPReward have 51.3 times as many. QRM (expert) matches the strongest baseline's final IQM, while QRM (full) achieves the highest IQM. LoRA reduces the number of parameters updated during fine-tuning but retains the full backbone at inference.

A.6 TRAINING ALGORITHMS

A.6.1 QRM TRAINING

Algorithm 1 summarizes the two training stages. Sampling and optimization settings are given in Appendix A.3.

Algorithm 1 Quasimetric Reward Model Training

Require: trajectory dataset $\mathcal{D}$, state encoder E_ψ, quasimetric d_θ, cost-to-go model J_ω, distance cap H

// Stage 1. Learn the quasimetric
Initialize the Lagrange multiplier λ
for each distance-learning update **do**
  Sample local forward segments B_c and independent forward pairs B_p across the full length of the same trajectories
  Update E_ψ and d_θ with distance maximization on B_p and the upper-bound penalty on B_c using Eq. (22)
  Update λ using Eq. (23)
end for
// Stage 2. Distill the nearest goal distance
Build the candidate pool $\mathcal{B}$ from endpoints of successful training trajectories
Freeze the state encoder and quasimetric
for each distillation update **do**
  Sample query states and instructions (s, z) from $\mathcal{D}$
  Retrieve task-matching goals from $\mathcal{B}$ and encode them under each query's context to form $\widehat{\mathcal{G}}_z(\xi)$
  For queries with candidates, set $J_{\text{bank}}(s, z) \leftarrow \min_{g \in \widehat{\mathcal{G}}_z(\xi)} \bar{d}_\theta(s, g)$
  Train J_ω to match these targets and anchor sampled successful endpoints from the target environment at zero using Eq. (25)
end for
return cost-to-go model J_ω for scoring without a goal image

A.6.2 POLICY LEARNING

Algorithm 2 applies the frozen cost-to-go model during policy learning. Environment-specific success signals, bonus values, and reward conversions are given in Appendix A.3.5.

Algorithm 2 Policy Learning with QRM Reward

Require: policy π, frozen cost-to-go model J_ω, instruction z, sparse success reward r_t^{success}

Define $\Phi_\omega(s, z) \leftarrow -J_\omega(s, z)/H$
for each environment transition (s_t, a_t, s_{t+1}) **do**
  Set $\widetilde{\Phi}_{t+1} \leftarrow 0$ at a true terminal state, and $\Phi_\omega(s_{t+1}, z)$ otherwise
  Compute $r_t \leftarrow r_t^{\text{success}} + \alpha[\gamma\widetilde{\Phi}_{t+1} - \Phi_\omega(s_t, z)]$
  Update π using the shaped reward r_t
end for

A.7 UPPER BOUND AND TEMPORAL EQUALITY SUPERVISION

We train both reward models from scratch using the 100% suboptimal data from the fixed-budget study in Section 5.3. This corpus contains 100 MetaWorld trajectories from 20 training tasks, drawn from the mixed suboptimal pool of successful but inefficient trajectories and failed attempts (Appendix A.5.4). Both conditions share the data split and goal bank. The goal bank contains only the endpoints of the successful but inefficient trajectories, since failed endpoints do not enter the goal bank. No optimal trajectories or RoboReward data are included. The downstream offline policy dataset remains fixed.

Only the Stage 1 constraint statistic changes. On local batch B_c, let d_i^c denote each model's predicted distance and C_i the observed segment cost. UB and TE respectively use

$$v_{B_c}^{\mathrm{UB}} = \frac{1}{|B_c|} \sum_{i \in B_c} [d_i^c - C_i]_+^2, \qquad v_{B_c}^{\mathrm{TE}} = \frac{1}{|B_c|} \sum_{i \in B_c} (d_i^c - C_i)^2. \tag{29}$$

For either choice, we set $r_{B_c} = v_{B_c} - 0.25^2$ and use the same loss and dual update in Eqs. (22) and (23). Both arms maximize distances on the same independently sampled full-trajectory push pairs B_p. Equality supervision applies only to the local pairs and is not extended to B_p. Architecture, optimization, and training budgets follow Appendix A.3. Initial weights, training pairs, and Stage 2 queries and retrieved candidates are matched. Each cost-to-go model uses its own frozen Stage 1 distance model for nearest goal distillation, so the selected nearest goal may differ with the learned geometry.

Both conditions use the shared policy and evaluation settings in Appendices A.5.2 and A.5.3. UB reaches 0.777 IQM with 95% CI $[0.623, 0.877]$, while TE reaches 0.730 with $[0.533, 0.893]$.

A.8 Goal bank coverage and retrieval

Table 6 summarizes goal availability in the MetaWorld and RoboReward training data used by QRM (full). Coverage is the fraction of trajectories with at least one same-task candidate. Stage 2 samples from these trajectories within each domain, while trajectories without a matching goal remain available for Stage 1. Among eligible RoboReward trajectories, the median number of retrieved candidates is 12, and 92.8% have a candidate from another video.

Table 6: Goal-bank coverage and candidate availability. Candidate statistics use only Stage 2 trajectories with a matching goal, with at most 16 candidates retrieved per trajectory.

	MetaWorld	RoboReward
All training trajectories		
Trajectories	100	54,099
With a matching goal	100.0%	52.9%
Stage 2 trajectories with a matching goal		
Trajectories	100	28,617
Median retrieved candidates	5	12
With a candidate from another video	100.0%	92.8%

Figure 10 shows qualitative results of goal retrival where RoboReward queries are evaluated at each trajectory's midpoint. For label 1–4 trajectories, the original endpoints do not provide successful goals. Retrieval instead supplies label-5 endpoints from other videos that visually match the instructed outcomes in these examples. This illustrates how the goal bank can provide proxy goals even for unsuccessful trajectories.

