SLOPE: Optimistic Potential Landscape Shaping for Model-based Reinforcement Learning

Yao-Hui Li[1*] **Zeyu Wang**[1*] **Xin Li**[1†] **Wei Pang**[2] **Yingfang Yuan**[2,3]
Zhengkun Chen[1] **Boya Zhang**[4] **Riashat Islam**[5] **Alex Lamb**[6] **Yonggang Zhang**[7]

[1]Beijing Institute of Technology, Beijing, China [2]Heriot-Watt University, Edinburgh, U.K.
[3]Northumbria University, Newcastle upon Tyne, U.K.
[4]Shenzhen Institutes of Advanced Technology, CAS, Shenzhen, China
[5]Microsoft Research NY, USA [6]Tsinghua University, Beijing, China
[7]Key Lab. of Symbolic Computation and Knowledge Engineering of MOE, Jilin University, China

Abstract

Model-based reinforcement learning (MBRL) is sample-efficient but struggles in sparse reward settings. A critical bottleneck arises from the lack of informative gradients in sparse settings, where standard reward models often yield flat landscapes that struggle to guide planning. To address this challenge, we propose **S**haping **L**andscapes with **O**ptimistic **P**otential **E**stimates (**SLOPE**), a novel framework that shifts reward modeling from predicting sparse scalars to constructing informative potential landscapes. SLOPE employs optimistic distributional regression to estimate high-confidence upper bounds, which amplifies rare success signals and ensures sufficient exploration gradients. Evaluations on **30+ tasks across 5 benchmarks** and **real-world robotic deployments**, demonstrate that SLOPE consistently outperforms leading baselines in **fully sparse**, **semi-sparse**, and **dense** rewards. The project code is available at https://github.com/Arya87/SLOPE.

1 Introduction

Model-based reinforcement learning (MBRL) uses learned dynamics and reward models to simulate future trajectories and evaluate policies, relying on dense reward signals to provide fine-grained feedback for planning [1, 2, 3]. In MBRL, such rewards are typically approximated by a learned reward model to guide policy optimization. In practice, such dense and well-shaped rewards are rarely available, while sparse binary signals indicating task success or failure are much more common in real-world environments. This mismatch limits the ability of MBRL to exploit its planning capabilities, as sparse rewards provide insufficient guidance for policy improvement. Therefore, addressing this limitation is critical for fully realizing the potential of MBRL in real-world.

As illustrated in Fig.1, developing reward models in MBRL under sparse rewards presents two key challenges: i) **Reward learning failure under extreme data imbalance.** In sparse reward settings, the vast majority of trajectories yield zero rewards. This scarcity of successful samples makes it difficult for the reward model to extract meaningful success patterns, often causing the model to collapse toward the majority

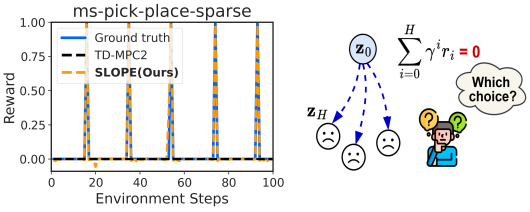


Figure 1: Key challenges in sparse-reward MBRL.

*Equal contribution.
†Correspondence to: Xin Li <xinli@bit.edu.cn>

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

class (zero) [4]. ii) **Ineffectiveness of ground-truth rewards for planning.** Most MBRL approaches [1, 2, 5] focus on accurately regressing ground-truth scalar rewards. However, in sparse settings, strictly fitting these values results in a reward model that outputs zeros almost everywhere, even with perfect accuracy. This creates a gradient-free landscape during rollouts, offering the planner no intermediate signals to guide the agent toward the goal.

To address the aforementioned challenges, recent studies have sought to incorporate expert demonstrations to mitigate the exploration dilemma in sparse reward tasks. MoDem series [6, 7] introduced a multi-phase training framework that effectively leverages demonstration data to accelerate the training of visual MBRL algorithms. However, this approach focuses solely on enhancing the training strategy and still adheres to regressing ground-truth sparse rewards. Meanwhile, inverse RL [8, 9], on the other hand, infers the reward function from expert demonstrations. However, in complex tasks, such approaches typically require a large amount of demonstration data to sufficiently cover the state-action space. Reward shaping techniques that incorporate intrinsic motivation, such as maximum entropy-based [10, 11], count-based [12, 13], or similarity-based [14, 15] approaches, can guide agents to discover meaningful behaviors in sparse reward environments. DEMO^3 [16] artificially decomposed the sparse reward into multi-stage semi-sparse reward, assigning higher reshaped rewards to states in the later stages. Nevertheless, heuristic shaping relies on delicate, task-specific engineering and often misaligns with the true objective. This misalignment makes agents prone to *reward hacking*, severely compromising the reliability of trajectory evaluation in sampling-based planners. Motivated by these limitations, a fundamental question arises.

How can we construct a dense, informative guiding signal for MBRL in sparse reward settings that provides reliable directional feedback for trajectory planning, while remaining aligned with the true task objective?

In this paper, we propose **SLOPE** (**S**haping **L**andscapes with **O**ptimistic **P**otential **E**stimates), a novel framework that integrates Potential-Based Reward Shaping (PBRS) into MBRL to tackle the fundamental challenge of gradient starvation in sparse-reward environments. Unlike standard MBRS algorithms that regress sparse scalar rewards, SLOPE constructs a dense internal potential landscape to provide continuous directional guidance for planning. Specifically, it treats the agent's learned Q-value function as a potential surface over the state space. This mechanism transforms the originally flat, gradient-free reward structure into a dense landscape, encouraging the agent to explore states with higher estimated potentials and providing directional reward signals during rollouts.

Although PBRS offers a dense potential construction, the effectiveness of the constructed landscape depends on the accuracy of the potential estimates. In sparse reward tasks, infrequent and delayed feedback often leads to underestimation of Q-values and slow convergence [17], resulting in a flat landscape that fails to facilitate exploration. To address this limitation, SLOPE employs optimism-driven landscape shaping. Unlike conventional methods that approximate the expected Q-value (mean), this mechanism leverages distributional value modeling to estimate a high-confidence upper bound. By amplifying the impact of rare but informative high-reward trajectories, the proposed mechanism ensures the shaped landscape possesses sufficient gradients to accelerate policy learning.

The versatility and robustness of SLOPE are underscored by an extensive empirical evaluation across **5** major benchmarks, encompassing over **30** diverse manipulation and motion tasks. By integrating SLOPE with diverse MBRL backbones (e.g., TD-MPC2 [2], Dreamerv3 [5]) and evaluating across **varying reward densities** (sparse, semi-sparse, and dense reward), we demonstrate its consistent superiority over traditional scalar regression baselines. Beyond simulated environments, we further validate SLOPE through **real-world robotic** experiments, where the challenges of sparse rewards are inherently prevalent and critical. This comprehensive cross-backbone validation and real-world deployment establish SLOPE as a powerful, general-purpose enhancement for modern MBRLs.

2 Related Work

2.1 MBRL with Sparse-Reward

MBRLs improve sample efficiency by leveraging learned environment dynamics for planning. However, sparse reward settings introduce significant challenges, primarily due to imbalanced training data and the lack of informative gradients during rollouts. Recent approaches like MoDem [6] and MoDem-v2 [7] mitigate the exploration dilemma by leveraging demonstration data through a

Table 1: Comparison of MBRL algorithmic features under sparse-reward settings. We contrast the reward modeling paradigms and supervision types of SLOPE against major baselines, highlighting how each method transforms or utilizes sparse environmental signals.

