

# General Process Reward Modeling for Robotic Reinforcement Learning

## Supplementary Material

### Appendix

This supplementary material provides comprehensive details regarding the proposed method, Dopamine-Reward, and the policy learning framework, Dopamine-RL, along with additional experimental results omitted from the main manuscript due to space constraints. Section A provides a complete and detailed survey of relevant literature as background. Section B provides rigorous theoretical proofs for the bounded global progress, the existence of the “semantic trap,” and the optimal policy invariance, etc. Section C elaborates on the composition, statistics, and sampling strategies of our 35M-sample training dataset. Section D outlines detailed experimental setups for GRM, simulation and real-world evaluations. Section E presents additional sensitivity analysis, evaluating the framework’s robustness to imperfect initial/goal images and noisy expert demonstrations. Section F presents additional qualitative visualizations to demonstrate the effectiveness of our General Reward Model (GRM). Finally, Section G discusses limitations and potential future research directions.

### A. Related Work

**Reinforcement Learning for Robotic Skills.** Reinforcement Learning (RL) has demonstrated the potential to create policies that surpass the capabilities of imitation learning [11, 29, 30, 32, 34, 36, 58, 60, 62], enabling the discovery of novel and robust strategies for complex, contact-rich and dexterous tasks. Research in this area has progressed along two principal directions. The first direction investigates various policy optimization strategies, including offline RL [19, 35, 36], online RL [13, 30, 32, 33, 60], and mixed variants [11, 29, 39]. The second direction explores the efficient application of RL to different model architectures, such as small fully-connected models [36, 39], autoregressive models [11, 34, 58], and diffusion/flow-based models [32, 60, 64]. Independent of the chosen optimization algorithm or policy architecture, a more fundamental bottleneck is to design a reward function that is effective and scalable in real-world RL, which has driven a broad shift away from manual reward engineering [15, 45, 51, 55, 57] toward learning-based reward models.

**Learned Process Reward Models.** In real-world RL, a common practice is to train a success classifier as Outcome Reward Models (ORMs) to provide a binary reward signal [11, 36], which renders exploration prohibitively difficult in complex, long-horizon tasks. To mitigate sparsity, recent work leverages vision–language models (VLMs) as Process Reward Models (PRMs) [2, 8, 37, 38, 61], provid-

ing denser feedback by, for example, predicting progress deltas between paired observations [61] or assigning per-frame progress scores with respect to a language goal [38]. While several methods introduce additional structure by decomposing tasks into steps [8, 17], some open challenges remain (Section 1). First, task-specific designs may limit generalization across diverse activities [8, 17]. Second, many approaches adopt nearly uniform reward allocations, which may underweight the salience of critical sub-steps [38, 61]. In addition, current PRMs typically rely on single-view observations [2, 8, 37, 38, 61], which can impede multi-perspective state estimation and increase sensitivity to occlusions. In contrast, our method, *Dopamine-Reward*, aims to address these issues by learning a general-purpose, step-aware reward model that explicitly fuses multi-view inputs, enabling a more robust and fine-grained reward estimation.

### B. Proof

#### B.1. Proof of Bounded Global Progress (Proof 1)

In this subsection, we provide a formal proof that iteratively applying the predicted relative progress hops guarantees that the reconstructed global progress  $\Phi^*(s)$  remains strictly within the bounds  $[0, 1]$ , provided that the initial state is bounded and the model predictions lie within  $[-1, 1]$ .

First, we define the general recursive update rule. Based on the definition of the hop label  $\mathcal{H}(s_p, s_q)$  in Equation 2, we derive the recursive update rule for estimating the global progress of the next state  $\Phi^*(s_t)$  given the current state  $\Phi^*(s_{t-1})$  and the predicted hop  $H = \mathcal{H}(s_{t-1}, s_t)$ . We assume the normalization where  $\Phi(s_0) = 0$  and  $\Phi(s_M) = 1$ . Rearranging the equation, the update rule is:

$$\Phi^*(s_t) = \begin{cases} \Phi^*(s_{t-1}) + H \cdot [1 - \Phi^*(s_{t-1})] & \text{if } H \geq 0 \\ \Phi^*(s_{t-1}) + H \cdot \Phi^*(s_{t-1}) & \text{if } H < 0 \end{cases} \quad (15)$$

Given that the initial progress  $\Phi^*(s_0) = 0$  and the predicted hop  $H \in [-1, 1]$ , the reconstructed global progress  $\Phi^*(s_t)$  satisfies  $\Phi^*(s_t) \in [0, 1]$  for all steps  $t$ .

We proceed by mathematical induction as follow: (1) *Base Case* ( $t = 0$ ): By definition,  $\Phi^*(s_0) = 0$ , which satisfies  $0 \in [0, 1]$ . (2) *Inductive Step*: Assume that for step  $t - 1$ , the hypothesis holds:  $0 \leq \Phi^*(s_{t-1}) \leq 1$ . Let  $G = \Phi^*(s_{t-1})$  for brevity, where  $G \in [0, 1]$ . We analyze the next state  $\Phi^*(s_t)$  under two cases (*i.e.*, Positive Hop and Negative Hop) based on the sign of the predicted hop  $H$ .

- **Case 1: Positive Hop (Progress)**,  $0 \leq H \leq 1$ .

From Equation (15), the update is written as:

$$\Phi^*(s_t) = G + H(1 - G) \quad (16)$$

Rearranging terms to view this as a convex combination:

$$\Phi^*(s_t) = H + G(1 - H) \quad (17)$$

*Lower Bound:* Since  $G \geq 0$ ,  $H \geq 0$ , and  $(1 - H) \geq 0$ , it follows that  $\Phi^*(s_t) \geq 0$ .

*Upper Bound:* Since  $G \leq 1$ , we substitute the maximum value of  $G$ :

$$\begin{aligned} \Phi^*(s_t) &= H + G(1 - H) \\ &\leq H + 1 \cdot (1 - H) \\ &= H + 1 - H \\ &= 1 \end{aligned}$$

Thus,  $0 \leq \Phi^*(s_t) \leq 1$  when  $H \geq 0$ .

- **Case 2: Negative Hop (Regress)**,  $-1 \leq H < 0$ .

From Equation (15), the update is:

$$\Phi^*(s_t) = G + H \cdot G = G(1 + H) \quad (18)$$

*Lower Bound:* Since  $H \in [-1, 0)$ , the term  $(1 + H) \geq 0$ . Since  $G \geq 0$ , the product  $G(1 + H) \geq 0$ .

*Upper Bound:* Since  $H < 0$ , the term  $(1 + H) < 1$ . Combining this with  $G \leq 1$ :

$$\Phi^*(s_t) = G(1 + H) \leq 1 \cdot (1) = 1$$

Thus,  $0 \leq \Phi^*(s_t) \leq 1$  when  $H < 0$ .

**Conclusion.** Since the property holds for the base case and is preserved in both update scenarios during the inductive step, we conclude that  $\Phi^*(s_t) \in [0, 1]$  for all  $t$ .

## B.2. Proof of the Semantic Trap (Proof 2)

In this subsection, we provide a theoretical derivation demonstrating why a naive dense reward formulation, defined as the direct increment of progress  $r(s_t, a_t, s_{t+1}) = \Phi(s_{t+1}) - \Phi(s_t)$ , fundamentally alters the reinforcement learning objective, leading to the ‘‘Semantic Trap’’ described in the main text.

Consider the standard objective in reinforcement learning, which is to maximize the expected discounted return with a discount factor  $\gamma \in (0, 1)$ :

$$J(\pi) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \middle| s_0 \right]. \quad (19)$$

Substituting the naive progress-difference reward  $r(s_t, a_t, s_{t+1}) = \Phi(s_{t+1}) - \Phi(s_t)$  into the cumulative return for a finite horizon  $T$ :

$$G_T = \sum_{t=0}^{T-1} \gamma^t [\Phi(s_{t+1}) - \Phi(s_t)]. \quad (20)$$

We can split the summation into two distinct terms:

$$G_T = \sum_{t=0}^{T-1} \gamma^t \Phi(s_{t+1}) - \sum_{t=0}^{T-1} \gamma^t \Phi(s_t). \quad (21)$$

By applying a variable substitution  $k = t + 1$  to the first term, we rewrite it as  $\sum_{k=1}^T \gamma^{k-1} \Phi(s_k)$ . The second term can be expanded as  $\Phi(s_0) + \sum_{t=1}^{T-1} \gamma^t \Phi(s_t)$ . Substituting these back into the expression for  $G_T$  yields:

$$\begin{aligned} G_T &= \left[ \sum_{t=1}^{T-1} \gamma^{t-1} \Phi(s_t) + \gamma^{T-1} \Phi(s_T) \right] - \\ &\quad \left[ \Phi(s_0) + \sum_{t=1}^{T-1} \gamma^t \Phi(s_t) \right]. \end{aligned} \quad (22)$$

We now group the terms for each time step  $t \in [1, T - 1]$ :

$$G_T = -\Phi(s_0) + \sum_{t=1}^{T-1} (\gamma^{t-1} - \gamma^t) \Phi(s_t) + \gamma^{T-1} \Phi(s_T) \quad (23)$$

$$= -\Phi(s_0) + (1 - \gamma) \sum_{t=1}^{T-1} \gamma^{t-1} \Phi(s_t) + \gamma^{T-1} \Phi(s_T). \quad (24)$$

Since the progress metric  $\Phi(s)$  is bounded within  $[0, 1]$  and  $\gamma \in (0, 1)$ , the term  $\gamma^{T-1} \Phi(s_T)$  vanishes as  $T \rightarrow \infty$ . The infinite horizon return converges to:

$$G = \lim_{T \rightarrow \infty} G_T = -\Phi(s_0) + (1 - \gamma) \sum_{t=1}^{\infty} \gamma^{t-1} \Phi(s_t). \quad (25)$$

Because  $\Phi(s_0)$  is a constant with respect to the policy  $\pi$ , maximizing the expected return is equivalent to:

$$\arg \max_{\pi} J(\pi) \propto \arg \max_{\pi} \mathbb{E}_{\pi} \left[ \sum_{t=1}^{\infty} \gamma^{t-1} \Phi(s_t) \middle| s_0 \right]. \quad (26)$$

**Conclusion:** The optimization objective implicitly shifts from maximizing the *change* in progress to maximizing the *accumulated value* of progress states over time. This creates a perverse incentive where the agent is encouraged to reach a high-progress state quickly and stagnate there to accumulate rewards at each step, rather than completing the task. This theoretical result confirms the existence of the ‘‘Semantic Trap.’’