A.9 Goal aggregation ablation

All aggregation variants use the full RoboReward training condition and the same frozen quasimetric. The cost-to-go model architecture and training settings are shared. Candidate retrieval follows Appendix A.3.3. Let $M \leq 16$ be the number of retrieved candidates for a query, and let $d_j = \bar{d}_\theta(s, g_j)$ be their clipped distances. We compare

$$\begin{aligned} y_{\mathrm{nearest}} &= \min_{1 \leq j \leq M} d_j, \\ y_{\mathrm{mean}} &= \frac{1}{M} \sum_{j=1}^{M} d_j, \\ y_{\mathrm{soft}} &= -\tau \log \left[\frac{1}{M} \sum_{j=1}^{M} \exp(-d_j/\tau) \right], \qquad \tau = 0.5. \end{aligned} \tag{30}$$

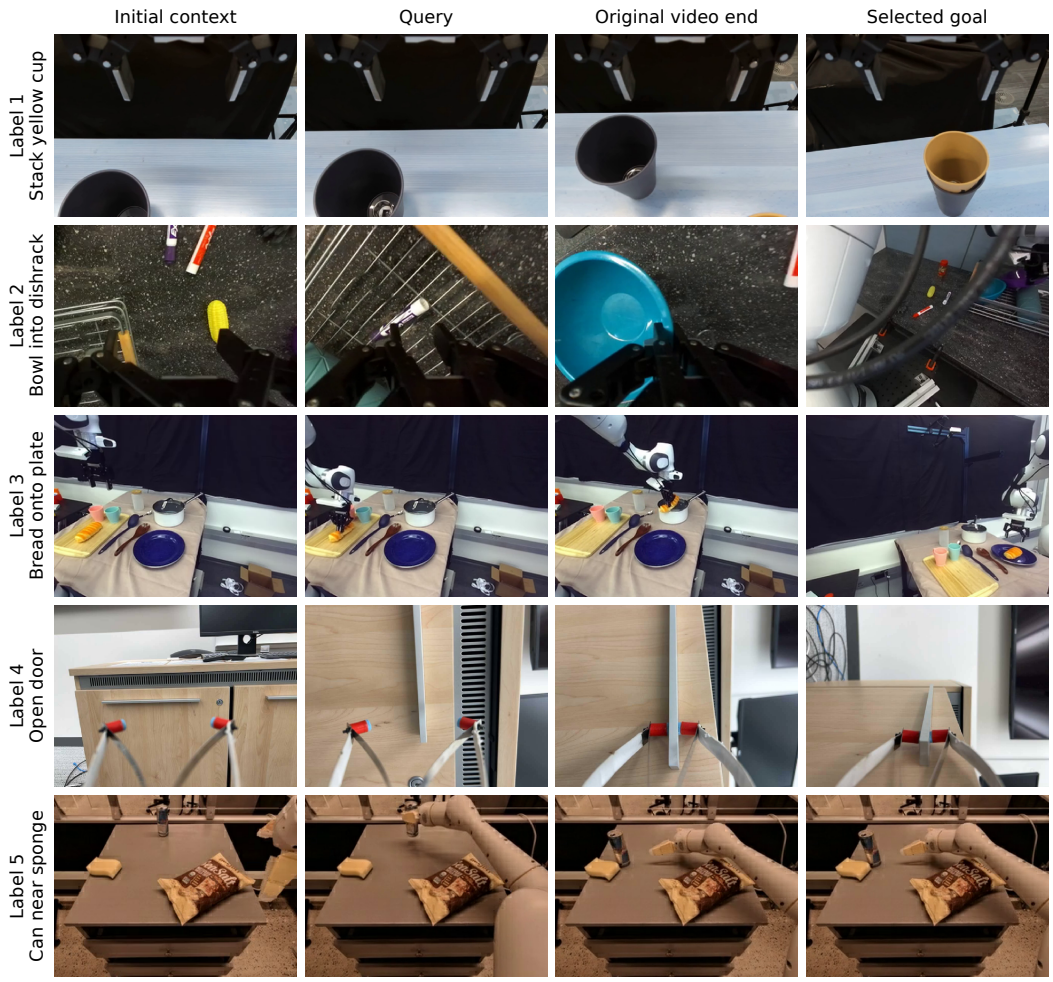


Figure 10: **Qualitative goal-bank retrieval.** Columns show the initial observation, current query, original video endpoint, and selected goal. All goal candidates are label-5 endpoints. The selected goal minimizes the frozen quasimetric distance among retrieved candidates.

Nearest goal distillation selects the closest candidate separately for each query. Bank averaging (Bank Avg) uses every retrieved candidate. The soft minimum (Soft-min) includes the normalization by M. All variants assign confidence 1 to MetaWorld and RoboReward candidates.

Table 7 reports final success IQM and 95% confidence intervals. Nearest goal achieves the highest IQM under the shared training settings. Figure 20 shows the learning curves.

Table 7: **Goal aggregation ablation.** Success IQM after 100,000 environment steps over eight MetaWorld tasks and three policy seeds, with 95% confidence intervals.

Aggregation	IQM	95% CI
Soft-min	0.837	[0.690, 0.903]
Bank Avg	0.887	[0.767, 0.943]
Nearest goal	**0.927**	[0.820, 0.967]

A.10 Reward construction ablation

The reward ablation uses the conversions in Appendix A.3.5 and the common downstream training settings in Appendix A.5.2. DINOv3 ReWiND and QRM (expert) are trained on the same data and both use PBRS. QRM (Absolute) and QRM (full) share the same frozen cost-to-go model trained on the full dataset. QRM (Absolute) directly uses the normalized negative cost, while QRM (full) uses PBRS. All conditions receive the same terminal sparse success bonus.

Table 8 reports the final results. With the same training data and PBRS rule, QRM (expert) reaches 0.853 IQM compared with ReWiND's 0.787, supporting QRM's reward-learning approach. For the same full-data QRM model, replacing absolute rewards with PBRS raises IQM from 0.797 to 0.927, supporting our choice of reward construction. Figure 21 shows the learning curves.

Table 8: **Reward construction ablation.** Success IQM after 100,000 environment steps over eight MetaWorld tasks and three policy seeds, with 95% confidence intervals. QRM (expert) and QRM (full) use PBRS.