Method	Reward Shaping	Demonstrations	Learning Target	Reward Supervision
Dreamer-v3 [5]	✗	✗	Ground-truth reward	Sparse
TD-MPC2 [2]	✗	✗	Ground-truth reward	Sparse
MoDem [6]	✗	✓	Ground-truth reward	Sparse
DreamSmooth [4]	✓	✗	Smoothed reward	Dense + Sparse
DEMO^3 [16]	✓	✓	Stage-wise reward	Semi-sparse
SLOPE (Ours)	✓	✓	**Potential-based reward**	**Sparse**

multi-stage training framework. While effective in pre-training, these methods generally adhere to the standard paradigm of regressing ground-truth scalar rewards. Consequently, even with improved initialization, the learned reward model often yields a flat, gradient-free landscape in sparse regions, failing to provide directional guidance for planning. Other works, such as DEMO^3 [16], rely on manually designed stage-wise rewards, which limits applicability. DreamSmooth [4] tackles long-horizon sparse-reward tasks by predicting temporally smoothed rewards. LS-Imagine [18] achieves strong performance in episodic sparse-reward tasks by constructing a long-short term integrated world model augmented with intrinsic rewards. In contrast, our proposed SLOPE framework fundamentally shifts from predicting scalars to constructing an optimistic potential landscape. By employing optimistic distributional regression, SLOPE ensures sufficient planning gradients even in sparse settings without requiring manual reward engineering. Detailed comparisons are presented in Table 1.

2.2 Reward Shaping and Exploration

PBRS [19] reshapes the reward and value landscape to provide additional feedback under sparse rewards while preserving optimal policies for fixed, bounded state potentials. This invariance alone does not guarantee improved sample efficiency. Wiewiora [20] relates fixed-potential PBRS to Q-value initialization without specifying online potential learning. Later work studies dynamic potentials [21], foundation-model value priors [22], and potentials derived from learned values [23]. Early model-based work also integrates PBRS [24].

Related exploration strategies include entropy regularization [10, 11], visitation bonuses [12, 13, 25], and similarity-based signals [14, 15, 26]. RND [27] uses random-feature prediction errors, whereas Plan2Explore [28] uses dynamics-ensemble disagreement. Such novelty and disagreement signals need not directly reflect task progress. POLO [29] constructs optimistic terminal values from a randomized-prior ensemble for MPC exploration, without using them as PBRS supervision for a learned reward model. DOIE [30] uses state–action knownness to maintain an optimistic value envelope and incorporates domain knowledge to guide continuous-control exploration.

SLOPE constructs a self-evolving potential from a slow-moving target critic for sparse-reward visual MBRL, treating the potential as fixed within each planning and optimization phase. Unlike POLO's epistemic optimism, its upper-quantile weighting emphasizes rare high-return targets amid frequent failures. Potential differences provide dense reward-model supervision for guiding imagined trajectories, without hand-designed potentials or predefined subgoals.

3 Preliminaries

This section explains the basic notation in modeling RL problems and the TD-MPC2 framework, a MBRL algorithm on which our work is built. We then analyze the theoretical limitations of standard planning methods in sparse-reward tasks, which motivate the development of SLOPE.

3.1 Problem Formulation

We consider learning in a standard Markov Decision Process (MDP) formulated by a tuple of $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, r, \mathcal{P}, \gamma \rangle$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, and r is a sparse reward function. This means that the rewards are typically binary, taking a value of $\{0, 1\}$ where a reward of 1 is received only upon successful completion of the task, and 0 otherwise. $\mathcal{P}\left(\mathbf{s}_{t+1} \,|\mathbf{s}_t, \mathbf{a}_t\right)$ is

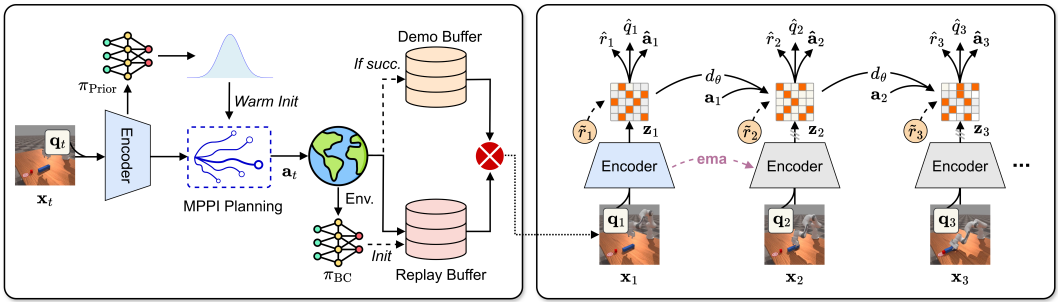


Figure 2: Framework of SLOPE. Building upon MoDem's multi-phase accelerated learning framework, we also introduce two training enhancements: (i) initializing MPPI sampling distribution from the prior policy π_{Prior}, and (ii) augmenting the demonstration buffer with successful trajectories. The shaped reward $\tilde{r}$ is used for training both the reward model R_θ and the Q value function Q_θ. The environment input consists of multimodal observations $\mathbf{o} = (\mathbf{x}, \mathbf{q})$, where $\mathbf{x}$ denotes raw RGB images captured by the robot's camera, and $\mathbf{q}$ represents the robot's proprioceptive sensory information, e.g., the gripper state. Subsequent observations are encoded by the target encoder (illustrated in grey) which is updated via an exponential moving average (EMA) of the online encoder.

the state transition function, and $\gamma \in [0, 1)$ is the discount factor to trade off immediate and future rewards. The goal of RL is to find an optimal policy π^* to maximize the expected cumulative return, $\pi^* = \arg\max_\pi \mathbb{E}_{\mathbf{a}_t \sim \pi(\cdot|\mathbf{s}_t), \mathbf{s}_t \sim \mathcal{P}} \left[\sum_{t=1}^{T} \gamma^t r\left(\mathbf{s}_t, \mathbf{a}_t\right) \right]$, starting from an initial state $\mathbf{s}_0 \in \mathcal{S}$ and following a policy $\pi_\theta\left(\cdot|\mathbf{s}_t\right)$ that is parameterized by a set of learnable parameters θ.

3.2 TD-MPC2

We build our method on **TD-MPC2** [2], a scalable MBRL algorithm that extends the TD-MPC [1] framework. In this work, we focus on the single-task formulation. TD-MPC2 learns a latent-space world model comprising five core components: an encoder $h_\theta(\mathbf{o})$ mapping observations to latent states $\mathbf{z}$; a dynamics model $d_\theta(\mathbf{z}, \mathbf{a})$ predicting future states $\mathbf{z}'$; a reward model $R_\theta(\mathbf{z}, \mathbf{a})$; a value function $Q_\theta(\mathbf{z}, \mathbf{a})$; and a prior policy $\pi_\theta(\mathbf{z})$.

The value function Q_θ is trained via the Bellman operator $\mathcal{T}$:

$$\mathcal{T}Q(\mathbf{z}, \mathbf{a}) = R(\mathbf{z}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{z}', \mathbf{a}'}[Q(\mathbf{z}', \mathbf{a}')] \tag{1}$$

During inference, TD-MPC2 performs planning via Model Predictive Path Integral (MPPI) [31]. It iteratively samples action sequences from a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ and updates the distribution parameters using the top-k trajectories to maximize the expected cumulative return $J(\tau)$:

$$\mu^*, \sigma^* = \arg\max_{(\mu,\sigma)} \mathbb{E}_{\mathbf{a}_{t:t+H} \sim \mathcal{N}(\mu, \sigma^2)} \left[J(\tau) \right], \tag{2}$$

where $J(\tau) = \sum_{h=t}^{H-1} \gamma^h R(\mathbf{z}_h, \mathbf{a}_h) + \gamma^H Q(\mathbf{z}_{t+H}, \mathbf{a}_{t+H})$.