## B.3. Derivation of Exponential Discounting from Time-Consistency (Proof 3)

In this subsection, we justify the use of the exponential discount factor  $\gamma = e^{-\lambda h}$ . We start from the principle of *time-consistency* (or memorylessness). Let  $D(t) : \mathbb{R}_{\geq 0} \rightarrow (0, 1]$

be a discount function. The memoryless property implies that the relative discount factor for an additional delay  $\Delta$  should not depend on how much time  $\tau$  has already passed:

$$\frac{D(\tau + \Delta)}{D(\tau)} = D(\Delta), \quad \forall \tau, \Delta \geq 0. \quad (27)$$

Setting  $\tau = t$  and  $\Delta = s$ , this yields the Cauchy functional equation:

$$D(t + s) = D(t)D(s). \quad (28)$$

Transforming to the logarithmic domain with  $\phi(t) = \ln D(t)$ , we obtain the linear equation  $\phi(t + s) = \phi(t) + \phi(s)$ , the unique continuous solution of which is  $\phi(t) = -\lambda t$  for some constant  $\lambda \geq 0$ . Thus, the discount function must take the exponential form:

$$D(t) = e^{-\lambda t}. \quad (29)$$

In the discrete setting with time step  $h$ , this corresponds to the discount factor  $\gamma = D(h) = e^{-\lambda h}$ . This theoretical foundation ensures that our reward formulation is robust to variations in control frequency.

## B.4. Consistency of Reward Shaping in Continuous and Discrete Time (Proof 4)

In this subsection, we derive the continuous-time counterpart of our discrete potential-based reward shaping term  $\gamma\Phi(s_{t+1}) - \Phi(s_t)$ . We demonstrate that the discrete shaping term is mathematically equivalent to the first-order Euler discretization of a specific differential equation. Furthermore, we show that its cumulative sum converges to a boundary term that ensures policy invariance.

### B.4.1. Notation and Definitions

Let  $h = \Delta t$  denote the discretization time step. We map the discrete time steps  $t, t + 1$  to continuous time moments  $t, t + h$ . Let  $s(t)$  denote the state trajectory in continuous time.

- The relationship between the continuous discount rate  $\lambda > 0$  and the discrete discount factor  $\gamma$  is defined as:

$$\gamma = \exp(-\lambda h). \quad (30)$$

- Let  $\Phi(t) := \Phi(s(t))$  denote the potential value along the trajectory. Its total time derivative is:

$$\dot{\Phi}(t) = \frac{d}{dt}\Phi(s(t)). \quad (31)$$

### B.4.2. First-Order Taylor Expansion

We perform a first-order Taylor expansion for both the potential function  $\Phi(s(t + h))$  and the discount factor  $\gamma$  with respect to the step size  $h$ :

$$\Phi(s(t + h)) = \Phi(s(t)) + h\dot{\Phi}(t) + O(h^2), \quad (32)$$

$$\gamma = e^{-\lambda h} = 1 - \lambda h + O(h^2). \quad (33)$$

### B.4.3. Derivation of the Instantaneous Shaping Term

Substituting the expansions into the discrete shaping term  $\gamma\Phi(s_{t+1}) - \Phi(s_t)$  (where  $s_{t+1} \equiv s(t + h)$ ):

$$\begin{aligned} & \gamma\Phi(s(t + h)) - \Phi(s(t)) \\ &= (1 - \lambda h)(\Phi(t) + h\dot{\Phi}(t)) - \Phi(t) + O(h^2) \\ &= \left( \Phi(t) + h\dot{\Phi}(t) - \lambda h\Phi(t) - \lambda h^2\dot{\Phi}(t) \right) \\ &\quad - \Phi(t) + O(h^2) \\ &= h \left( \dot{\Phi}(t) - \lambda\Phi(t) \right) + O(h^2). \end{aligned} \quad (34)$$

Dividing by  $h$  and taking the limit as  $h \rightarrow 0$  yields the *instantaneous density* of the reward shaping:

$$\lim_{h \rightarrow 0} \frac{\gamma\Phi(s(t + h)) - \Phi(s(t))}{h} = \dot{\Phi}(t) - \lambda\Phi(t). \quad (35)$$

This result indicates that the single-step potential-based reward is, to the first order, the rectangular integration of  $\dot{\Phi}(t) - \lambda\Phi(t)$  over the interval  $[t, t + h]$ .

### B.4.4. From Cumulative Sum to Integral Form

We now consider the cumulative discounted sum of the shaping reward over a horizon  $N$ . Let  $t_k = k \cdot h$ . The discount factor at step  $k$  is  $\gamma^k = (e^{-\lambda h})^k = e^{-\lambda t_k}$ . The discrete cumulative sum is:

$$\sum_{k=0}^{N-1} \gamma^k [\gamma\Phi(s_{k+1}) - \Phi(s_k)]. \quad (36)$$

Substituting the first-order approximation derived above:

$$\begin{aligned} & \sum_{k=0}^{N-1} e^{-\lambda t_k} \left[ h(\dot{\Phi}(t_k) - \lambda\Phi(t_k)) + O(h^2) \right] \\ &= \sum_{k=0}^{N-1} h \cdot e^{-\lambda t_k} (\dot{\Phi}(t_k) - \lambda\Phi(t_k)) + O(h). \end{aligned} \quad (37)$$

As  $h \rightarrow 0$ , this Riemann sum converges to the definite integral:

$$\xrightarrow{h \rightarrow 0} \int_0^T e^{-\lambda t} (\dot{\Phi}(t) - \lambda\Phi(t)) dt. \quad (38)$$

### B.4.5. Boundary Terms and Consistency

Using the product rule for differentiation, we observe that the integrand is exactly the total derivative of the discounted potential:

$$\frac{d}{dt} (e^{-\lambda t} \Phi(t)) = e^{-\lambda t} \dot{\Phi}(t) - \lambda e^{-\lambda t} \Phi(t) = e^{-\lambda t} (\dot{\Phi}(t) - \lambda\Phi(t)). \quad (39)$$

Thus, the integral can be evaluated analytically:

$$\begin{aligned} \int_0^T \frac{d}{dt} (e^{-\lambda t} \Phi(t)) dt &= [e^{-\lambda t} \Phi(s(t))]_0^T \\ &= e^{-\lambda T} \Phi(s(T)) - \Phi(s(0)). \end{aligned} \quad (40)$$

Assuming  $\Phi(s)$  is bounded, as  $T \rightarrow \infty$ , the term  $e^{-\lambda T} \Phi(s(T))$  vanishes. The cumulative shaping reward simplifies to a constant boundary term:

$$\lim_{T \rightarrow \infty} \int_0^T e^{-\lambda t} (\dot{\Phi} - \lambda \Phi) dt = -\Phi(s(0)). \quad (41)$$

This confirms that in continuous time, the total shaping reward depends only on the initial state, preserving policy invariance.

#### B.4.6. The ODE / Euler Method Perspective

Finally, we provide an intuitive interpretation using Ordinary Differential Equations (ODEs). Let us define the *discounted potential* as a state variable  $y(t)$ :

$$y(t) := e^{-\lambda t} \Phi(s(t)). \quad (42)$$

The dynamics of  $y(t)$  are governed by:

$$\frac{dy}{dt} = e^{-\lambda t} (\dot{\Phi}(t) - \lambda \Phi(t)). \quad (43)$$

If we apply the **Forward Euler method** to solve this ODE numerically at discrete steps  $t_k$  with step size  $h$ :

$$y(t_{k+1}) \approx y(t_k) + h \cdot \left. \frac{dy}{dt} \right|_{t_k}. \quad (44)$$

Substituting the definitions back:

$$e^{-\lambda(t_k+h)} \Phi(s_{k+1}) \approx e^{-\lambda t_k} \Phi(s_k) + h \cdot e^{-\lambda t_k} (\dot{\Phi}(t_k) - \lambda \Phi(t_k)). \quad (45)$$

Multiplying both sides by  $e^{\lambda t_k}$  (noting that  $e^{-\lambda h} = \gamma$ ):

$$\gamma \Phi(s_{k+1}) \approx \Phi(s_k) + h(\dot{\Phi}(t_k) - \lambda \Phi(t_k)). \quad (46)$$

Rearranging terms yields:

$$\gamma \Phi(s_{k+1}) - \Phi(s_k) \approx h(\dot{\Phi}(t_k) - \lambda \Phi(t_k)). \quad (47)$$

**Conclusion:** The discrete potential-based reward shaping term  $\gamma \Phi_{next} - \Phi_{curr}$  is exactly the update step of the Forward Euler method applied to the differential equation of the discounted potential. This proves that our method is structurally consistent with the underlying continuous-time physics of the task.

#### B.5. Policy Invariance under GRM (Proof 5)

In this subsection, we prove that adding the shaping term derived above preserves the optimal policy. We show this by demonstrating that the cumulative shaped reward telescopes to a boundary term that is independent of the policy's actions.