Reward	IQM	95% CI
ReWiND (PBRS)	0.787	[0.653, 0.883]
QRM (expert)	0.853	[0.677, 0.983]
QRM (Absolute)	0.797	[0.573, 0.930]
QRM (full)	**0.927**	[0.820, 0.967]

A.11 Distance architecture ablation

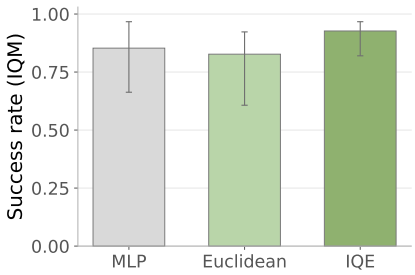


Figure 11: **Distance architecture ablation.** Final success IQM over eight MetaWorld tasks and three policy seeds, with 95% confidence intervals.

We compare IQE (Wang & Isola, 2022a), Euclidean distance, and an unconstrained MLP using the full RoboReward training condition and nearest goal distillation. The contextual encoder, 2,048-dimensional projector, training pairs, objective, goal bank, and optimization budget are shared. The

Euclidean head computes the unsquared L2 distance between projected states. The MLP concatenates the projected source and goal representations and applies a depth-two residual MLP with hidden dimension 512 and a softplus output. It enforces nonnegativity but imposes no zero diagonal, symmetry, or triangle inequality.

IQE reaches an IQM of 0.927 ($[0.820, 0.967]$), the MLP reaches 0.853 ($[0.663, 0.967]$), and Euclidean distance reaches 0.827 ($[0.607, 0.923]$) (Figure 11). IQE achieves the highest IQM among the three architectures. Euclidean distance imposes symmetry, restricting its ability to represent different forward and reverse costs. The MLP allows asymmetry but does not enforce the triangle inequality used to combine segment bounds. These results suggest that IQE benefits from retaining quasimetric structure without imposing symmetry.

Following QRL (Wang et al., 2023), we use IQE to also accommodate tasks with different forward and reverse costs. Figure 12 gives a simple example in a maze with a one-way passage. The upper passage permits movement only from right to left, so states on the left must reach the goal on the right through the lower corridor. This illustrates why allowing asymmetric distances can be useful.

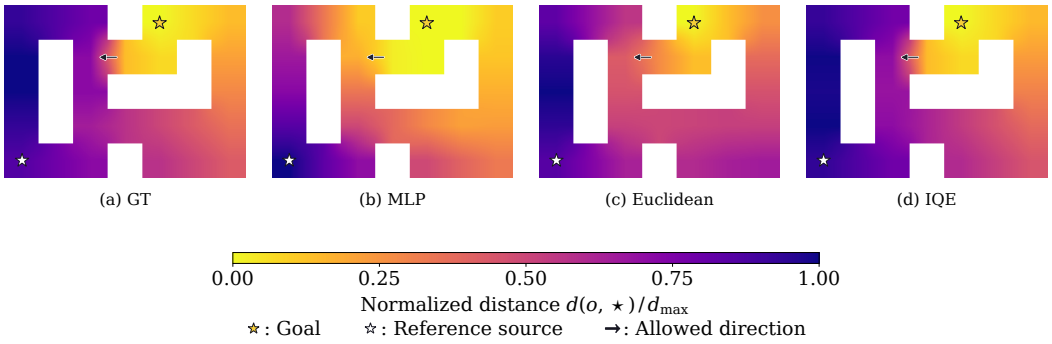


Figure 12: **Distance predictions in an asymmetric toy environment.** Panels show ground truth, MLP, Euclidean distance, and IQE. Colors indicate normalized distance to the goal, marked by the yellow star. The white star marks a reference source, and the arrow marks the allowed direction through the upper passage.

A.12 Qualitative distance estimates

Figure 14 shows qualitative comparison for all eight held-out MetaWorld tasks.

A.13 Individual results for RL

A.13.1 Q1 experiments

Table 9 reports final success IQMs and 95% confidence intervals after 100,000 online environment steps, aggregated over eight held-out MetaWorld tasks and three policy seeds per task. Figure 15 shows per-task learning curves for the MetaWorld reward-model comparison in Section 5.1.

A.13.2 Q2 experiments

Table 10 reports model-level LIBERO IQMs and 95% confidence intervals over eight tasks and three policy seeds per task. Figure 22 shows LIBERO learning curves under the protocol in Appendix A.5.6.

On LIBERO, QRM (full) achieves the highest IQM of 91.0%, compared with 86.8% for QRM (expert) and 50.7% for RoboMeter (LoRA), the strongest baseline. The relative performance of the two QRM variants varies across tasks.

Table 11 and Figure 4 report real-robot success after online RL alongside the pretrained policy via behavioral cloning. Relative to BC, QRM (full) improves success by 30, 60, and 50 percentage points on the cube, stapler, and towel tasks, respectively. Its gains over QRM (expert) are 5, 10, and 5