The Challenge of Sparse Rewards. Standard MBRLs are based on the assumption that R_θ and Q_θ provide dense signals to guide MPPI optimization. However, in sparse reward tasks, this assumption is broken down.

Consider a scenario in which the goal is unreachable within the horizon H. With ground truth rewards $r_t = 0$ for all sampled trajectories, the learned model collapses to zero and Q_θ fails to differentiate between the superior and inferior states. TD-MPC2 employs a variant of MPPI that updates the sampling distribution parameters using only the top-k trajectories with the highest estimated returns. The mean μ is updated at iteration j as:

$$\mu^{j+1} = \frac{\sum_{i=1}^{k} \Omega_i \mathbf{a}_i^*}{\sum_{i=1}^{k} \Omega_i}, \quad \text{where } \Omega_i = e^{\kappa \cdot J(\tau_i^*)}. \tag{3}$$

Here, $\{\mathbf{a}_i^*\}_{i=1}^k$ represents the action sequences of the top-k elites sorted by their cumulative return $J(\tau)$, and κ is a temperature parameter. In a sparse setting where $J(\tau) \approx 0$ across all N sampled

trajectories, the top-k selection degenerates into a random subset of the original noise. Furthermore, the importance weights become uniform. Consequently, the update μ^{j+1} essentially averages a random selection of noise, preventing the planner from shifting the search distribution towards the goal. This theoretical limitation motivates our shift from fitting sparse scalar rewards to learning a potential landscape.

4 Methodology

In this section, we introduce **SLOPE**, a specialized framework that extends PBRS through optimism-driven landscape construction. Rather than relying on standard scalar reward regression, SLOPE formulates a dense potential landscape to provide continuous guidance in model-based planning. The method comprises three core components: **Model-Based Planning via Potential Landscape**, which leverages PBRS guarantees to embed directional guidance directly into the planner's objective; **Optimism-Driven Landscape Shaping**, which approximates a quantile-driven optimistic distributional Bellman operator via asymmetric weighting to capture upper-quantile returns and drive exploration; and **Accelerated Training Strategies** to ensure stable and efficient convergence.

4.1 Model-Based Planning via Potential Landscape

Standard MBRL methods are based on planning over imagined trajectories by maximizing the sum of predicted scalar rewards. However, in sparse reward settings, this reliance on immediate scalar feedback creates a fundamental gradient starvation problem. As illustrated in the 10×10 GridWorld example (Fig.3, Left), a standard reward model effectively predicts zero everywhere outside the goal state. This topological flatness means R_θ provides negligible gradients, depriving the latent encoder and dynamics model of crucial representation learning signals. Even with a terminal value term (Eq. 2), the planner's overall objective landscape remains uninformative, depriving sampling-based planners like MPPI of the directional information necessary to distinguish between stagnant states and progressive paths.

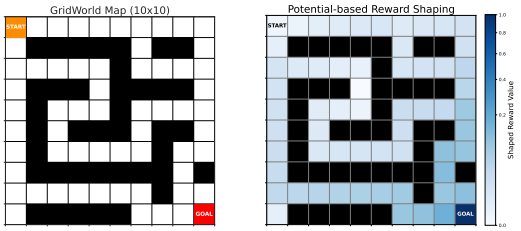


Figure 3: Toy example on a 10×10 GridWorld. **Left**: The original environment with sparse rewards ($+1$ at goal, 0 elsewhere). **Right**: The dense reward signal generated by PBRS using converged optimal value function V^*. ■: wall.

To overcome this structural bottleneck, our approach transitions reward modeling from predicting sparse rewards to constructing a dense potential landscape. Instead of learning a flat R_θ, we formulate a reshaped reward model $\widetilde{R}_\theta(\mathbf{z}, \mathbf{a})$ based on PBRS [19]:

Theorem 4.1 (Potential-Based Reward Shaping [19])**.** *Given the MDP $\mathcal{M}$ with optimal policy π^*, the reshaped MDP $\widetilde{\mathcal{M}}$ with the reward function defined as:*

$$\widetilde{r}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim \mathcal{P}} \left[\Phi\left(\mathbf{s}'\right)\right] - \Phi(\mathbf{s}) \tag{4}$$

preserves the optimal policy, i.e., $\widetilde{\pi}^* = \pi^*$, for any bounded potential function $\Phi \colon \mathcal{S} \to \mathbb{R}$. Furthermore, the optimal value functions satisfy:

$$\widetilde{V}^*(\mathbf{s}) = V^*(\mathbf{s}) - \Phi(\mathbf{s}), \quad \widetilde{Q}^*(\mathbf{s}, \mathbf{a}) = Q^*(\mathbf{s}, \mathbf{a}) - \Phi(\mathbf{s}). \tag{5}$$

The topological quality of this reshaped landscape heavily depends on the choice of Φ. Ideally, as shown in Fig. 3 (Right), setting Φ to the true optimal value function V^* provides perfect directional guidance. By replacing the sparse target r with such a dense potential difference, the reward model is forced to encode local value gradients at *every single time step*. By predicting this local trend of the potential landscape instead of merely sparse scalar values, the reward model learns to capture the trend of the potential landscape, providing the planner with dense, step-by-step directional guidance.

Since the optimal value is unknown a priori, we employ a self-evolving landscape strategy [23]. To stabilize bootstrapping and reduce fluctuations in the shaping signal, we construct the potential from the agent's slow-moving target value network:

$$\Phi_k(\mathbf{s}) = \eta \max_{\mathbf{a}} Q_{\bar{\theta}_k}(\mathbf{s}, \mathbf{a}), \tag{6}$$

where η is a scaling factor and $\bar{\theta}_k$ denotes the target parameters at iteration k, updated across iterations via exponential moving average (EMA). The resulting potential differences provide transition-level supervision for the learned reward model, enabling dense guidance throughout imagined rollouts.

Although the potential evolves across training iterations, the policy-invariance guarantee applies separately to each snapshot in which the target parameters are held fixed during the corresponding planning and optimization phase. Under this fixed-potential assumption, the following corollary follows directly from Theorem 4.1.

Corollary 4.2 (Per-Iteration Policy Invariance)**.** *Consider a discounted MDP $\mathcal{M}$ with $\gamma \in [0, 1)$. At iteration k, let*

$$\Phi_k(\mathbf{s}) = \eta \max_{\mathbf{a}} Q_{\bar{\theta}_k}(\mathbf{s}, \mathbf{a})$$

be a bounded state potential, with the target parameters $\bar{\theta}_k$ held fixed during the corresponding planning and optimization phase. Let $\widetilde{\mathcal{M}}_k$ have the same states, actions, transition dynamics, and discount factor as $\mathcal{M}$, but use the shaped reward

$$\widetilde{r}_k(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim \mathcal{P}(\cdot|\mathbf{s},\mathbf{a})} \left[\Phi_k(\mathbf{s}')\right] - \Phi_k(\mathbf{s}).$$

Then,

$$\Pi^*(\widetilde{\mathcal{M}}_k) = \Pi^*(\mathcal{M}), \qquad \forall k,$$

where $\Pi^*(\cdot)$ denotes the set of optimal policies. Thus, although the potential evolves across iterations, each fixed-potential snapshot preserves the optimal-policy set of the original MDP.

Proof. For each fixed iteration k, Φ_k is a bounded state potential, so the result follows directly from Theorem 4.1. □

4.2 Optimism-Driven Landscape Shaping

Although PBRS offers a mechanism for dense feedback, its efficacy depends on the quality of the value estimates. Standard regression techniques focusing on the expectation frequently suppress rare success signals in sparse reward tasks, yielding a flat potential landscape.