Let the shaped reward be  $r_{GRM} = r_{env} + F$ , where  $F(s_t, s_{t+1}) = \gamma \Phi(s_{t+1}) - \Phi(s_t)$ . The shaping component of the Q-function,  $S^\pi(s, a)$ , is the expected sum of discounted shaping rewards:

$$S^\pi(s, a) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t (\gamma \Phi(s_{t+1}) - \Phi(s_t)) \right]. \quad (48)$$

We analyze the finite horizon sum  $G_T^F$ :

$$\begin{aligned} G_T^F &= \sum_{t=0}^{T-1} \gamma^{t+1} \Phi(s_{t+1}) - \sum_{t=0}^{T-1} \gamma^t \Phi(s_t) \\ &= \left[ \sum_{k=1}^T \gamma^k \Phi(s_k) \right] - \left[ \Phi(s_0) + \sum_{t=1}^{T-1} \gamma^t \Phi(s_t) \right]. \end{aligned} \quad (49)$$

This is a telescoping sum. The intermediate terms cancel out, leaving only the boundary terms:

$$G_T^F = \gamma^T \Phi(s_T) - \Phi(s_0). \quad (50)$$

Assuming  $\Phi \in [0, 1]$  and  $\gamma < 1$ , taking the limit  $T \rightarrow \infty$  yields  $\lim_{T \rightarrow \infty} \gamma^T \Phi(s_T) = 0$ . Thus:

$$\sum_{t=0}^{\infty} \gamma^t F(s_t, s_{t+1}) = -\Phi(s_0). \quad (51)$$

This matches the integral of the continuous form derived in Proof 4:  $\int_0^\infty \frac{d}{dt} (e^{-\lambda t} \Phi) dt = [e^{-\lambda t} \Phi]_0^\infty = -\Phi(0)$ . Since the shaping term evaluates to  $-\Phi(s)$  (where  $s$  is the state at  $t = 0$ ) regardless of the future trajectory, the Q-values are shifted by a state-dependent constant:

$$Q_{GRM}^\pi(s, a) = Q_{env}^\pi(s, a) - \Phi(s). \quad (52)$$

This shift preserves the ordering of actions:

$$\begin{aligned} Q_{GRM}^\pi(s, a_1) &\geq Q_{GRM}^\pi(s, a_2) \\ &\Updownarrow \\ Q_{env}^\pi(s, a_1) &\geq Q_{env}^\pi(s, a_2). \end{aligned} \quad (53)$$

Therefore, the optimal policy remains:  $\pi_{GRM}^* = \pi_{env}^*$ .

### C. Details of GRM Training Data

In this section, we present a comprehensive breakdown of the training data used to construct our General Reward Model (GRM). To ensure the model possesses robust generalizability across diverse embodiments, environments, and semantic tasks, we curated a massive corpus comprising over 35 million training samples derived from approximately 3,400 hours of raw video footage. This dataset aggregates diverse sources spanning real-world robotics, high-fidelity simulation, and human-centric interactions. We begin by categorizing the constituent datasets (Section C.1),

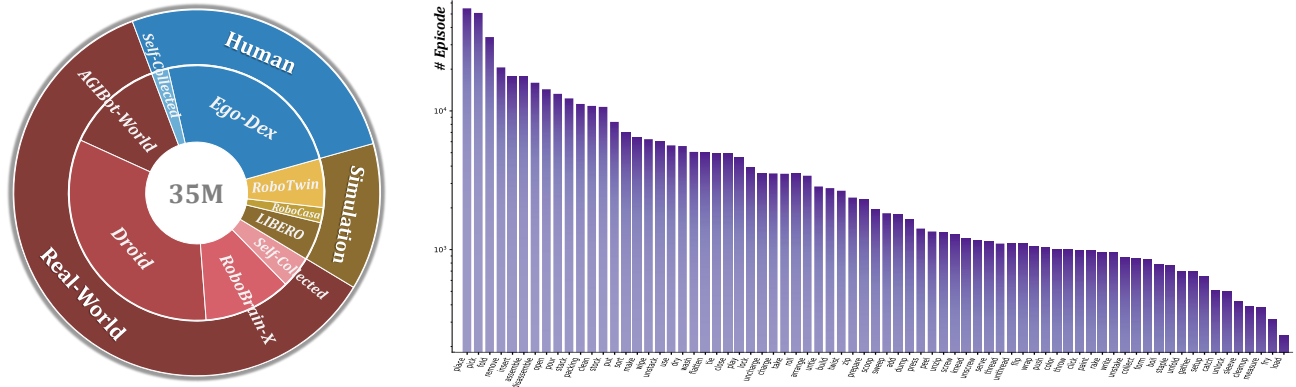


Figure 5. **Overview of GRM training data.** (Left) The hierarchical composition of our 35M-sample training corpus. The dataset is derived from episodes spanning Real-World Robotics, Simulation, and Human-Centric domains, and is further expanded via multi-view augmentation. (Right) The long-tail distribution of task categories sorted by episode count (log scale). The dataset covers a broad spectrum of manipulation skills, ranging from atomic primitives (e.g., pick, push) to complex, multi-stage horizons (e.g., assemble, fold).

followed by an analysis of the embodiment diversity and task statistics (Section C.2). Finally, we detail the rigorous data processing pipeline, specifically our *Stratified Relative Progress Sampling* strategy (Section C.3), which transforms raw trajectories into balanced and high-quality supervision signals for reward learning.

## C.1. Data Sources

Our training data is composed of the following established datasets and self-collected supplements:

### C.1.1. Real-World Datasets

- **AGIBot-World [7]:** A large-scale bimanual manipulation dataset collected on the AGIBot-A2D humanoid platform. It provides approximately 3.4M samples focusing on high-dimensional, dual-arm coordination tasks and contact-rich interactions, serving as a critical source for humanoid embodiment generalization.
- **DROID [25]:** A distributed robot interaction dataset collected across multiple institutions. It contributes 8.98M samples featuring the Franka Emika Panda robot. DROID is characterized by its extreme diversity in background scenes, lighting conditions, and object instances, providing robust priors for visual perception.
- **RoboBrain-X [16]:** A comprehensive skill-learning dataset covering a wide range of everyday manipulation skills. We utilize subsets totaling  $\sim 3.0$ M samples, collected on agile platforms (e.g., Agilex Piper, Galaxea R1, AGIBot-A2D). This data enriches the GRM with fine-grained motion primitives.
- **Self-Collected (Real):** To bridge the domain gap between open datasets and our specific experimental setups, we collected an additional 1.1M samples with the RoboOS [49, 50]. These include specific long-horizon tasks (e.g., folding, assembly) and corner cases to en-

hance model robustness.

Collectively, these real-world datasets ground our GRM in physical reality, effectively mitigating the “sim-to-real” gap often observed in reward modeling. The combination of DROID’s extreme visual diversity with the complex morphologies present in AGIBot-World and RoboBrain-X ensures that the model learns robust, embodiment-invariant representations. This allows the GRM to remain stable across varying lighting conditions, textures, and kinematic structures, which is essential for reliable deployment in unstructured environments.

### C.1.2. Simulation Datasets

- **LIBERO [31]:** A benchmark originally designed for life-long robot learning. We incorporate 1.33M samples from its task suites (Spatial, Object, Goal, 100), which provide procedurally generated tasks with language instructions, aiding the model in aligning visual progress with semantic goals.
- **RoboCasa [41]:** A large-scale simulation framework focused on everyday kitchen activities. It leverages Generative AI to create diverse assets and layouts. We use 0.52M samples to capture realistic object interactions and complex scene semantics in a controlled environment.
- **RoboTwin [9]:** A high-fidelity digital twin dataset designed for bimanual manipulation. We utilize a paired dual-view configuration (yielding 1.68M samples from 839k trajectories), which helps the model learn geometry-aware representations crucial for depth disambiguation.

While real-world data offers physical fidelity, these simulation environments provide a controlled testbed for learning precise semantic and geometric alignments. The procedural generation inherent in LIBERO and RoboCasa exposes the model to a vast combinatorial space of task instructions and object layouts, fostering strong instruction-following capa-



Table 6. **Statistics of Raw Data Sources.** The table lists the *raw* frame counts (in thousands, ‘k’) before augmentation. The final training set is expanded to **35M** via multi-view expansion and augmentation strategies.

Domain	Dataset Source	Raw Samples
Real-World	AGIBot-World [7]	3,400k
	DROID [25]	8,983k
	RoboBrain-X [16]	3,025k
	Self-Collected (Real)	1,107k
	<i>Subtotal</i>	<i>16,515k</i>
Simulation	LIBERO [31]	1,330k
	RoboTwin [9]	1,678k
	RoboCasa [41]	523k
	<i>Subtotal</i>	<i>3,531k</i>
Human	EgoDex [18]	6,610k
	Self-Collected (Human)	574k
	<i>Subtotal</i>	<i>7,184k</i>
<b>Total</b>	<b>Raw Corpus</b>	<b>27,230k</b>

bilities. Furthermore, the clean, occlusion-free labels and paired views from RoboTwin allow the GRM to learn fine-grained geometric correspondences that are often noisy or unavailable in real-world data.

### C.1.3. Human-Centric Datasets

- **EgoDex [18]:** A large-scale egocentric video dataset capturing dexterous human hand-object interactions. With 6.61M samples, this dataset provides strong priors for understanding hand-object affordances and manipulation logic independent of robot morphology.
- **Self-Collected (Human):** We supplemented the human data with 0.57M samples of domain-specific demonstrations, ensuring coverage of tasks analogous to our robot evaluation protocols.

Integrating human video data is pivotal for scaling general manipulation intelligence beyond the limits of available robot demonstrations. By leveraging the massive scale of EgoDex, our GRM acquires universal object affordance priors, learning “how” objects should be manipulated regardless of the specific actuator. This cross-embodiment transfer is critical for evaluating progress in novel tasks where robot-specific data may be scarce, enabling the reward model to generalize purely based on the observed state changes of the objects themselves.

## C.2. Statistics and Embodiment Diversity

Our compiled dataset represents one of the most comprehensive collections for robotic manipulation to date. As detailed in Table 6, the final training corpus of 35 million samples is strategically balanced to maximize generaliza-

Table 7. **Distribution by Robot Embodiment.** Our dataset spans diverse robot hardware, from single-arm industrial robots to bi-manual humanoids, ensuring strong physical generalization.

Robot Embodiment	Primary Data Sources	Samples
Franka Emika Panda	DROID, LIBERO, RoboCasa	8,983k
AGIBot-A2D	AGIBot-World, RoboBrain-X	3,400k
Agilex Piper	RoboBrain-X, Self-Collected	3,552k
ARX-X5	RoboTwin	882k
Galaxea	RoboBrain-X, Self-Collected	695k
UR5	Self-Collected	433k

tion. Real-world interaction data constitutes the majority ( $\sim 60\%$ ) to ensure the model is grounded in physical reality, while high-fidelity simulation ( $\sim 13\%$ ) and large-scale human videos ( $\sim 26\%$ ) provide necessary semantic breadth and affordance priors that are difficult to obtain from robots.