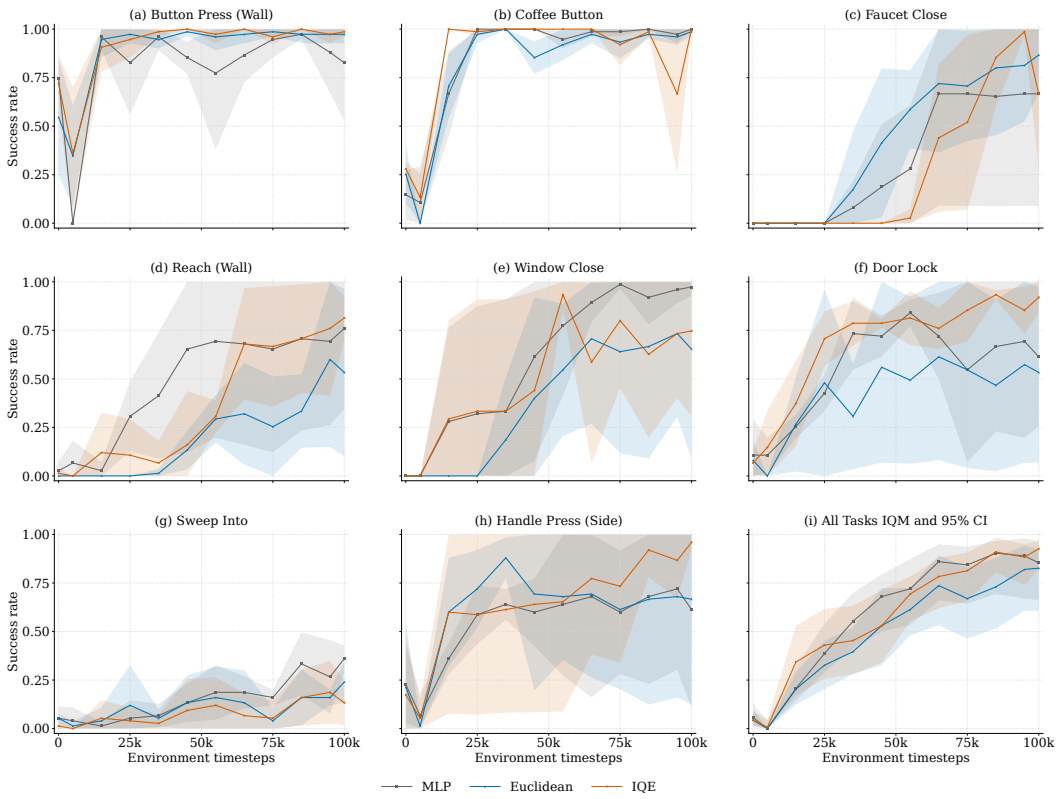


Figure 13: **Per-task success curves for the distance-architecture ablation.** Learning curves corresponding to Figure 11.

Table 9: **Final MetaWorld results for Q1.** Success IQMs and 95% confidence intervals for the comparison in Figure 2.

Method	Success IQM	95% CI
LIV	0.210	[0.000, 0.513]
VLC	0.440	[0.173, 0.737]
ReWiND	0.610	[0.347, 0.877]
ReWiND (DINOv3)	0.723	[0.493, 0.877]
ReWiND (full)	0.587	[0.333, 0.803]
RoboDopamine	0.853	[0.667, 0.957]
TOPReward	0.750	[0.593, 0.873]
RoboMeter	0.583	[0.347, 0.813]
RoboMeter (LoRA)	0.653	[0.440, 0.837]
RynnValue	0.767	[0.617, 0.887]
QRM (expert)	0.853	[0.677, 0.983]
QRM (full)	**0.927**	[0.820, 0.967]

percentage points. Averaged across tasks, QRM (full) reaches 65.0% success, compared with 58.3% for QRM (expert) and 40.0% for RoboMeter (LoRA), the strongest baseline.

A.13.3 Q3 EXPERIMENTS

Tables 12, 13, and 14 report final success for the failed-data scaling, fixed-budget, and successful but inefficient data studies in Figure 5. The following figures show per-task learning curves for the three Q3 studies.

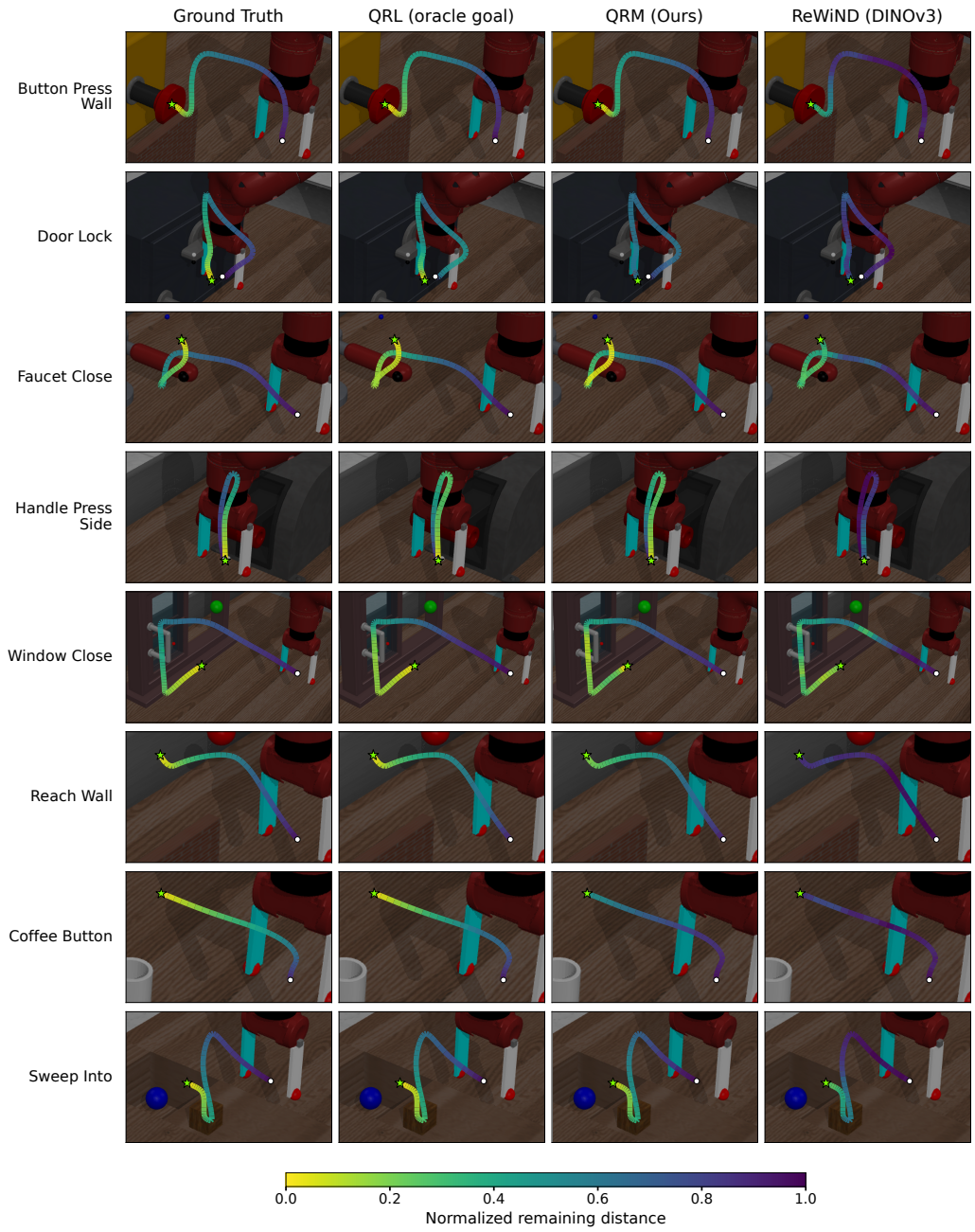


Figure 14: **Qualitative remaining distance estimates across eight MetaWorld tasks.** Each row shows one task. Columns compare ground truth, the goal-conditioned distance model with an oracle goal image, QRM, and ReWiND (DINOv3). All shown trajectories are generated by scripted policy which is designed to be near-optimal. With unit step costs, ground truth is the normalized remaining time $(T - t)/T$, where t is the current step and T is the completion step. Colors indicate normalized remaining distance. White circles mark initial states, and green stars mark successful outcomes.