To address this, we employ an **Optimism-Driven Value Learning** mechanism to actively shape the potential landscape. By introducing an optimistic bias, this approach amplifies the impact of rare successful trajectories against the background of failure. Consequently, the constructed potential landscape retains informative gradients pointing towards high-reward regions.

We implement this mechanism via **Quantile-weighted Cross-Entropy (QCE)** loss. Unlike standard distributional approaches (e.g., TD-MPC2) that use symmetric objectives, the QCE loss imposes an asymmetric penalty on prediction errors. We parameterize the action-value function $Q_\theta(\mathbf{s}, \mathbf{a})$ as a categorical distribution over discrete support bins, where the scalar Q-value is obtained via the expectation $\mathbb{E}[Q_\theta(\mathbf{s}, \mathbf{a})]$.

Formally, the regression target y is first computed using the standard scalar expectation of the target network:

$$y = \widetilde{r}(\mathbf{s}, \mathbf{a}) + \gamma \, \mathbb{E}_{\mathbf{s}' \sim \mathcal{P}} \left[\max_{\mathbf{a}'} \mathbb{E}[Q_{\bar{\theta}}(\mathbf{s}', \mathbf{a}')] \right]. \tag{7}$$

Following TD-MPC2 [2], we project the scalar target y into a distributional representation $\Psi(y)$ using soft two-hot encoding. This technique distributes probability mass across the fixed discrete support bins via linear interpolation. To enforce optimism, we define the QCE objective as:

$$\mathcal{L}_{\text{QCE}}(\theta) = \mathbb{E}_{\mathcal{D}} \left[w(\delta) \cdot \text{CE} \left(\Psi(y), Q_\theta(\mathbf{s}, \mathbf{a}) \right) \right]. \tag{8}$$

Here, $\text{CE}(\cdot, \cdot)$ denotes the Cross-Entropy loss, and $w(\delta)$ is a dynamic weighting term based on the scalar estimation error $\delta = \mathbb{E}[Q_\theta(\mathbf{s}, \mathbf{a})] - y$:

$$w(\delta) = \tau \cdot \mathbb{1}_{\{\delta < 0\}} + (1 - \tau) \cdot \mathbb{1}_{\{\delta \geq 0\}}, \tag{9}$$

where $\tau \in (0.5, 1]$ represents the quantile coefficient. By setting $\tau > 0.5$, the objective penalizes underestimation ($\delta < 0$) more heavily than overestimation. This asymmetry drives the learned value distribution towards the upper quantiles of the return distribution, effectively encouraging exploration in uncertain but potentially high-reward regions.

To theoretically guarantee the stability of this optimistic update, we formalize it as dynamic reward shaping and prove its convergence properties below.

Proposition 4.3 (Contraction Mapping under Target-Anchored Shaping)**.** *Consider the reshaped MDP $\widetilde{\mathcal{M}} = \langle \mathcal{S}, \mathcal{A}, \widetilde{r}, \mathcal{P}, \gamma \rangle$ with the reward dynamically modified by the target-anchored potential function. Let $\Phi(\mathbf{s}) = \eta \max_{\mathbf{a}} Q_{\bar{\theta}}(\mathbf{s}, \mathbf{a})$, where $Q_{\bar{\theta}}$ is a target network held fixed during the current planning and optimization step. The reshaped reward is defined as:*

$$\widetilde{r}_{\Phi}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim \mathcal{P}}[\Phi(\mathbf{s}')] - \Phi(\mathbf{s})$$

For any scaling factor $\eta > 0$, the Bellman Operator $\mathcal{T}_{\Phi}$ applied to this target-anchored reshaped MDP is a strict γ-contraction in the L_{∞} norm.

Proof. See Appendix A.1 for detailed proof. □

Proposition 4.4 (Consistent Policy Improvement)**.** *Let π_{k+1} be derived greedily from the reshaped landscape, i.e., $\pi_{k+1}(\mathbf{s}) = \arg\max_{\mathbf{a}} Q_k^{\pi_k}(\mathbf{s}, \mathbf{a})$. Then, this update monotonically improves the policy not only in the reshaped MDP $\widetilde{\mathcal{M}}_k$ but also with respect to the original objective $\mathcal{M}$, such that $Q_{\mathcal{M}}^{\pi_{k+1}}(\mathbf{s}, \mathbf{a}) \geq Q_{\mathcal{M}}^{\pi_k}(\mathbf{s}, \mathbf{a})$.*

Proof. See Appendix A.2 for detailed proof. □

4.3 Accelerating MBRL with Sparse Rewards

In the early stage of training, a limited amount of high-quality demonstration data is crucial for efficient policy exploration and model learning, while it can also be leveraged to initialize an informative potential landscape. Building upon previous work MoDem [6], which accelerates MBRL with limited demonstrations, we adopt its three-phase training framework. Additionally, we also introduce two training enhancements: (1) initializing the MPPI sampling distribution with a prior policy in place of the random distribution to accelerate planning; and (2) augmenting the demonstration buffer with successful trajectories. To mitigate the sample imbalance in sparse-reward tasks, successful trajectories encountered during training is actively incorporated into the demo buffer.

5 Experiments

We evaluate our method on five continuous control benchmarks and additionally validate it in real-world sparse-reward robotic tasks (see Fig.4). Our suite includes sparse-reward tasks from *ManiSkill3* [32] (5 tasks), *Meta-World* [33] (10 tasks, categorized as `medium`, `hard`, and `very hard` by [34]), *Robosuite* [35] (3 tasks), and *Adroit* [36] (2 tasks), alongside dense-reward tasks from *DeepMind Control Suite* [37]. Through these experiments, we aim to investigate the following key questions: (1) *Can our proposed method improve MBRL algorithms under sparse reward settings?* (2) *Can SLOPE learn discriminative reward signals that accurately reflect task progress to better guide policy learning?* (3) *How important are the individual components of SLOPE?* (4) *How does SLOPE compare with intrinsic-motivation and disagreement-based exploration methods?*

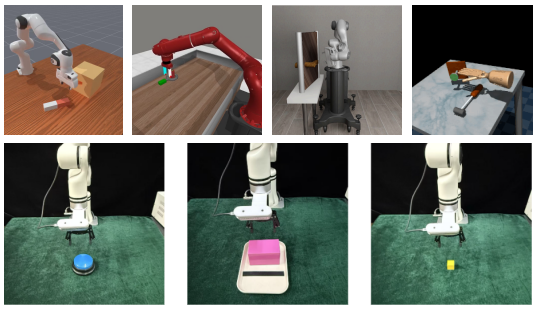


Figure 4: Visualizations of representative tasks. **Top**: Simulation benchmarks including *ManiSkill3* (`ms`), *Meta-World* (`mw`), *Robosuite*, and *Adroit*. **Bottom**: Real-world tasks including Press Button, Push Cube, and Grasp Cube.