**Embodiment Agnosticism.** A core design principle of our GRM is robustness to morphological variations. As illustrated in Table 7, the dataset covers a wide spectrum of robot embodiments, preventing the model from overfitting to specific kinematics or camera calibrations. By training on this heterogeneous mixture, the GRM learns to focus on the *state changes of the objects* rather than the robot’s motion, enabling zero-shot transfer to unseen robot configurations.

**Task and Instruction Diversity.** Figure 5 (Right) highlights the semantic diversity of the dataset. The task distribution follows a natural long-tail pattern, spanning from high-frequency atomic primitives (*e.g.*, pick, place, push) that build a solid generalist foundation, to complex, multi-stage horizons (*e.g.*, fold, assemble, pour) that challenge the model’s ability to track long-term progress. To further enhance semantic robustness, we employed Gemini 2.5 Pro [14] to re-annotate task instructions, thereby introducing linguistic diversity and reducing overfitting to rigid template prompts.

## C.3. Sampling Strategy and Data Balancing

To train the General Reward Model (GRM) effectively, it is crucial to generate training pairs that cover the full spectrum of task progress dynamics. Naively sampling random pairs from a trajectory typically results in a long-tail distribution dominated by small, positive progress steps, leading to model bias. To address this, we use the Stratified Relative Progress Sampling strategy combined with an aggressive augmentation pipeline, as described in main paper. More details are as follow:

### C.3.1. Hop-based Relative Progress Formulation

Consistent with the main paper, we define the ground-truth relative progress label  $\mathcal{H}(s_p, s_q)$  using a *relative-relative* formulation. Let a trajectory consist of  $M$  steps  $\{s_0, \dots, s_M\}$  with linear global progress  $\Phi(s_i) = i/M$ . Without loss of generality, we normalize the global progress

such that the initial state has  $\Phi(s_0) = 0$  and the goal state has  $\Phi(s_M) = 1$ . For any training pair  $(s_p, s_q)$ :

$$\mathcal{H}(s_p, s_q) = \begin{cases} \frac{\Phi(s_q) - \Phi(s_p)}{1 - \Phi(s_p)} & \text{if } q \geq p \text{ (PROGRESS)} \\ \frac{\Phi(s_q) - \Phi(s_p)}{\Phi(s_p)} & \text{if } q < p \text{ (REGRESS)}. \end{cases} \quad (54)$$

This formula ensures that forward progress is normalized by the *remaining distance*  $(1 - \Phi(s_p))$ , making late-stage steps statistically significant, while regression is normalized by the *accumulated distance*  $(\Phi(s_p))$ . The continuous output  $\mathcal{H} \in [-1, 1]$  is quantized into integer percentage tokens for VLM training. See Figure 8 for the whole prompt.

### C.3.2. Hierarchical Score-Gap Stratified Sampling

To prevent the model from overfitting to specific step sizes, we implement a two-stage sampling mechanism:

- **Score Balancing:** We discretize the progress score range  $[-100\%, +100\%]$  into  $N_{score}$  uniform bins (e.g.,  $N_{score} = 25$ ). We generate a large candidate pool of random pairs and fill these bins to enforce a uniform distribution of reward values. This ensures the model encounters rare events, such as significant regression (errors) or large forward jumps, as frequently as common small steps.
- **Temporal Gap Diversification:** Within each score bin, we further categorize samples based on their temporal distance  $\Delta t = |t_{post} - t_{pre}|$  into  $N_{gap}$  bins (e.g.,  $N_{gap} = 4$ ). This method explicitly forces the model to distinguish between *fast progress* (large score change in short time) and *slow progress* (large score change over long time), effectively decoupling visual state change from the temporal duration of trajectories.

### C.3.3. Zero-Hop Anchoring

Standard comparative ranking often struggles with static or near-static pairs, leading to hallucinated progress values. To mitigate this, we explicitly inject a fixed ratio  $\alpha$  (e.g.,  $\alpha = 5\%$ ) of **Zero-Score Samples**. These samples are constructed by selecting pairs  $(t, t + \delta)$  where the temporal delta  $\delta$  is negligible (randomly sampled within  $\pm 10\%$  of the local window). These pairs are labeled with a strict score of 0%. This creates a “semantic anchor” for the model, teaching it that visual similarity implies zero progress, thereby reducing false positives in stable robotic states.

### C.3.4. Data Augmentation and Robustness

Finally, to bridge the gap between the raw data count and our effective training scale, and to ensure robustness against perceptual failures, we employ an aggressive data augmentation pipeline:

- **Multi-View Expansion:** We maximize the utility of multi-camera setups (e.g., permuting available wrist and

third-person views) by treating distinct viewpoints as complementary training signals. This strategy expands the dataset size from 27.2M raw samples to 35M training samples, improving the model’s geometric consistency.

- **Perceptual Robustness Training:** To prevent the model from over-relying on any single input modality, we introduce *Random Viewpoint Dropout* (randomly masking specific camera feeds) and *Context Dropout* (randomly masking  $s_{init}$  or  $s_{goal}$  tokens) during training. This forces the GRM to infer progress from partial or occluded observations, significantly enhancing resilience in real-world deployments.

## D. Experiment Details

### D.1. Progress Consistency Checking (Optional)

While the multi-perspective fusion via averaging (Equation (8)) serves as a baseline, its naive application in online RL faces the risk of Out-of-Distribution (OOD) hallucination. Due to the inherent limitations of data coverage, it is impossible for the training set to encompass every corner of the state space. During RL, the policy inevitably explores unseen regions where the reward model may yield spurious high signals, leading to “reward hacking.” To address these, we propose a bi-directional consistency checking strategy that leverages consistency as a proxy for reliability, which is motivated by the observation that forward  $\Phi_F^*$  and backward  $\Phi_B^*$  predictions tend to exhibit significant divergence in OOD scenarios or observations, whereas they remain consistent in familiar states.

**Consistency-Aware Weighting.** We first define the mean estimated progress  $\bar{\Phi}^*(s_t) = (\Phi_F^*(s_t) + \Phi_B^*(s_t))/2$ . To quantify uncertainty, we calculate a normalized discrepancy metric:

$$\Delta_{\text{norm}}(s_t) = \frac{|\Phi_B^*(s_t) - \Phi_F^*(s_t)|}{\bar{\Phi}^*(s_t) + \epsilon}, \quad (55)$$

where  $\epsilon$  is a small constant for numerical stability. Normalization by  $\bar{\Phi}^*$  ensures that discrepancies are penalized more heavily during the early stages (where  $\Phi$  is small), as precise guidance is critical initially. We then derive a confidence weight  $w_t \in (0, 1]$  using a Gaussian kernel with sensitivity  $\alpha$ :

$$w_t = \exp(-\alpha \cdot (\Delta_{\text{norm}}(s_t))^2). \quad (56)$$

**Conservative State Update.** To prevent the policy from exploiting erroneous estimates in OOD scenarios, we employ a conservative update rule for the maintained progress state  $\Phi^*(s_t)$  instead of Equation (8):

$$\Phi^*(s_t) = \Phi^*(s_{t-1}) + \frac{w_t}{2} \cdot (\bar{\Phi}^*(s_t) - \Phi^*(s_{t-1}) + \Delta\Phi_{t-1,t}^*). \quad (57)$$

This mechanism acts as a semantic filter: it ignores uncertain updates when  $w_t \rightarrow 0$  (retaining  $\Phi^*(s_{t-1})$ ) and fully trusts the estimate when consistency is high ( $w_t \rightarrow 1$ ).

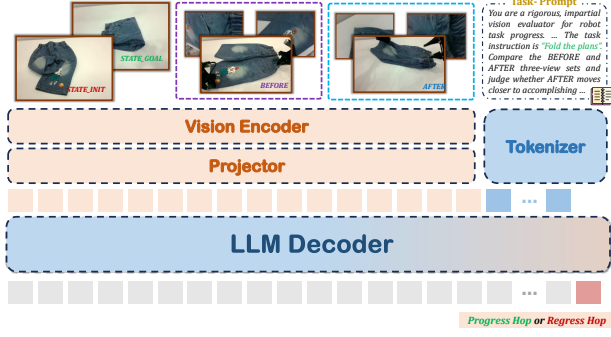


Figure 6. **Overview of GRM model structure.** The GRM is built upon the Qwen2.5-VL architecture. It processes a multi-modal interleaved input sequence consisting of task text instructions and multi-view images: the initial state ( $s_{init}$ ), the goal state ( $s_{goal}$ ), and the paired “BEFORE” ( $s_{pre}$ ) and “AFTER” ( $s_{post}$ ) observation sets. The visual signals are processed by a shared Vision Encoder and Projector, then fed into the LLM Decoder, which autoregressively predicts a quantized relative progress token (e.g.,  $\langle \text{score} \rangle + 15\% \langle / \text{score} \rangle$ ) or a regress token (e.g.,  $\langle \text{score} \rangle - 4\% \langle / \text{score} \rangle$ ).

## D.2. GRM Training and Evaluation

**Model Details.** Our GRM leverages the RoboBrain 2.0 [52] architecture, as shown in Figure 6, exploring two parameter scales: a lightweight 3B variant for efficient inference and a powerful 8B variant for maximum reasoning capability. For inputs, the model accepts a multimodal prompt sequence. Visual inputs include the Initial State  $I_{init}$ , Goal State  $I_{goal}$ , and two sets of multi-view observations for the transition being evaluated:  $S_{pre} = \{I_{pre}^v\}_{v=1}^V$  and  $S_{post} = \{I_{post}^v\}_{v=1}^V$ , where  $V$  is the number of available camera views (e.g., one front, two wrists). Textual inputs include the system prompt defining the rigorous evaluator persona and the specific task instruction  $d_{task}$ . For outputs, the model outputs a discrete token representing the relative progress hop  $\mathcal{H}(S_{pre}, S_{post}) \in [-1, 1]$ .

**Training Infrastructure.** We conducted large-scale training on a high-performance cluster consisting of  $128 \times$  NVIDIA H100 (80GB) GPUs. The training framework utilizes the Megatron-LM architecture [48] for efficient distributed training. We employed Tensor Parallelism (TP) and Pipeline Parallelism (PP) to maximize throughput. Detailed hyperparameters are provided in Table 8. The 3B model was trained for approximately 8 days, while the 8B model required 14 days to converge on the 35M-sample dataset.