Table 10: **LIBERO success after 300,000 environment steps.** Success IQMs and 95% bootstrap intervals over eight tasks and three policy seeds. Bold marks the highest IQM.

Method	**Success IQM**	**95% CI**
ReWiND (DINOv3)	0.410	[0.223, 0.607]
RoboMeter	0.120	[0.043, 0.228]
RoboMeter (LoRA)	0.507	[0.335, 0.692]
QRM (expert)	0.868	[0.798, 0.917]
QRM (full)	**0.910**	[0.828, 0.938]

Table 11: **Real-robot manipulation on the RB-Y1.** Task entries show success rates with Wilson 95% intervals from 20 evaluation trials per task. Total is the mean success rate across the three tasks. BC is the behavior-cloned GR00T policy before online RL. Each RL condition uses one policy seed. Bold marks the highest success rate in each column.

Policy / reward	Cube in box	Stapler in drawer	Cover knife	Total
BC	0.25 [0.112, 0.469]	0.15 [0.052, 0.360]	0.15 [0.052, 0.360]	0.183
ReWiND	0.35 [0.181, 0.567]	0.20 [0.081, 0.416]	0.25 [0.112, 0.469]	0.267
RoboMeter	0.40 [0.219, 0.613]	0.30 [0.145, 0.519]	0.30 [0.145, 0.519]	0.333
RoboMeter (LoRA)	**0.55** [0.342, 0.742]	0.30 [0.145, 0.519]	0.35 [0.181, 0.567]	0.400
QRM (expert)	0.50 [0.299, 0.701]	0.65 [0.433, 0.819]	0.60 [0.387, 0.781]	0.583
QRM (full)	**0.55** [0.342, 0.742]	**0.75** [0.531, 0.888]	**0.65** [0.433, 0.819]	**0.650**

Table 12: **Scaling with failed trajectories.** Success IQM after 100,000 environment steps over eight MetaWorld tasks and three policy seeds, with 95% confidence intervals. Added failures are measured relative to the fixed set of 100 optimal trajectories. The successful demonstrations and goal bank remain unchanged.

Added failures	IQM	95% CI
0×	0.800	[0.607, 0.887]
1×	0.747	[0.570, 0.917]
10×	0.833	[0.620, 0.933]
50×	0.867	[0.777, 0.927]

Table 13: **Suboptimal data under a fixed trajectory budget.** Success IQM after 100,000 environment steps over eight MetaWorld tasks and three policy seeds, with 95% confidence intervals. Each reward model is trained on 100 trajectories.

Suboptimal fraction	IQM	95% CI
0%	0.800	[0.607, 0.897]
25%	0.777	[0.617, 0.910]
50%	0.767	[0.583, 0.900]
75%	0.797	[0.607, 0.940]
100%	0.777	[0.623, 0.877]

Table 14: **Successful but inefficient data.** Success IQM after 100,000 environment steps over eight MetaWorld tasks and three policy seeds, with 95% confidence intervals. Both reward models are trained on the same 100 successful but inefficient trajectories, with no optimal trajectories.

Reward model	IQM	95% CI
ReWiND (DINOv3)	0.410	[0.163, 0.673]
QRM	0.673	[0.480, 0.823]

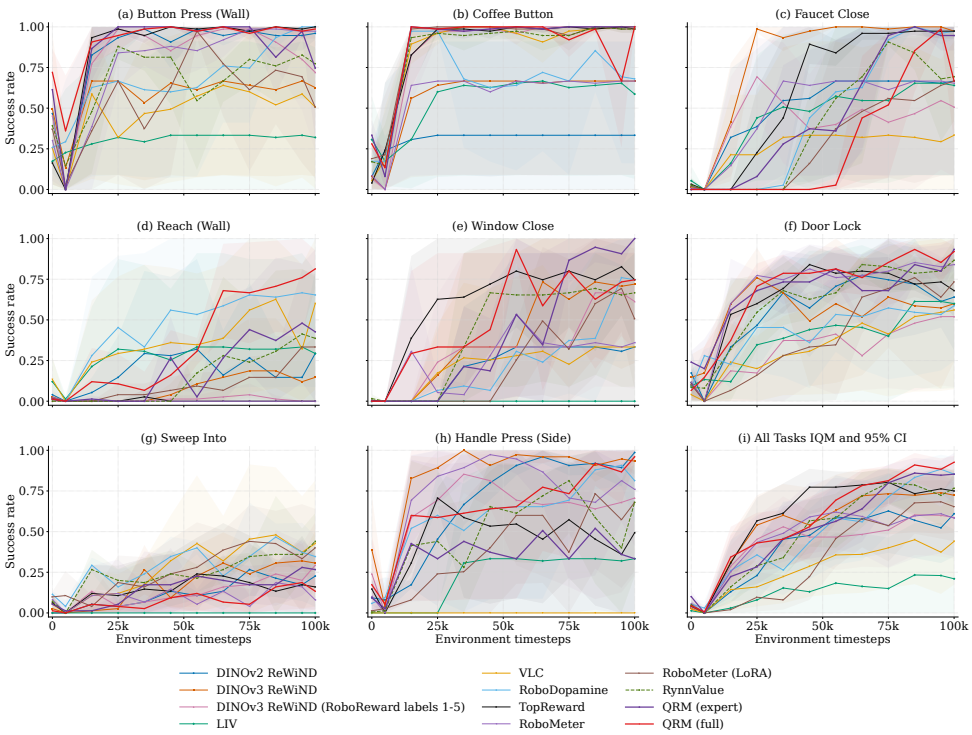


Figure 15: **Per-task success curves for Q1.** Success rates over online environment steps for each held-out MetaWorld task. The final panel reports the aggregate IQM with 95% confidence intervals.