5.1 Experimental Settings

Baselines. To evaluate the effectiveness of SLOPE, we compare it against several closely related MBRL methods: (1) **TD-MPC2** [2], a strong and widely adopted baseline known for its scalability and multi-task performance. We focus on its effectiveness under sparse reward settings. (2) **MoDem** [6], a demonstration-augmented MBRL method with a three-phase training pipeline. Originally built on TD-MPC, we reimplement it with TD-MPC2 for fair comparison. MoDem achieves SOTA results on sparse-reward benchmarks like *Meta-World* and *Adroit*, highlighting its sample efficiency from

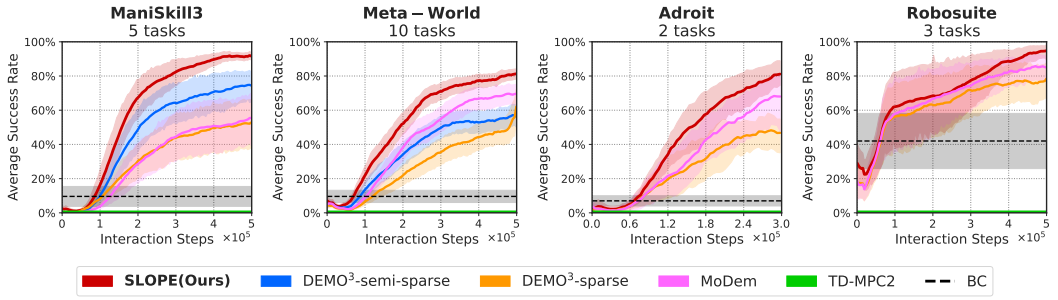


Figure 5: Average success rates across 20 tasks from 4 benchmarks. Curves and shaded areas represent the mean and 95% confidence intervals (CIs) over 5 independent runs. We include both sparse and semi-sparse tasks for *ManiSkill3* and *Meta-World*, while *RoboSuite* and *Adroit* involve sparse rewards only, following their native task designs. See Appendix C.3 for individual task details.

Table 2: Comparison of methods on real-world tasks at 15k environment steps ($\approx$ 1 hour of real-world training). Results are averaged over 20 evaluation trials. Success Rate and Intervention Rate are reported in percentage (%). The best results are highlighted in **colorbox**.

Method	**Press Button**		**Push Cube**		**Grasp Cube**	
	Success (↑)	Intervention (↓)	Success (↑)	Intervention (↓)	Success (↑)	Intervention (↓)
SAC	45.0	16.8	0.0	46.5	0.0	45.3
MoDem	65.0	10.0	15.0	18.4	35.0	24.0
DEMO^3	**100.0**	11.1	15.0	19.1	20.0	23.2
SLOPE (Ours)	**100.0**	**7.3**	**65.0**	**18.2**	**65.0**	**21.3**

expert demonstrations. (3) **DEMO^3** [16], An MBRL method that reformulates sparse-reward tasks into manually defined multi-stage sub-tasks with denser intermediate rewards (semi-sparse). DEMO^3 has reported SOTA results on complex manipulation domains such as *ManiSkill3* and *Robosuite*. However, some tasks are inherently single stage or cannot be decomposed into multiple stages. We evaluate DEMO^3 under both semi-sparse (DEMO^3-semi-sparse) and fully sparse (DEMO^3-sparse) settings to assess its adaptability to different reward structures. (4) **Behavior Cloning** (BC), a supervised imitation learning baseline that directly learns policies from expert demonstrations.

We also implement three exploration baselines on the same MoDem framework with a TD-MPC2 backbone: (1) **Latent RND** [27], whose intrinsic reward is the prediction error of a trainable network predicting the output of a fixed randomly initialized network from the next-state latent representation; (2) **Q-ensemble disagreement (QD)**, which uses the standard deviation of scalar predictions from TD-MPC2's five target Q-heads as an exploration bonus, capturing value disagreement rather than dynamics disagreement; and (3) **Plan2Explore-style disagreement (P2E)** [28], which uses disagreement among five auxiliary one-step latent-dynamics predictors trained with bootstrap masks. For each baseline, the normalized bonus is added to the task reward for reward-model and value learning. All variants use the same demonstrations, backbone, interaction budget, and evaluation protocol. We compare them on Pick Place and Peg Insertion, two challenging sparse-reward ManiSkill3 tasks on which MoDem alone performs poorly.

Reward Settings. In this work, we consider: (1) **sparse** rewards, where $r \in \{0, 1\}$ is given only upon task completion; (2) **semi-sparse** rewards, formulated as a stage indicator $r : S \to \{1, \ldots, N\}$; and (3) **dense** rewards, included as a supplementary benchmark.

Real Robot Settings. Real-robot experiments are conducted using a Realman RM75-B robotic arm equipped with two RGB cameras to capture image observations. Following the Human-in-the-Loop Sample Efficient Robotic Reinforcement Learning (HIL-SERL) pipeline [38], we train SLOPE entirely from scratch in the real world without any simulation pre-training. We evaluate our method on three sparse-reward tasks: `Press Button`, `Push Cube`, and `Grasp Cube`. The tasks are illustrated in Fig.4. Additional real-robot implementation details are available in the Appendix B.4.

5.2 Simulation Experiments Results

As shown in Fig.5, the pure MBRL baseline TD-MPC2 fails across all tasks under sparse reward settings. While the BC policy achieves decent initial performance, it is limited by the amount of available expert data and struggles to improve further. In contrast, SLOPE demonstrates superior performance across all four benchmarks using only sparse rewards. On *ManiSkill3*, it significantly outperforms the previous SOTA, DEMO^3-semi-sparse. Unlike DEMO^3-sparse, which suffers a notable drop without handcrafted stage priors, SLOPE consistently surpasses competing methods (including MoDem and DEMO^3 variants) on *Meta-World*, *Adroit*, and *Robosuite*. These results verify SLOPE's ability to learn effectively without dense rewards or stage-wise priors.

5.3 Real-World Experiments Results

Table 2 demonstrates that SLOPE consistently outperforms baselines in both success and efficiency. On the simple `Press Button` task, SLOPE matches DEMO^3 at 100.0% success but with significantly lower intervention. In complex tasks, SLOPE attains 65.0% success, surpassing baselines significantly. Notably, on `Grasp Cube`, SLOPE significantly exceeds both MoDem (35.0%) and DEMO^3 (20.0%) while maintaining the lowest intervention rate, whereas SAC fails completely. Collectively, these results validate SLOPE's robustness and sample efficiency in solving complex sparse-reward manipulation problems with minimal human intervention.

5.4 Comparison with Other Exploration Baselines

Fig. 6 compares the success rate learning curves over 5×10^5 interaction steps. On `Pick Place`, RND and P2E substantially improve over MoDem, but require considerably more interactions to achieve high success rates than SLOPE. QD provides only a modest and unstable improvement. SLOPE reaches a high success rate early in training and maintains it throughout the remaining budget, demonstrating stronger sample efficiency on this task.

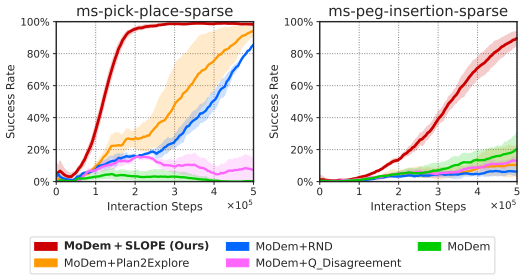


Figure 6: Comparison with intrinsic-motivation and disagreement-based exploration baselines on sparse-reward *ManiSkill3* tasks. All variants share the MoDem backbone and demonstrations. Compared with RND, Q-ensemble disagreement, and Plan2Explore-style disagreement, SLOPE learns faster on Pick Place and achieves substantially higher final success on Peg Insertion.

The pattern differs on `Peg Insertion`, where all three intrinsic baselines underperform MoDem, whereas SLOPE achieves approximately 90% success rate. `Peg Insertion` requires precise alignment and carefully controlled contact interactions, leaving only a narrow range of successful behaviors. In this setting, visitation novelty or disagreement may prioritize under-visited or difficult-to-predict transitions that do not reflect meaningful insertion progress, thereby steering the planner away from the demonstration-supported success region.

These results indicate that generic novelty and uncertainty bonuses can be effective when broad coverage is sufficient, as in `Pick Place`, but may become counterproductive in precision-sensitive tasks. In contrast, SLOPE derives its guidance from learned task values and yields the strongest performance on both tasks.