**Evaluation Protocols.** We evaluate the GRM on two distinct tasks to assess both its fine-grained temporal understanding and its high-level task success judgment via vLLM engine [28]. For *Video Frame Rank-Correlation*, we use Value-Order Consistency (VOC) [38] as main metric, which assesses whether the reward model correctly orders states

Table 8. GRM Training Hyperparameters.

Hyperparameter	GRM-3B	GRM-8B
Global Batch Size	512	256
LR: $\{\psi_v^{ViT}, \phi_v^{LLM}\}$	$\{5e^{-6}, 1e^{-5}\}$	$\{5e^{-6}, 1e^{-5}\}$
LR Scheduler	Cosine Decay	Cosine Decay
Warmup Ratio	0.03	0.03
Optimizer	AdamW	AdamW
Weight Decay	0.1	0.1
Max Sequence Length	8192	8192
Tensor Parallelism (TP)	1	2
Pipeline Parallelism (PP)	1	2
GPU nodes	16×8 H100	16×8 H100
Training Duration	~ 8 Days	~ 14 Days

based on temporal progress. We measure VOC on unseen test data, where details are as follow:

- **Data Selection:** For each dataset (e.g., AgiBot-World, DROID), we randomly sample 10 trajectories per subclass in each dataset from the hold-out validation set.
- **Metric:** Given two frames  $t_A$  and  $t_B$  where  $t_A < t_B$ , a correct prediction requires  $R(s_{t_A}) < R(s_{t_B})$ . The VOC score is the correlation coefficient  $[-1, 1]$ .
- **Baselines:** We compare with VLAC [61] and GVL [38] using their official codebase or recommended prompts.

For *Task Completion Judgment*, to verify if the GRM can serve as a reliable success detector, we collected 60 real-world rollouts for each of three challenging tasks: *Stacking Blocks*, *Folding T-shirt / Pants*, and *Clearing Desktop*. These rollouts are uniformly distributed into 20 Successful (SE), 20 Partially Successful (PSE), and 20 Failed (FE) episodes. We define the automated judgment logic following SARM [8]. Let  $P_t \in [0, 1]$  be the accumulated global progress at step  $t$  predicted by the GRM. The trajectory classification label is determined by:

$$\text{Label} = \begin{cases} \text{SE}, & \text{if } P_{\text{final}} > 0.8 \wedge \frac{3}{T} \sum_{t=2T/3}^T P_t > 0.6, \\ \text{PSE}, & \text{if } \frac{1}{T} \sum_{t=1}^T P_t \geq \xi, \\ \text{FE}, & \text{otherwise.} \end{cases} \quad (58)$$

where  $\xi = 0.4$  is a threshold for partial progress.

## D.3. Simulation Experiments

This section details the simulation experiments: the benchmark and two different “RL algorithm + VLA model” settings we employ to validate our methods.

### D.3.1. Benchmark

We evaluate our framework on the LIBERO-Goal suite [31], a challenging subset of the LIBERO benchmark consisting



of 10 distinct long-horizon manipulation tasks. These tasks require the agent to reason about specific goal configurations and perform precise object interactions, serving as a rigorous testbed for progress-based reward modeling.

### D.3.2. Setting 1: PPO + OpenVLA-OFT

In this setting, we combine PPO (Proximal Policy Optimization) [47], a stable and sample-efficient on-policy RL algorithm, with OpenVLA-OFT [27], an optimized vision-language-action (VLA) model, to build the reinforcement learning pipeline, implementing it via the RLinf codebase [60]. OpenVLA-OFT acts as the base policy model, refined from the OpenVLA backbone through an Optimized Fine-Tuning (OFT) recipe that integrates parallel decoding with action chunking (for high-throughput action generation), continuous action representation (avoiding lossy discretization), and an L1 regression objective (simplifying training while preserving precision). It maps multimodal inputs to continuous robot actions in one forward pass. For LIBERO-Goal’s long-horizon tasks, we set an action chunk size of 8 to capture temporal dependencies and reduce compounding errors.

PPO optimizes the OpenVLA-OFT policy by confining updates to a trust region, using a clipped surrogate objective to balance exploration and exploitation. Its core optimization objective is defined as:

$$J^{\text{PPO}}(\theta) = \mathbb{E}_t \left[ \min \left( \rho_t(\theta) \hat{A}_t, \text{clip}(\rho_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right]. \quad (59)$$

Here,  $\rho_t(\theta) = \frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{\text{old}}}(a_t|o_t)}$  is the ratio between the current policy  $\pi_\theta$  and the fixed rollout policy  $\pi_{\theta_{\text{old}}}$ ,  $\hat{A}_t$  is the estimated advantage at timestep  $t$ , and  $\epsilon$  is the clipping parameter that prevents excessive updates. We follow RLinf-VLA [60]’s PPO best practices: using action-level value estimation (superior for chunked policies) and partial reset (resetting sub-environments post-completion to boost sample efficiency).

### D.3.3. Setting 2: ReinFlow + $\pi_0$

In this setting, we combine ReinFlow [63], a flow-matching based reinforcement learning algorithm, and  $\pi_0$  [6], a flow-matching based vision-language-action model to build the reinforcement learning pipeline and implement it using the RLinf [60] codebase. The training is conducted on a compute node equipped with 8 NVIDIA H100 GPUs. The total training time is approximately 50 hours, achieving an average success rate of 81% across the LIBERO-Goal tasks (Figure 7). ReinFlow facilitates stable fine-tuning by injecting learnable noise into the flow matching policy’s deterministic path, converting it into a discrete-time Markov process. Specifically, the denoising process is modified as

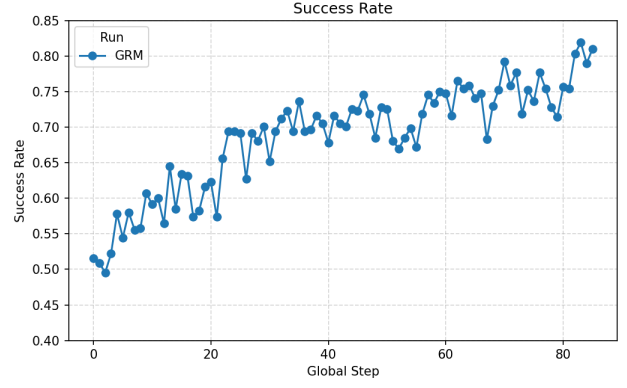


Figure 7. **Training Curve on LIBERO-Goal.** The success rate of our ReinFlow agent fine-tuned with GRM-based reward shaping. The agent demonstrates stable convergence and achieves an average success rate of over 80% across the benchmark tasks.

follows:

$$a^{k+1} \sim \mathcal{N}(a^k + v_\theta(t_k, a^k, o) \Delta t_k, \sigma_{\theta'}^2(t_k, a^k, o)) \quad (60)$$

where  $v_\theta$  is the velocity network and  $\sigma_{\theta'}$  is a learnable noise network. This formulation allows for an exact computation of the transition probability:

$$\pi^\theta(a^{k+1}|a^k, o) = \ln \mathcal{N} \left( a^{k+1} \middle| a^k + v_\theta(t_k, a^k, o) \Delta t_k, \sigma_{\theta'}^2(t_k, a^k, o) \right) \quad (61)$$

The policy is then optimized using a PPO-style objective that jointly updates the velocity and noise networks:

$$\theta, \theta' = \arg \min_{\theta, \theta'} \sum_{i=1}^B \left[ -A^{\bar{\theta}_{\text{old}}}(o_i, a_i) \sum_{k=0}^{K-1} \ln \pi^{\bar{\theta}}(a_i^{k+1}|a_i^k, o_i) + \alpha \cdot \mathcal{R}(a_i, o_i) \right] \quad (62)$$

where  $A$  is the advantage function, and  $\mathcal{R}$  is a regularization term (e.g., entropy). Furthermore, detailed hyperparameters for the ReinFlow training are provided in Table 9.

Table 9. Hyperparameters for LIBERO-Goal Experiments.

Hyperparameter	Value
Algorithm	ReinFlow
Consistency Sensitivity $\alpha$	20
Discount Factor $\gamma$	0.99
Batch Size	1792
Learning Rate	5e-6
Compute Resources	8 × H100

## D.4. Real-World Experiments

This section details the real-world robotic experiments: the algorithm we use and 8 representative tasks we conduct.

### D.4.1. Algorithm

We adopt the Consistency-based Reinforced Fine-Tuning (ConRFT) algorithm [12] in our real-world experiments. ConRFT is an Cal-QL [40] algorithm variant designed as an offline-to-online reinforcement learning (RL) method rooted in Q-learning for fine-tuning pre-trained Vision-Language-Action (VLA) models. ConRFT extends Cal-QL’s framework by adding a consistency policy and a behavior cloning loss, enabling efficient and safe adaptation for real-world robotic manipulation tasks. Key components are as follows:

**Algorithm Overview.** ConRFT operates in two sequential phases: offline initialization and online adaptation. It maintains two data buffers: a demonstration buffer ( $\mathcal{D}$ ) and a replay buffer ( $\mathcal{R}$ ). The offline phase leverages a small set of human demonstrations (20-30 trajectories) stored in  $\mathcal{D}$  to initialize a policy via behavior cloning (BC) and calibrated Q-learning (Cal-QL) [40], ensuring stable and safe initial behavior without requiring real-world exploration. The online phase then refines this policy through interaction with the physical world, storing transitions in  $\mathcal{R}$ . A human-in-the-loop (HIL) component intervenes to correct unsafe actions during online deployment, with these corrected trajectories added back to  $\mathcal{D}$  to guide subsequent policy updates. Symmetric sampling from the combined dataset  $\mathcal{D} \cup \mathcal{R}$  is employed to mitigate distribution shift between the offline and online stages.

**Policy model.** The policy model is built on the Octo-Small model [54], a lightweight VLA backbone selected for its balance of performance and inference efficiency. The model is trained with visual encoders and transformer backbone and its original action head is replaced with a consistency policy [10], a diffusion-based module that maps Gaussian-sampled random actions to expert-like behaviors, conditioned on the observation embeddings.