A.13.4 Q4 EXPERIMENTS

Figures 19, 20, and 21 show learning curves for the supervision, goal aggregation, and reward shaping ablations.

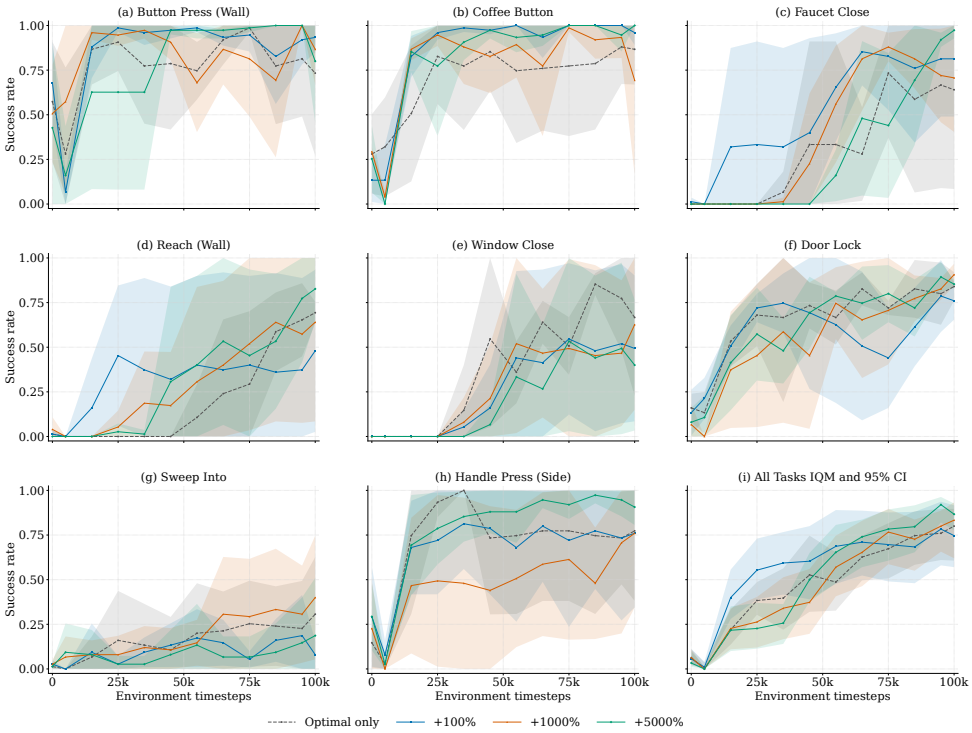


Figure 16: **Per-task success curves for failed-trajectory scaling.** Learning curves for the study in Figure 5a, adding 0, 100, 1,000, or 5,000 failed trajectories to 100 fixed optimal trajectories.

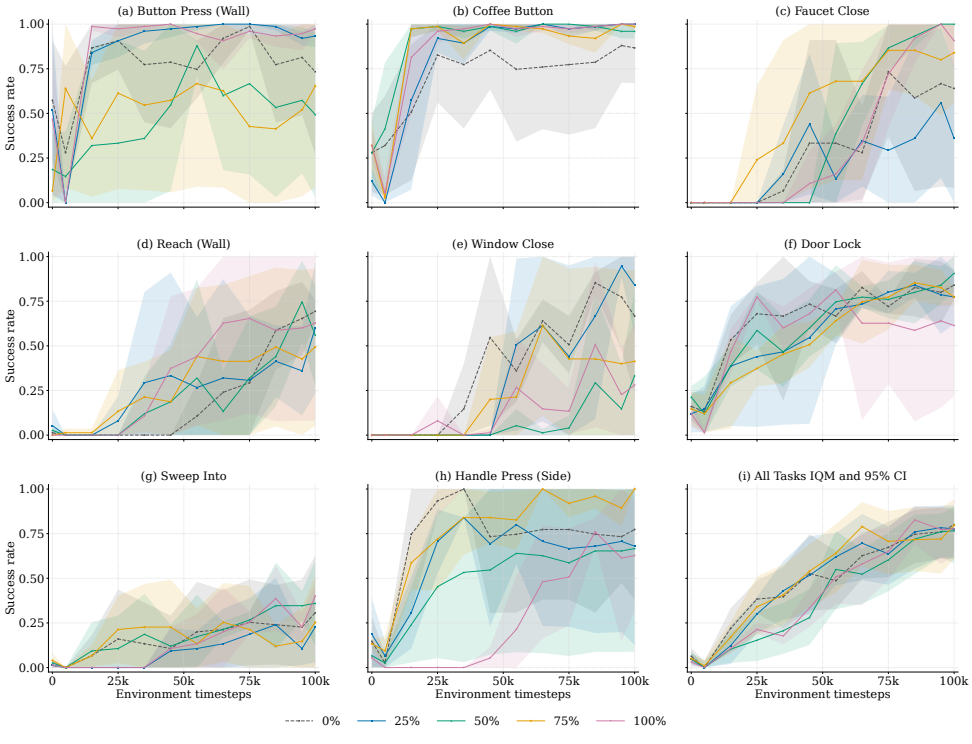


Figure 17: **Per-task success curves for the fixed-budget composition study.** Learning curves corresponding to Figure 5b.