5.5 Further Analysis

Ablation Study. To better understand the contribution of each component in SLOPE, we conduct an ablation study comparing the full model against two key variants: (1) **SLOPE w/o ATS**, which removes the Accelerated Training Strategies to isolate the effect of our core contributions, and (2) **SLOPE w/o PL&ODS**, which strips away the Potential Landscape and Optimism-Driven Shaping mechanisms, relying solely on the auxiliary training strategies. The results are shown in Fig. 7 (Left).

Removing potential landscape and optimism-driven shaping mechanisms leads to a severe performance degradation across all tasks, confirming that our core potential-based shaping mechanism is the

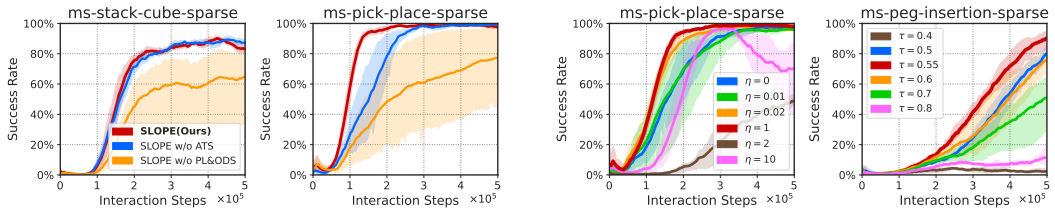


Figure 7: **Left**: Ablation study. **Right**: Hyperparameter sensitivity analysis of SLOPE.

primary driver for efficient learning under sparse rewards. Conversely, disabling accelerated training strategies only result in a marginal performance decrease, maintaining highly competitive success rates across most tasks. This strongly indicates that our core potential landscape and optimism-driven shaping mechanisms are the primary drivers of the overall performance improvements. Nevertheless, we observe that integrating the auxiliary accelerated training strategies still provide practical benefits by further accelerating convergence and stabilizing the training process.

Hyperparameter Sensitivity. We evaluate the sensitivity to η and τ in Fig. 7 (Right). For the shaping weight, we find $\eta = 1.0$ to be the optimal value for providing effective directional guidance. However, performance degrades if η continues to increase, as an overly strong shaping signal tends to overshadow the original task reward. Regarding τ, moderate optimism at $\tau = 0.55$ yields the best exploration-exploitation trade-off compared to the unbiased baseline ($\tau = 0.5$). Similarly, excessively high τ values cause performance drops, as extreme optimism severely biases the value distribution and reduces the reliability of the shaping potential.

Visualization and Analysis To address Question (2), we visualize the reward predictions of different algorithms along a successful trajectory in Fig.8. It can be observed that the reward models of MoDem and DEMO^3-sparse produce relatively flat predictions, failing to capture the underlying task structure. In contrast, SLOPE demonstrates a strong correlation with the ground truth reward. Crucially, even in the absence of dense supervision, SLOPE's predicted reward exhibits a clear upward trend that aligns with the temporal progression of task completion. This indicates that our method successfully learns a discriminative reward model capable of providing dense, meaningful guidance within sparse-reward tasks.

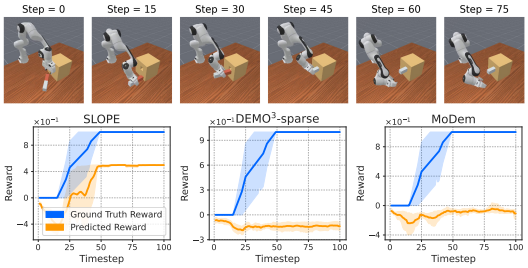


Figure 8: Visualization of reward predictions. **Top:** Keyframes of task execution. **Bottom:** Ground truth vs. predicted rewards along the trajectory.

See Appendix C for comprehensive evaluations, including performance on dense reward tasks, transferability to other MBRL algorithms, and further analysis.

6 Conclusion

In this paper, we presented SLOPE, a novel framework that enhances MBRL in sparse-reward tasks by integrating optimistic potential landscape construction into the reward modeling. By utilizing the agent's value estimates as a self-evolving potential function, SLOPE provides dense directional guidance absent in sparse structures. SLOPE also incorporates a distributional value mechanism to generate optimistic upper bounds, effectively preventing the suppression of rare success signals during exploration. Extensive experiments across 5 challenging benchmarks and real-world tasks demonstrate that SLOPE consistently outperforms SOTA methods.

Limitations and Future Work. One potential limitation of SLOPE is the inherent tension between optimistic exploration and safety constraints, as aggressive potential scaling may inadvertently encourage high-risk behaviors in unvisited, hazardous regions. To address this, future work will extend our framework into Safety-Aware MBRL by incorporating safety-centric priors, such as Control Barrier Functions (CBF), to construct repulsive potential fields.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable feedback. This work was partially supported by the National Natural Science Foundation of China (Grant No. 62276024), the Beijing Natural Science Foundation (Grant No. 4262066), the Fundamental Research Funds for the Central Universities, Jilin University (Grant No. 93K172025K01), and the Fundamental Research Funds for the Central Universities (Grant No. 2025CX01010). This work was also supported by the Intelligent Computing Platform of Beijing Institute of Technology (BIT).

References

[1] Nicklas A Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8387–8406. PMLR, 17–23 Jul 2022.

[2] Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024.

[3] Yuhang Wang, Hanwei Guo, Sizhe Wang, Long Qian, and Xuguang Lan. Bootstrapped model predictive control. In *The Thirteenth International Conference on Learning Representations*, 2025.

[4] Vint Lee, Pieter Abbeel, and Youngwoon Lee. Dreamsmooth: Improving model-based reinforcement learning via reward smoothing. In *The Twelfth International Conference on Learning Representations*, 2024.

[5] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, pages 1–7, 2025.

[6] Nicklas Hansen, Yixin Lin, Hao Su, Xiaolong Wang, Vikash Kumar, and Aravind Rajeswaran. Modem: Accelerating visual model-based reinforcement learning with demonstrations. In *The Eleventh International Conference on Learning Representations*, 2023.

[7] Patrick Lancaster, Nicklas Hansen, Aravind Rajeswaran, and Vikash Kumar. Modem-v2: Visuomotor world models for real-world robot manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7530–7537, 2024.

[8] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

[9] Sateesh Kumar, Jonathan Zamora, Nicklas Hansen, Rishabh Jangir, and Xiaolong Wang. Graph inverse reinforcement learning from diverse videos. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 55–66. PMLR, 14–18 Dec 2023.

[10] Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. State entropy maximization with random encoders for efficient exploration. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9443–9454. PMLR, 18–24 Jul 2021.

[11] Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18459–18473. Curran Associates, Inc., 2021.

[12] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[13] Aditya Ramesh, Louis Kirsch, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Exploring through random curiosity with general value functions. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 18733–18748. Curran Associates, Inc., 2022.

[14] Nikolay Savinov, Anton Raichuk, Damien Vincent, Raphael Marinier, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. In *International Conference on Learning Representations*, 2019.

[15] Mikael Henaff, Minqi Jiang, and Roberta Raileanu. A study of global and episodic bonuses for exploration in contextual MDPs. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 12972–12999. PMLR, 23–29 Jul 2023.

[16] Adrià López Escoriza, Nicklas Hansen, Stone Tao, Tongzhou Mu, and Hao Su. Multi-stage manipulation with demonstration-augmented reward, policy, and world model learning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 15542–15563. PMLR, 13–19 Jul 2025.

[17] 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, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[18] Jiajian Li, Qi Wang, Yunbo Wang, Xin Jin, Yang Li, Wenjun Zeng, and Xiaokang Yang. Open-world reinforcement learning over long short-term imagination. *arXiv preprint arXiv:2410.03618*, 2024.

[19] 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, page 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[20] Eric Wiewiora. Potential-based shaping and q-value initialization are equivalent. *J. Artif. Int. Res.*, 19(1):205–208, September 2003.