**Offline Stage.** The policy is initialized exclusively using data from the demonstration buffer  $\mathcal{D}$ , with a diffusion-based consistency policy (parameterized by  $\psi$ ) serving as the action head. The unified training objective integrates BC and Q-learning to balance expert imitation and reward alignment, defined as

$$\mathcal{L}_\pi^{\text{offline}}(\psi) = \beta \mathcal{L}_\pi^{\text{BC}} + \eta \mathcal{L}_\pi^{\text{Q}}, \quad (63)$$

where  $\beta$  and  $\eta$  are hyperparameters that balance the two loss terms. The BC loss minimizes the Euclidean distance between actions generated by the consistency policy and ex-

pert demonstrations from  $\mathcal{D}$ , formulated as

$$\mathcal{L}_\pi^{\text{BC}} = \mathbb{E}_{\substack{(s,a) \sim \mathcal{D} \\ m \sim U[1, M-1]}} \left[ \left\| f_\psi(a + k_m z, k_m \mid E_\phi(s)) - a \right\|_2 \right], \quad (64)$$

in which  $(s, a) \sim \mathcal{D}$  denotes sampling state-action pairs from the demonstration buffer,  $m \sim U[1, M-1]$  indicates random sampling of a diffusion sub-interval,  $k_m$  is the diffusion step corresponding to sub-interval  $m$ ,  $z \sim \mathcal{N}(0, I)$  is Gaussian noise following a standard normal distribution,  $E_\phi(s)$  is the observation embedding output by the frozen Octo-Small backbone,  $f_\psi$  is the consistency policy module parameterized by  $\psi$ , and  $\|\cdot\|_2$  denotes the Euclidean distance. The Q loss maximizes the action-value estimates from a learned critic network  $Q_\theta$  (where  $\theta$  are the critic’s parameters), given by

$$\mathcal{L}_\pi^{\text{Q}} = -\mathbb{E}_{s \sim \mathcal{D}, a \sim \pi_\psi} [Q_\theta(s, a)], \quad (65)$$

with  $a \sim \pi_\psi$  indicating actions sampled from the consistency policy. The critic  $Q_\theta$  is updated using the Cal-QL objective:

$$\begin{aligned} \mathcal{L}_Q^{\text{offline}}(\theta) = & \alpha (\mathbb{E}_{s \sim \mathcal{D}, a \sim \pi} [\max(Q_\theta(s, a), V^\mu(s))] \\ & - \mathbb{E}_{(s,a) \sim \mathcal{D}} [Q_\theta(s, a)]) \\ & + \frac{1}{2} \mathbb{E}_{(s,a,s') \sim \mathcal{D}} \left[ (Q_\theta(s, a) - \mathcal{B}^\pi \bar{Q}_{\bar{\theta}}(s, a))^2 \right]. \end{aligned} \quad (66)$$

In this formula,  $\alpha$  is the conservative penalty hyperparameter,  $V^\mu(s)$  is the value of a reference policy  $\mu$  that stabilizes estimates for out-of-distribution actions,  $(s, a, s') \sim \mathcal{D}$  samples transition triples (state, action, next state) from  $\mathcal{D}$ ,  $\mathcal{B}^\pi$  is the Bellman backup operator defined as  $\mathcal{B}^\pi \bar{Q}(s, a) = r(s, a) + \gamma \mathbb{E}_{a' \sim \pi} \bar{Q}(s', a')$ , and  $\bar{Q}_{\bar{\theta}}$  is a delayed target Q-network whose parameters  $\bar{\theta}$  are fixed during critic updates to ensure training stability.

**Online Stage.** The online phase refines the policy using combined data from the demonstration buffer  $\mathcal{D}$  which is augmented with corrected trajectories from HIL interventions and the replay buffer  $\mathcal{R}$  that stores online interaction transitions. The policy update objective retains the same structure as the offline stage but with adjusted weights to prioritize reward-driven exploration over expert imitation:

$$\mathcal{L}_\pi^{\text{online}}(\psi) = \beta' \mathcal{L}_\pi^{\text{BC}} + \eta' \mathcal{L}_\pi^{\text{Q}}, \quad (67)$$

where  $\beta'$  is reduced and  $\eta'$  is increased, shifting the focus toward learning from real-world feedback. The BC and Q losses follow the same formulations as in the offline phase, with the only difference being that data sampling is performed from  $\mathcal{D} \cup \mathcal{R}$  instead of  $\mathcal{D}$  alone, and all symbols retain their previously defined meanings. The critic  $Q_\theta$  is

updated via a standard Bellman error loss without the conservative penalty (as online data reduces distribution shift and mitigates overestimation risks), given by

$$\mathcal{L}_Q^{\text{online}}(\theta) = \mathbb{E}_{(s,a,s') \sim \mathcal{D} \cup \mathcal{R}} \left[ (Q_\theta(s,a) - \mathcal{B}^\pi \bar{Q}_{\bar{\theta}}(s,a))^2 \right], \quad (68)$$

ensuring accurate value estimation as the policy adapts to real-world dynamics.

**Hyperparameters** Detailed hyperparameters for real-world experiments are provided in Table 10.

#### D.4.2. Real-World Tasks

In real-world experiments, we selected 8 representative tasks to verify the promotional effect of our reward model on physical robotic reinforcement learning. These tasks cover single-arm tasks, dual-arm tasks, fine-grained tasks, and long-horizon tasks, with detailed descriptions of each task provided as follows.

**Insert Square:** The end-effector of the robotic arm grasps a square block with four holes. On the table, there is a target board designed for block insertion, which is equipped with four upright pegs corresponding to the four holes on the block. The robot is required to adjust the position and orientation of the block to insert it onto the pegs of the board. This task demands millimeter-level precision tolerance, categorizing it as a single-arm fine-grained task.

**Pick and Place:** A white gasket and a toy are placed on the table. The robotic arm needs to first grasp the toy and then place it accurately on the gasket. The initial positions of both the gasket and the toy are randomly generated within a predefined range. This task is defined as a single-arm fine-grained long-horizon task.

**Complete Circuit:** A disconnected circuit is placed on the table, and the right arm of the dual-arm robotic system holds a battery. The robot is required to first insert the battery (held by the right arm) into the battery slot, then use the left arm to toggle the circuit switch, thereby lighting up the bulb in the middle of the circuit. This task is classified as a dual-arm fine-grained long-horizon task.

**Arrange Flowers:** A vase is placed on the table, with the left and right arms of the robotic system each holding a flower. The robot needs to sequentially insert the two flowers into the vase. This task is categorized as a dual-arm fine-grained long-horizon task.

**Fold Towel:** A towel is laid on the table, and the robotic arm is required to fold it neatly. This task is defined as a dual-arm fine-grained long-horizon task.

**Build Blocks:** Three building blocks are placed on the table. The robotic arm needs to stack them sequentially to form a castle-like structure. This task is classified as a dual-arm fine-grained long-horizon task.

**Cap the Pen:** The dual-arm robot holds a pen in one arm and a pen cap in the other. It needs to slowly adjust the

Table 10. Real-World Experiment Hyperparameters.

Hyperparameter	Symbol	Value
Global Batch Size	-	256
Learning Rate	-	$3 \times 10^{-4}$
Reward Discount Factor	$\gamma$	0.98
Offline BC Loss Weight	$\beta$	1.0
Offline Q Loss Weight	$\eta$	0.1
Conservative Penalty Coefficient	$\alpha$	0.1
Online BC Loss Weight	$\beta'$	0.5
Online Q Loss Weight	$\eta'$	1.0
Compute Resources	-	$1 \times \text{RTX4090}$

positions and orientations of the two objects to finally fit the cap onto the pen. This task is categorized as a dual-arm fine-grained task.

**Zip the Bag:** The left arm of the robot holds a bag, while the right arm clamps the zipper of the bag. The robot is required to coordinate the movements of both arms to zip up the bag completely. This task is defined as a dual-arm fine-grained task.

## E. Additional Sensitivity Analysis

### E.1. Sensitivity to Imperfect Initial/Goal Images

Under imperfect initial or goal specifications, we observe that the anchor-based modes of GRM (i.e., forward- and backward-anchor) exhibit higher sensitivity. In contrast, the incremental mode demonstrates notable robustness, as it conditions solely on adjacent pre- and post-states, remaining largely decoupled from the absolute initial or goal states (e.g., accuracy drops are more pronounced for anchor modes,  $0.96 \rightarrow 0.87$ , compared to the incremental mode,  $0.95 \rightarrow 0.93$ ). Consequently, a straightforward adaptation strategy during deployments with ambiguous, imperfect, or unavailable goals and targets is to simply upweight the incremental mode.

### E.2. Sensitivity to Suboptimal Demonstrations

We evaluate how suboptimal or noisy demonstrations affect our framework. Conceptually, the single demonstration provided serves merely as a semantic anchor for scale alignment; GRM leverages generalizable motion priors rather than overfitting to the specific provided trajectory. Empirically, tests with suboptimal demonstrations show marginal degradation in reward accuracy ( $\Delta \text{Acc} < 3\%$ ). When integrated with RL, such imperfect demonstrations primarily lead to slightly slower convergence speeds rather than inducing suboptimal learned policies.

## F. More Qualitative Results

In this section, we provide additional qualitative visualizations to further substantiate the effectiveness and robustness

of our method. We focus on three key aspects: the generalization of GRM across diverse semantic tasks, the temporal robustness of progress estimation under varying sampling intervals, and the trajectory visualization of real-world RL.

**GRM Predictions on Diverse Tasks.** Figure 9 illustrates the predicted Hop and Progress curves generated by our GRM across a variety of manipulation tasks. The results demonstrate that our model accurately captures the monotonic increase in progress for successful trajectories while effectively identifying stagnation or regression in failed attempts, validating its semantic generalization capabilities.

**Robustness to Temporal Intervals.** A robust reward model should remain consistent regardless of the video sampling rate. Figure 10 compares the GRM’s progress estimation when inputs are sampled at different intervals ( $\Delta t = 10, 25, 50, 100$  frames). Despite the significant variation in visual disparity between adjacent frames, our Hop-based formulation ensures that the reconstructed global progress remains consistent, highlighting the model’s ability to decouple physical progress from temporal duration.