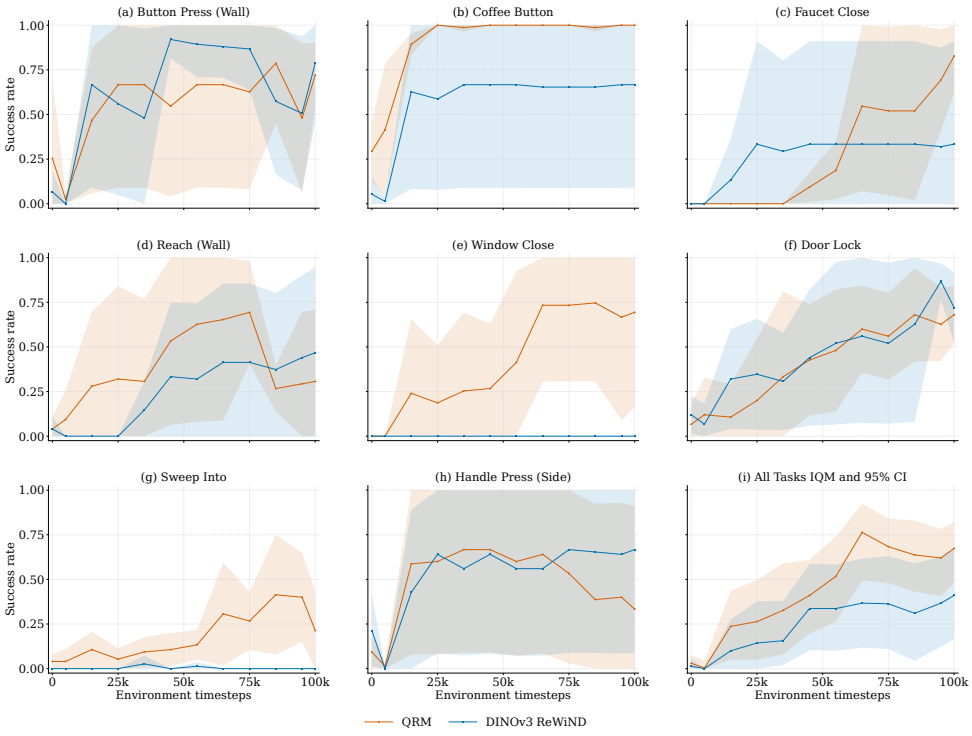


Figure 18: **Per-task success curves with successful but inefficient data.** Learning curves corresponding to Figure 5c.

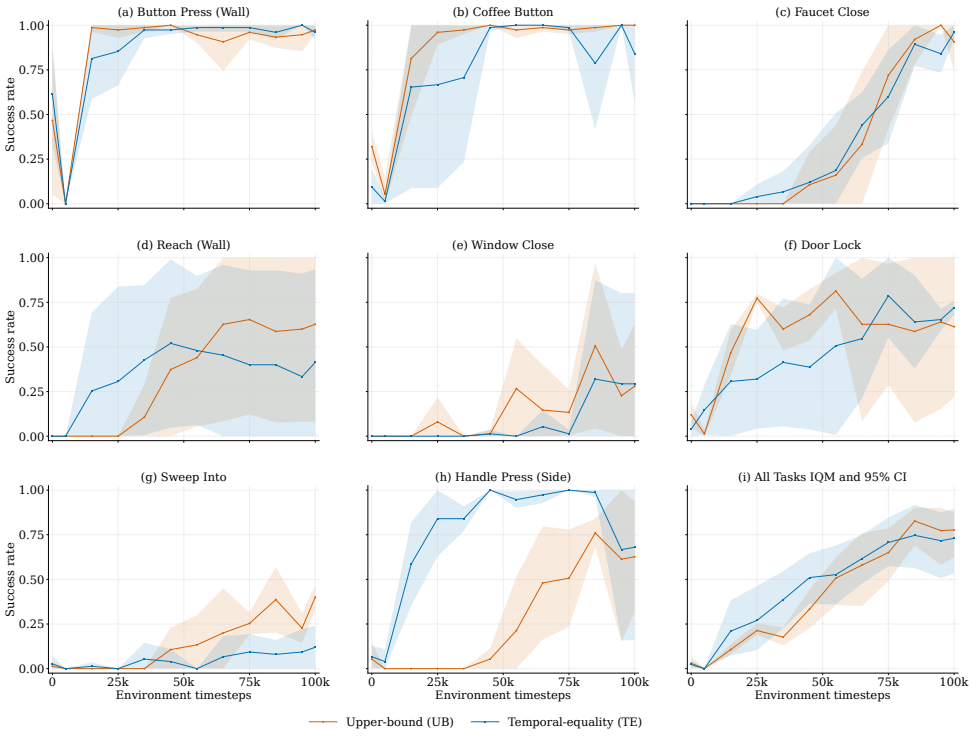


Figure 19: **Per-task success curves for upper bound versus temporal equality.** Learning curves corresponding to Figure 6a. Both models use the same 100% suboptimal MetaWorld data, which contains successful but inefficient trajectories and failed attempts.