[21] Sam Devlin and Daniel Kudenko. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, page 433–440, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[22] Weirui Ye, Yunsheng Zhang, Haoyang Weng, Xianfan Gu, Shengjie Wang, Tong Zhang, Mengchen Wang, Pieter Abbeel, and Yang Gao. Reinforcement learning with foundation priors: Let the embodied agent efficiently learn on its own. *arXiv preprint arXiv:2310.02635*, 2023.

[23] Jacob Adamczyk, Volodymyr Makarenko, Stas Tiomkin, and Rahul V. Kulkarni. Bootstrapped reward shaping. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(15):15302–15310, Apr. 2025.

[24] John Asmuth, Michael L. Littman, and Robert Zinkov. Potential-based shaping in model-based reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, page 604–609. AAAI Press, 2008.

[25] Jesse Mu, Victor Zhong, Roberta Raileanu, Minqi Jiang, Noah Goodman, Tim Rocktäschel, and Edward Grefenstette. Improving intrinsic exploration with language abstractions. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.

[26] Yuhua Jiang, Qihan Liu, Yiqin Yang, Xiaoteng Ma, Dianyu Zhong, Hao Hu, Jun Yang, Bin Liang, Bo Xu, Chongjie Zhang, et al. Episodic novelty through temporal distance. *arXiv preprint arXiv:2501.15418*, 2025.

[27] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

[28] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8583–8592. PMLR, 13–18 Jul 2020.

[29] Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. In *International Conference on Learning Representations*, 2019.

[30] Sam Lobel, Omer Gottesman, Cameron Allen, Akhil Bagaria, and George Konidaris. Optimistic initialization for exploration in continuous control. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence*, pages 7612–7619, 2022.

[31] Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling, 2015.

[32] Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Viswesh Nagaswamy Rajesh, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *Robotics: Science and Systems*, 2025.

[33] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1094–1100. PMLR, 30 Oct–01 Nov 2020.

[34] Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1332–1344. PMLR, 14–18 Dec 2023.

[35] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, Yifeng Zhu, and Kevin Lin. robosuite: A modular simulation framework and benchmark for robot learning, 2020.

[36] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[37] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

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

[39] Jiajian Li, Qi Wang, Yunbo Wang, Xin Jin, Yang Li, Wenjun Zeng, and Xiaokang Yang. Open-world reinforcement learning over long short-term imagination. In *The Thirteenth International Conference on Learning Representations*, 2025.

[40] Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation. *arXiv preprint arXiv:2403.10506*, 2024.

A Proof

A.1 Proof of Proposition 4.3: Contraction Mapping

Lemma A.1 (Non-expansiveness of the Max Operator)**.** *Let $Q(\mathbf{s}, \mathbf{a})$ be a scalar value function. The max operator over the action space is non-expansive in the L_∞ norm. Specifically, for any two scalar value functions Q_1 and Q_2:*

$$\left\| \max_{\mathbf{a}} Q_1(\cdot, \mathbf{a}) - \max_{\mathbf{a}} Q_2(\cdot, \mathbf{a}) \right\|_\infty \leq \|Q_1 - Q_2\|_\infty.$$

Proof. For any state $\mathbf{s}$, let $\mathbf{a}_1^* = \arg\max_{\mathbf{a}} Q_1(\mathbf{s}, \mathbf{a})$ and $\mathbf{a}_2^* = \arg\max_{\mathbf{a}} Q_2(\mathbf{s}, \mathbf{a})$. Without loss of generality, assume $Q_1(\mathbf{s}, \mathbf{a}_1^*) \geq Q_2(\mathbf{s}, \mathbf{a}_2^*)$. Then:

$$\begin{aligned}
\max_{\mathbf{a}} Q_1(\mathbf{s}, \mathbf{a}) - \max_{\mathbf{a}} Q_2(\mathbf{s}, \mathbf{a}) &= Q_1(\mathbf{s}, \mathbf{a}_1^*) - Q_2(\mathbf{s}, \mathbf{a}_2^*) \\
&\leq Q_1(\mathbf{s}, \mathbf{a}_1^*) - Q_2(\mathbf{s}, \mathbf{a}_1^*) \\
&\leq \max_{\mathbf{a}} |Q_1(\mathbf{s}, \mathbf{a}) - Q_2(\mathbf{s}, \mathbf{a})| \\
&= \|Q_1 - Q_2\|_\infty.
\end{aligned}$$

The case where $Q_2(\mathbf{s}, \mathbf{a}_2^*) > Q_1(\mathbf{s}, \mathbf{a}_1^*)$ follows by symmetry. Thus, the operation is non-expansive.

□

Proposition 4.3 (Contraction Mapping under Target-Anchored Shaping)**.** *Consider the reshaped MDP $\widetilde{\mathcal{M}} = \langle \mathcal{S}, \mathcal{A}, \widetilde{r}, \mathcal{P}, \gamma \rangle$ with the reward dynamically modified by the target-anchored potential function. Let $\Phi(\mathbf{s}) = \eta \max_{\mathbf{a}} Q_{\bar{\theta}}(\mathbf{s}, \mathbf{a})$, where $Q_{\bar{\theta}}$ is a target network held fixed during the current planning and optimization step. The reshaped reward is defined as:*

$$\widetilde{r}_\Phi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim \mathcal{P}}[\Phi(\mathbf{s}')] - \Phi(\mathbf{s})$$

For any scaling factor $\eta > 0$, the Bellman Operator $\mathcal{T}_\Phi$ applied to this target-anchored reshaped MDP is a strict γ-contraction in the L_∞ norm.

Proof. Let Q_1 and Q_2 be two arbitrary scalar online Q-functions being optimized. The target-anchored Bellman operator $\mathcal{T}_\Phi$ is defined as:

$$\mathcal{T}_\Phi Q_i(\mathbf{s}, \mathbf{a}) = \widetilde{r}_\Phi(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim \mathcal{P}} \left[\max_{\mathbf{a}'} Q_i(\mathbf{s}', \mathbf{a}') \right] \quad \text{for } i \in \{1, 2\}$$

Step 1: Independence of the Reshaped Reward. Crucially, because the potential function Φ is derived exclusively from the target network $Q_{\bar{\theta}}$ (which is held fixed during the current optimization step), it is completely independent of the online networks Q_1 and Q_2. Consequently, the reshaped reward $\widetilde{r}_\Phi(\mathbf{s}, \mathbf{a})$ acts as a stationary scalar for both evaluations, yielding a zero difference:

$$\Delta \widetilde{r} = \widetilde{r}_\Phi(\mathbf{s}, \mathbf{a})\big|_{Q=Q_1} - \widetilde{r}_\Phi(\mathbf{s}, \mathbf{a})\big|_{Q=Q_2} = 0$$

This isolates the contraction analysis to the discounted future value terms.