**Real-World RL Rollout Visualization.** Finally, Figure 11 visualizes the robustness of the policy learned for the “Insert Block” task. The policy used in this rollout was trained for approximately 20 minutes and achieved a success rate of over 95%. To evaluate the policy’s reactivity and the reward model’s accuracy, we introduced an artificial disturbance during execution. As shown in the sequence, a human operator manually moves the target slot while the robot is in motion (a). This intervention causes the robot to miss the target and fall into misalignment (b). Crucially, the inset plots show that our Generative Reward Model (GRM) immediately reflects this setback: the estimated *Progress* drops sharply, correctly identifying that the state has regressed from the goal. This accurate negative feedback guides the agent to adjust its trajectory. The robot successfully recovers from the misalignment (c), repositions itself above the target (d), aligns with the slot (e), and completes the insertion (f). This shows that GRM provides dense, semantically meaningful rewards that enable the agent to recover from unexpected external perturbations.

## G. Future Work

In future research, we plan to expand the capabilities and efficiency of Robo-Dopamine in four primary directions:

- **Efficient GRM Inference:** The current VLM-based reward model, while accurate, incurs high computational latency which can bottleneck online RL training loops. We aim to investigate low-precision quantization techniques (*e.g.*, INT4/INT8 quantization or KV-cache compression) to significantly accelerate GRM inference and reduce memory footprint without compromising reward accuracy for RL phase.
- **Continuous Video Stream Reasoning:** We aim to further evolve the GRM from discrete frame-pair inference to continuous video stream reasoning. By incorporating historical context sequences and integrating temporal modeling architectures (*e.g.*, Video Transformers, Temporal CNNs, or Video-LLM [3, 59]), the model will gain the ability to capture dynamic motion trends, such as inertia and trajectory, which are essential for high-dynamic tasks. Furthermore, this temporal continuity resolves state ambiguities inherent in static snapshots (*e.g.*, distinguishing between “grasping” and “releasing” an object, or tracking cumulative progress in cyclic tasks like scrubbing a test tube where multiple repetitions are required), thereby significantly enhancing the robustness of reward assessment.
- **Large-Scale Generalization:** We plan to scale the Dopamine-RL framework to a broader range of tasks in both simulation and the real world. Specifically, we aim to validate the framework on highly dynamic tasks (*e.g.*, throwing, catching) and long-horizon mobile manipulation scenarios to test the limits of the policy-invariant reward shaping mechanism.
- **Multi-Modal Reward Modeling:** While vision provides rich global context, fine-grained manipulation often requires non-visual cues. We intend to incorporate tactile and auditory modalities into the GRM. Tactile feedback is crucial for contact-rich tasks (*e.g.*, sensing force during insertion), while audio can detect discrete events (*e.g.*, the “click” of a latch), enabling more precise reward shaping for delicate operations.

## User Prompt for General Reward Model (GRM)

"" You are a rigorous, impartial vision evaluator for robot task progress. Your job is to judge whether the AFTER image set moves closer to the task objective than the BEFORE image set, using the provided reference examples only as anchors.

<Task>

*{task\_description}*

REFERENCE EXAMPLES (for visual anchoring only; not necessarily this run's actual START/END):

- REFERENCE START — Robot Front Image (task just starting): *<image>*
- REFERENCE END — Robot Front Image (task fully completed): *<image>*

</Task>

BEFORE Robot Front Image: *<image>*

BEFORE Robot Left Wrist Image: *<image>*

BEFORE Robot Right Wrist Image: *<image>*

AFTER Robot Front Image: *<image>*

AFTER Robot Left Wrist Image: *<image>*

AFTER Robot Right Wrist Image: *<image>*

### ### Goal

Compare the BEFORE and AFTER three-view sets and judge whether AFTER moves closer to accomplishing the task than BEFORE, using the REFERENCE START/END images as conceptual anchors.

Progress Estimation (no formulas)

1) Calibrate using the references:

- REFERENCE START = “just beginning”; REFERENCE END = “fully completed.”
- Visually estimate how far BEFORE and AFTER are along this START→END continuum.

2) Direction:

- AFTER better than BEFORE → positive score.
- AFTER worse than BEFORE → negative score.
- Essentially the same → 0.

3) Normalize to an integer percentage in [-100%, +100%]:

- For improvements, scale the improvement relative to what remained from BEFORE to END.
- For regressions, scale the deterioration relative to how far BEFORE had progressed from START.
- Clip to [-100%, +100%] and round to the nearest integer percent.

### ### Evaluation Criteria (apply across all three views)

1) Task Alignment: Evidence directly tied to *{task\_description}*.

2) Completeness & Accuracy: Correct pose, contact, placement, orientation, grasp quality, absence of collisions, stability, etc.

3) View-Specific Evidence & Consistency:

- Use the **Front** view for global layout, object pose, approach path, end-state geometry, and scene-level constraints.
- Use the **Left/Right Wrist** views to inspect **fine-grained gripper state** (finger closure, contact location/area, slippage, wedge/misalignment, object deformation, cable/wire/cloth entanglement, unintended contact, occluded collisions).
- When views disagree, prioritize the view that provides **decisive cues** for the criterion at hand. In particular, wrist views often **override** for grasp/contact validity and safety.
- If any single view shows a failure that invalidates success (e.g., mis-grasp, collision, unsafe/unstable pose), let that override when judging progress.

4) Ignore Irrelevant Factors: Lighting, color shifts, background clutter, or UI/watermarks that don't affect task success.

5) Ambiguity: If evidence is genuinely inconclusive or conflicting without decisive cues, treat progress as unchanged → 0%.

### ### Output Format (STRICT)

Return ONLY one line containing the score wrapped in <score> tags, as an integer percentage with a percent sign:

<score>+NN%</score> or <score>-NN%</score> or <score>0%</score>

""

Figure 8. User Prompt for General Reward Model (GRM).



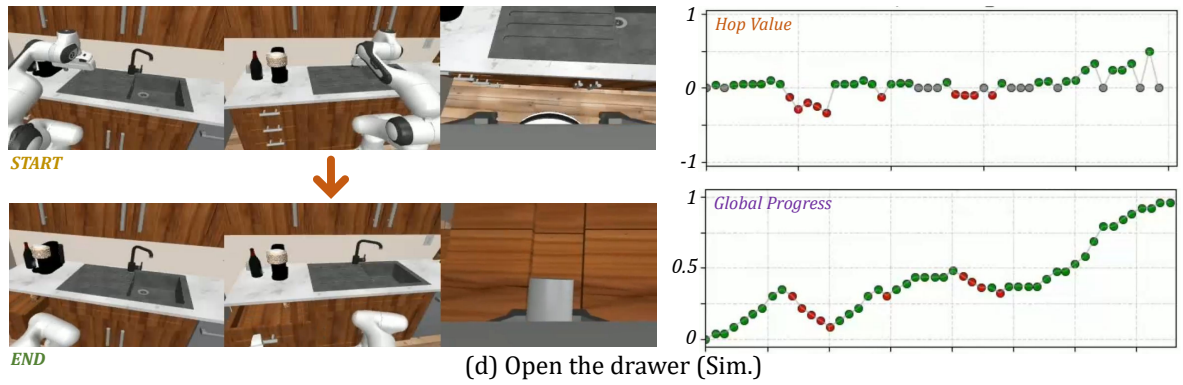
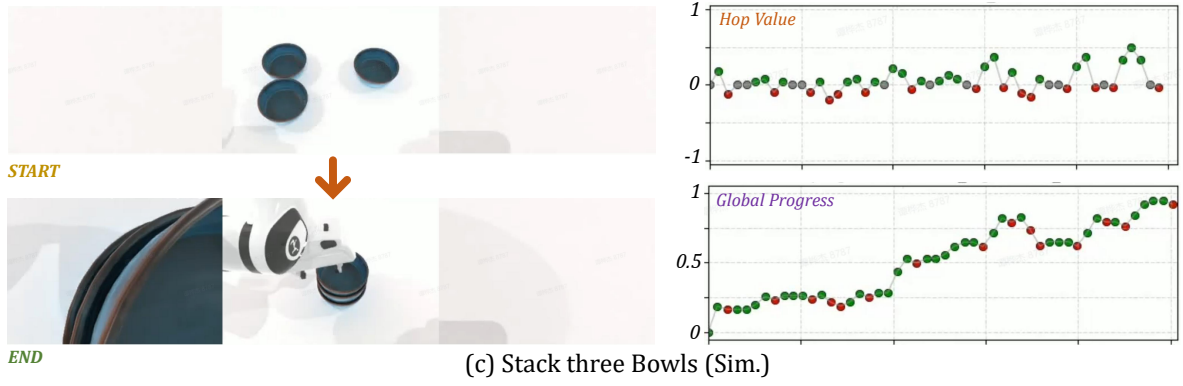
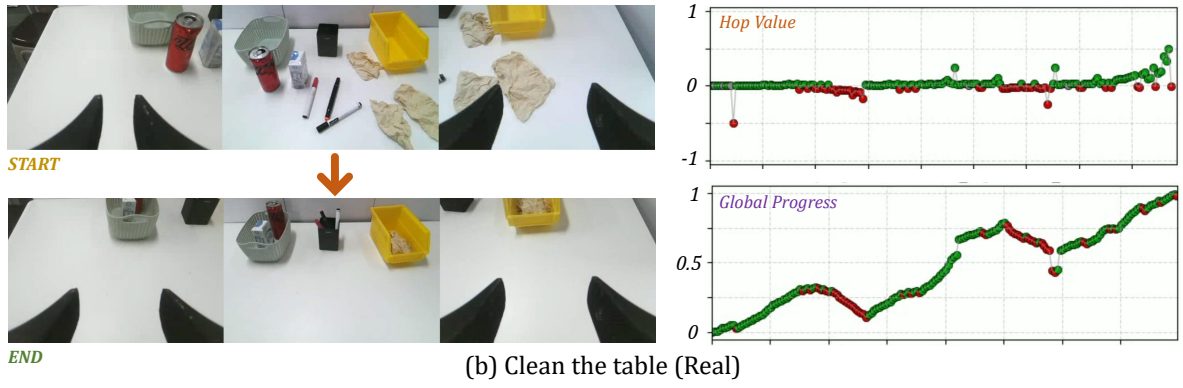
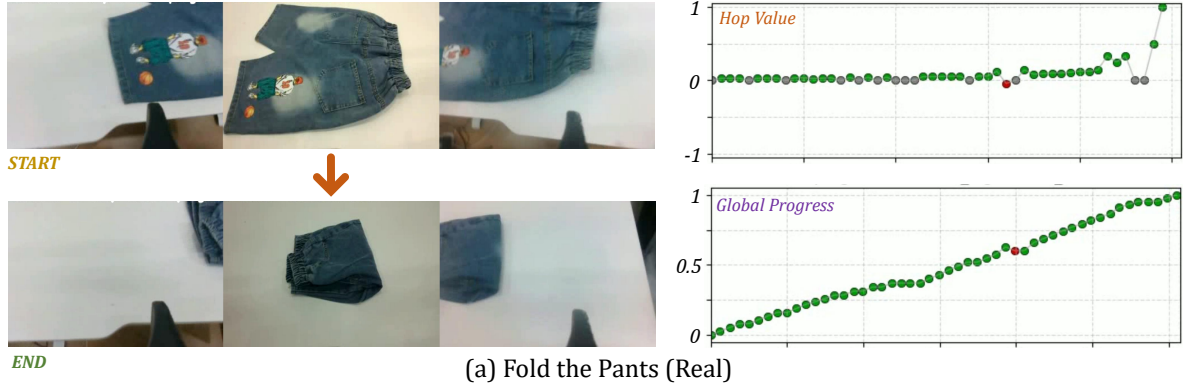


Figure 9. **GRM Progress Predictions across Diverse Tasks.** We visualize the frame-wise *Hop* (instantaneous change) and accumulated *Progress* predicted by GRM on unseen validation tasks.