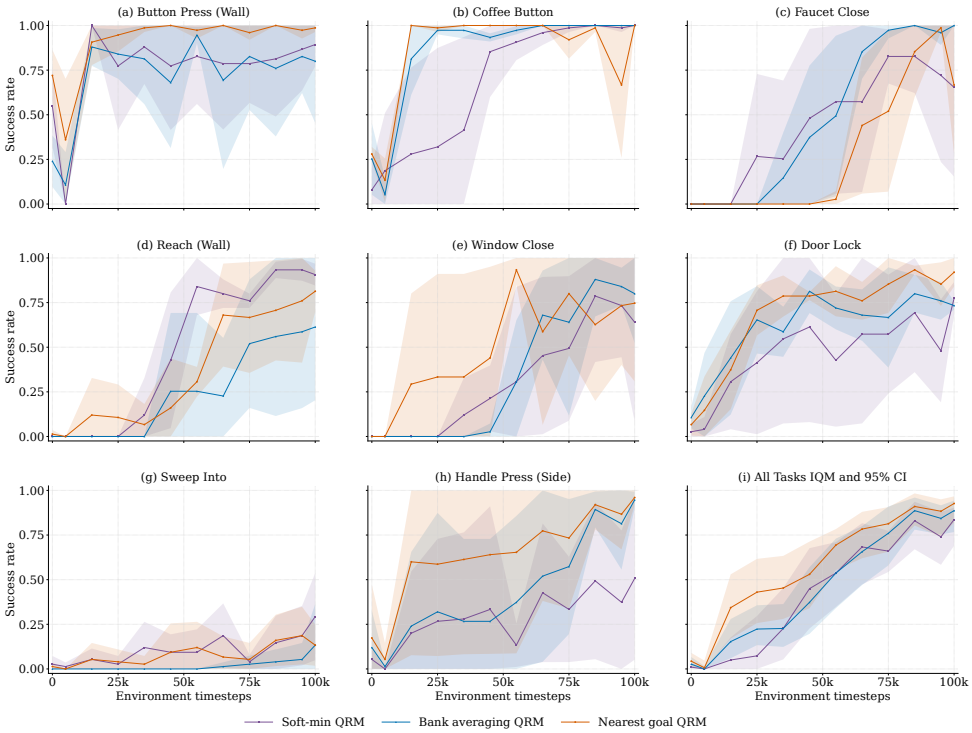


Figure 20: **Per-task success curves for the goal aggregation ablation.** Learning curves corresponding to Figure 6b.

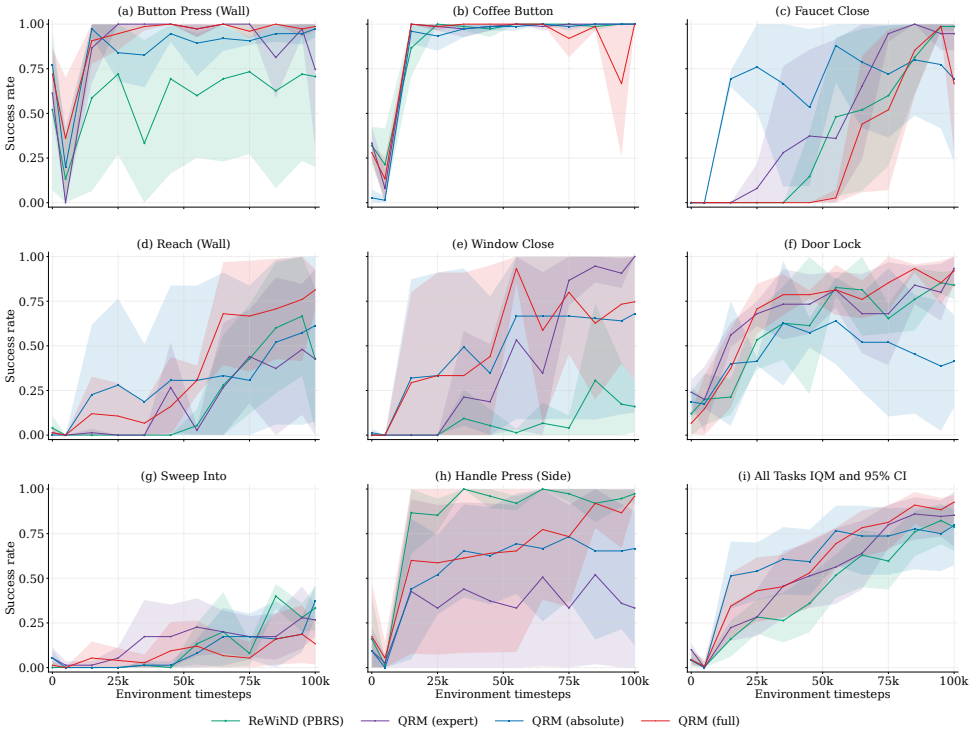


Figure 21: **Per-task success curves for the reward-shaping ablation.** Comparison of QRM (Absolute) with PBRS applied to QRM (expert), QRM (full), and DINOv3 ReWiND.

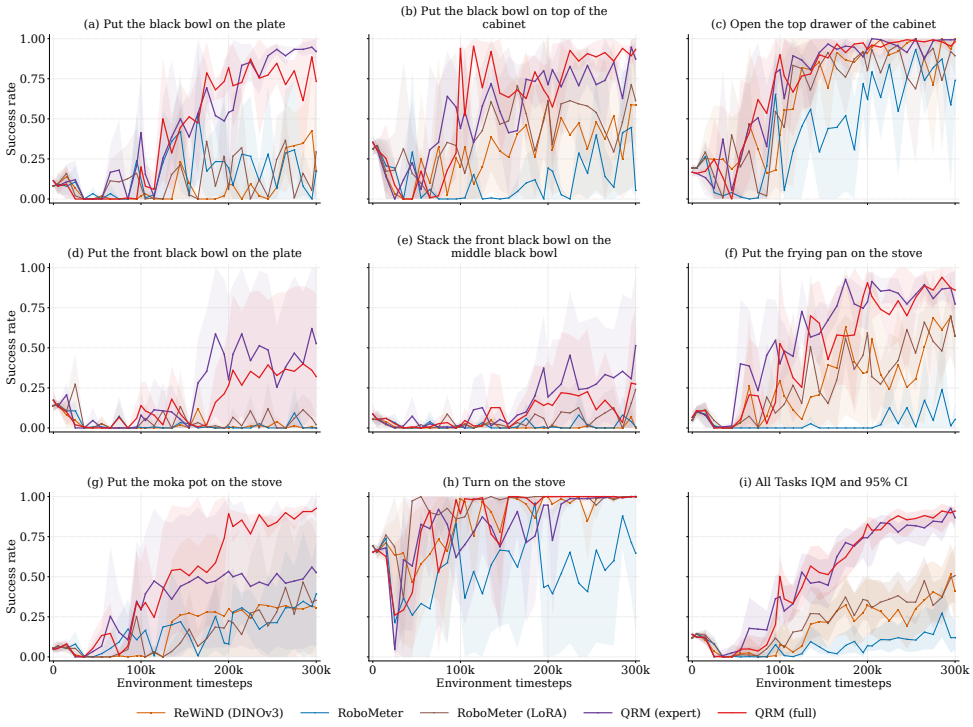


Figure 22: **LIBERO learning curves for pretrained policy fine-tuning.** Eight task panels show mean success with ± 1 sample standard deviation across three policy seeds. The final panel shows IQM over all 24 task–seed rates with pointwise 95% bootstrap intervals. Each evaluation uses 50 initial states. Curves show recorded evaluations without smoothing; lines connect available checkpoints.