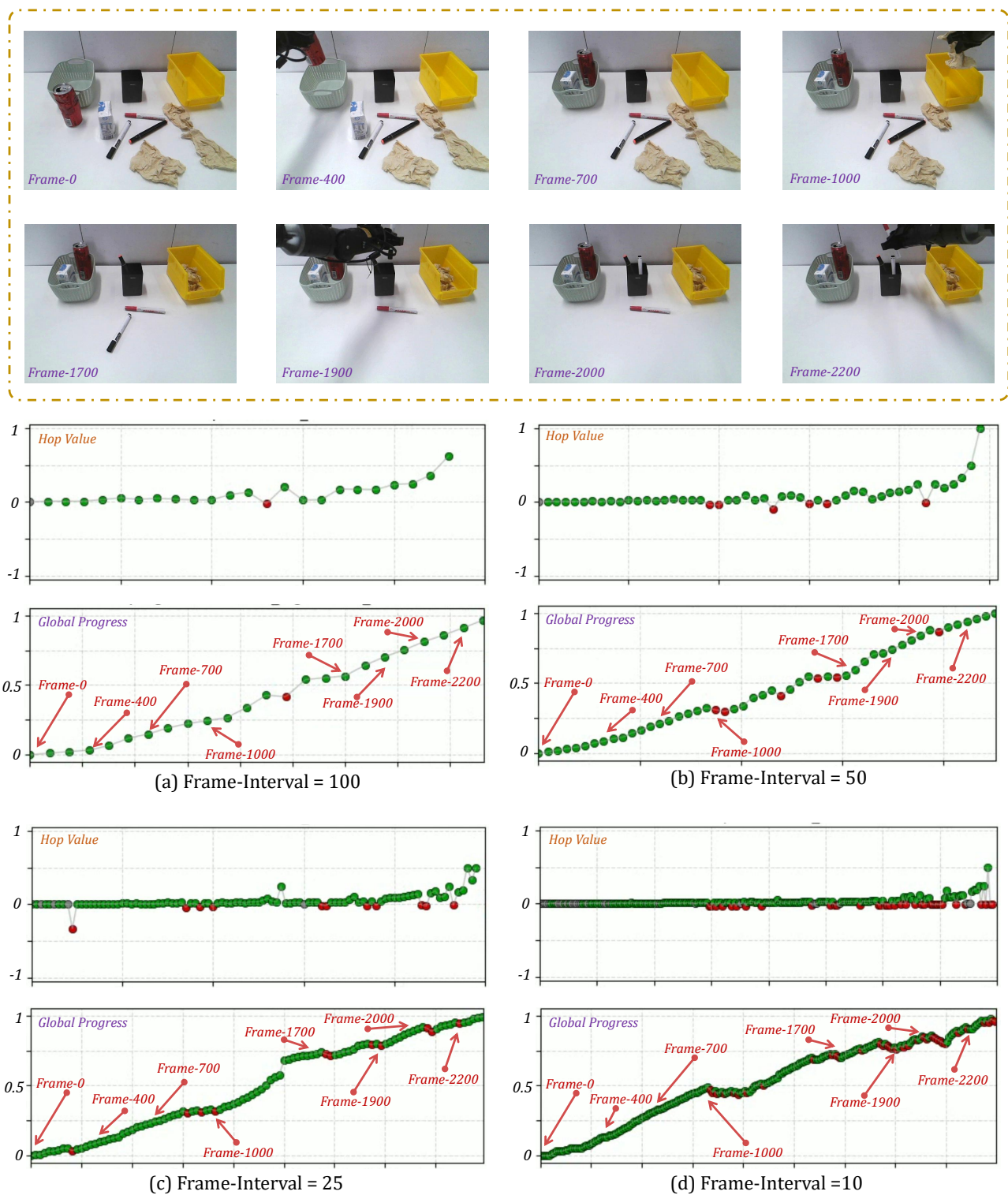


Figure 10. **Progress Estimation Consistency across Sampling Intervals.** We plot the reconstructed progress curves for the same trajectory using different frame strides (10, 25, 50, and 100 frames). The high overlap between curves demonstrates that our GRM is robust to temporal granularity and does not simply overfit to a specific frame rate.

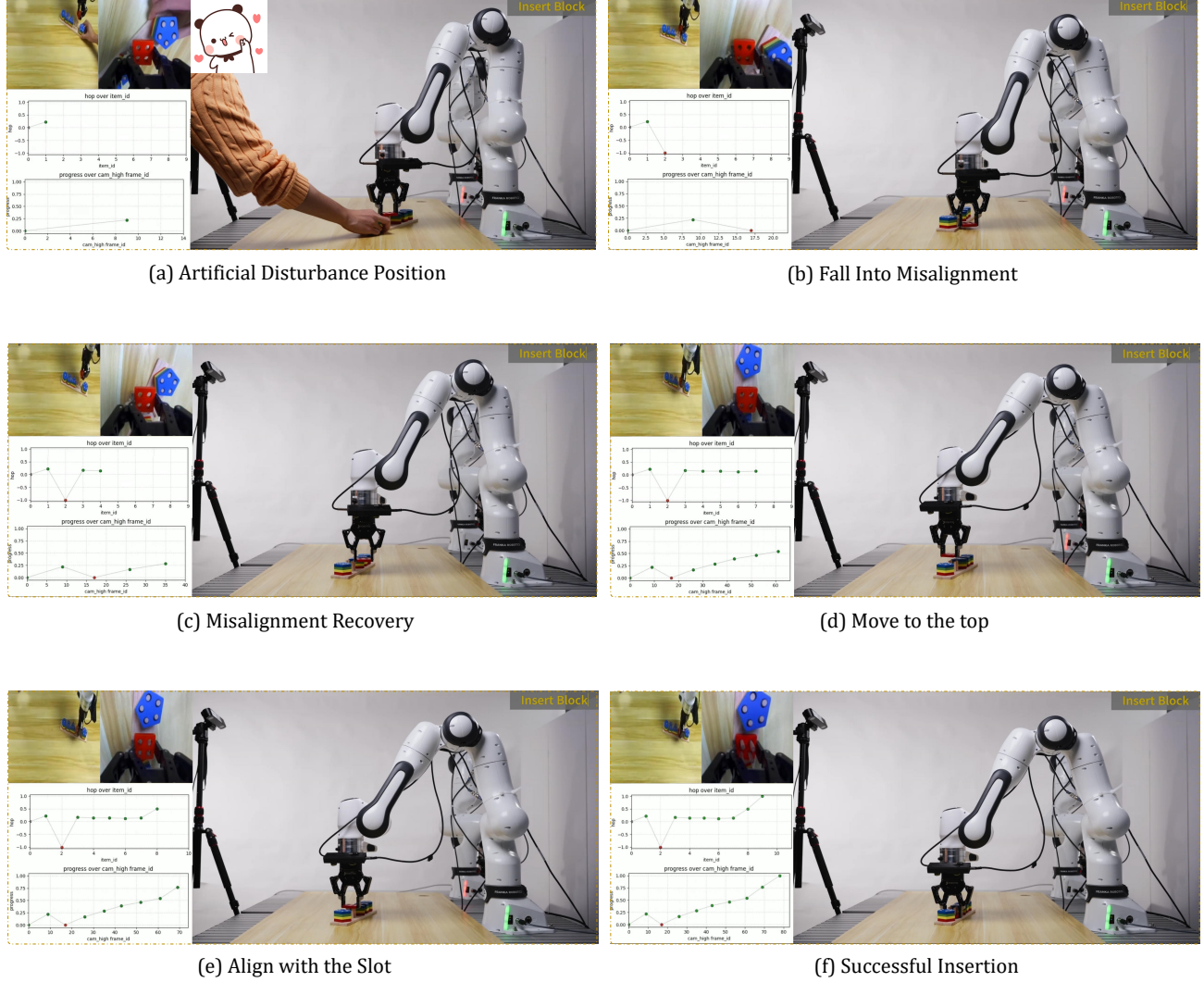


Figure 11. **Robustness to Artificial Disturbance during Real-World Execution.** We visualize a rollout of the converged policy (success rate  $> 95\%$ ) under human interference. Each sub-figure shows the third-person view, the ego-centric view, and the real-time GRM inference (Top: *Hop*, Bottom: *Progress*). **(a) Artificial Disturbance Position:** A human hand intervenes and shifts the target board while the robot attempts to approach. **(b) Fall Into Misalignment:** The robot misses the new position. Note that the GRM *Progress* curve drops significantly (indicated by the red dot in the bottom inset), reflecting the failure state. **(c) Misalignment Recovery:** The policy reacts to the visual feedback and the drop in reward, adjusting the end-effector position. **(d) Move to the top:** The robot realigns directly above the target slot. **(e) Align with the Slot:** Precise fine-tuning before insertion. **(f) Successful Insertion:** The task is completed, with the progress estimation reaching its peak.