Step 2: Bounding the Bellman Update Difference. We now evaluate the absolute difference between the Bellman updates for Q_1 and Q_2 at any state-action pair $(\mathbf{s}, \mathbf{a})$. By leveraging the linearity of expectation, the triangle inequality for integrals, and the non-expansiveness of the max operator (Lemma A.1), we establish the following continuous bound:

$$\begin{aligned}
|\mathcal{T}_\Phi Q_1(\mathbf{s}, \mathbf{a}) - \mathcal{T}_\Phi Q_2(\mathbf{s}, \mathbf{a})| &= \left| \left(\widetilde{r}_\Phi(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}'} \left[\max_{\mathbf{a}'} Q_1(\mathbf{s}', \mathbf{a}') \right] \right) \right. \\
&\qquad \left. - \left(\widetilde{r}_\Phi(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}'} \left[\max_{\mathbf{a}'} Q_2(\mathbf{s}', \mathbf{a}') \right] \right) \right| \\
&= \gamma \left| \mathbb{E}_{\mathbf{s}'} \left[\max_{\mathbf{a}'} Q_1(\mathbf{s}', \mathbf{a}') - \max_{\mathbf{a}'} Q_2(\mathbf{s}', \mathbf{a}') \right] \right| \\
&\leq \gamma \mathbb{E}_{\mathbf{s}'} \left[\left| \max_{\mathbf{a}'} Q_1(\mathbf{s}', \mathbf{a}') - \max_{\mathbf{a}'} Q_2(\mathbf{s}', \mathbf{a}') \right| \right] \\
&\leq \gamma \mathbb{E}_{\mathbf{s}'} \left[\|Q_1 - Q_2\|_\infty \right] \\
&= \gamma \|Q_1 - Q_2\|_\infty
\end{aligned}$$

Taking the supremum over all $(\mathbf{s}, \mathbf{a})$, we obtain the global contraction bound:

$$\|\mathcal{T}_\Phi Q_1 - \mathcal{T}_\Phi Q_2\|_\infty \leq \gamma \|Q_1 - Q_2\|_\infty$$

Since the discount factor $\gamma \in [0, 1)$, the operator $\mathcal{T}_\Phi$ is a strict γ-contraction mapping under the L_∞ norm, regardless of the scaling factor η. By Banach's Fixed-Point Theorem, repeated applications of $\mathcal{T}_\Phi$ will converge to a unique fixed point for the current target-anchored landscape.

□

A.2 Proof of Proposition 4.4: Consistent Policy Improvement

Proposition 4.4 (Consistent Policy Improvement)**.** *Let π_{k+1} be derived greedily from the reshaped landscape, i.e., $\pi_{k+1}(\mathbf{s}) = \arg\max_{\mathbf{a}} Q_k^{\pi_k}(\mathbf{s}, \mathbf{a})$. Then, this update monotonically improves the policy not only in the reshaped MDP $\widetilde{\mathcal{M}}_k$ but also with respect to the original objective $\mathcal{M}$, such that $Q_{\mathcal{M}}^{\pi_{k+1}}(\mathbf{s}, \mathbf{a}) \geq Q_{\mathcal{M}}^{\pi_k}(\mathbf{s}, \mathbf{a})$.*

Proof. Let $\mathcal{T}_k^{\pi_k}$ denote the Bellman evaluation operator for policy π_k in the reshaped MDP snapshot $\widetilde{\mathcal{M}}_k$. By definition, the action-value function $Q_k^{\pi_k}$ is the fixed point of this operator:

$$Q_k^{\pi_k}(\mathbf{s}, \mathbf{a}) = \mathcal{T}_k^{\pi_k} Q_k^{\pi_k}(\mathbf{s}, \mathbf{a}) = \widetilde{r}_k(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim \mathcal{P}} \left[Q_k^{\pi_k}(\mathbf{s}', \pi_k(\mathbf{s}')) \right]$$

Step 1: Local Policy Improvement in $\widetilde{\mathcal{M}}_k$. Consider a new policy π_{k+1} that is perfectly greedy with respect to $Q_k^{\pi_k}$. By the definition of the max operator, for any state $\mathbf{s}'$, we have:

$$Q_k^{\pi_k}(\mathbf{s}', \pi_{k+1}(\mathbf{s}')) = \max_{\mathbf{a}' \in \mathcal{A}} Q_k^{\pi_k}(\mathbf{s}', \mathbf{a}') \geq Q_k^{\pi_k}(\mathbf{s}', \pi_k(\mathbf{s}'))$$

Applying the Bellman evaluation operator $\mathcal{T}_k^{\pi_{k+1}}$ to $Q_k^{\pi_k}$ and leveraging the monotonicity of the expectation, we obtain:

$$\begin{aligned} \mathcal{T}_k^{\pi_{k+1}} Q_k^{\pi_k}(\mathbf{s}, \mathbf{a}) &= \widetilde{r}_k(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim \mathcal{P}} \left[Q_k^{\pi_k}(\mathbf{s}', \pi_{k+1}(\mathbf{s}')) \right] \\ &\geq \widetilde{r}_k(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim \mathcal{P}} \left[Q_k^{\pi_k}(\mathbf{s}', \pi_k(\mathbf{s}')) \right] = Q_k^{\pi_k}(\mathbf{s}, \mathbf{a}) \end{aligned}$$

By induction, repeated applications of the monotonic operator yield $(\mathcal{T}_k^{\pi_{k+1}})^n Q_k^{\pi_k} \geq Q_k^{\pi_k}$ for all $n \geq 1$. Since $\mathcal{T}_k^{\pi_{k+1}}$ is a strict γ-contraction in L_∞ norm within the stationary snapshot $\widetilde{\mathcal{M}}_k$, the sequence converges to the unique fixed point $Q_k^{\pi_{k+1}}$, establishing the improvement in the reshaped landscape:

$$Q_k^{\pi_{k+1}}(\mathbf{s}, \mathbf{a}) \geq Q_k^{\pi_k}(\mathbf{s}, \mathbf{a})$$

Step 2: Alignment with the Original Objective $\mathcal{M}$. To ensure this improvement is not a byproduct of shaping bias, we invoke the action-ordering invariance of PBRS. The relationship between the reshaped value and the original value is $Q_k^\pi(\mathbf{s}, \mathbf{a}) = Q_{\mathcal{M}}^\pi(\mathbf{s}, \mathbf{a}) - \Phi_k(\mathbf{s})$. In our framework, the potential function is instantiated as the state-value estimate from the target network:

$$\Phi_k(\mathbf{s}) = \eta \max_{\mathbf{a}' \in \mathcal{A}} Q_{\bar{\theta}_k}(\mathbf{s}, \mathbf{a}').$$

Crucially, although the underlying target network $Q_{\bar{\theta}_k}$ is a function of both state and action, the $\max$ operator identifies the supremum over the entire action manifold $\mathcal{A}$ for a given state $\mathbf{s}$. Consequently, the potential $\Phi_k(\mathbf{s})$ depends solely on the state topology and is strictly independent of the specific action $\mathbf{a}$ being evaluated in the current Bellman update.

The magnitude of value improvement between any two policies π_{k+1} and π_k is preserved exactly:

$$Q_k^{\pi_{k+1}}(\mathbf{s}, \mathbf{a}) - Q_k^{\pi_k}(\mathbf{s}, \mathbf{a}) = Q_{\mathcal{M}}^{\pi_{k+1}}(\mathbf{s}, \mathbf{a}) - Q_{\mathcal{M}}^{\pi_k}(\mathbf{s}, \mathbf{a})$$

Thus, the non-negative improvement proven in Step 1 directly implies $Q_{\mathcal{M}}^{\pi_{k+1}}(\mathbf{s}, \mathbf{a}) \geq Q_{\mathcal{M}}^{\pi_k}(\mathbf{s}, \mathbf{a})$.

□

B Implementation Details

B.1 Hyperparameters

The majority of hyperparameters in our experiments are inherited from baseline algorithm TD-MPC2 and DEMO^3, to ensure a fair comparison and reproducibility. For consistency, we retain the default settings for most optimization, model, and policy-related parameters as adopted in previous works. In this section, we highlight only the hyperparameters that are most pertinent to our proposed method, such as those related to reward shaping, quantile selection. A detailed listing of these key hyperparameters is provided in Table 3. For all remaining parameters and further implementation details, we refer readers to the original papers of TD-MPC2.

Table 3: SLOPE's hyperparameters.

Hyperparameter	Value
SLOPE	
Potential weight η	1.0
Quantile coefficient τ	0.55
TD-MPC2	
Planning horizon H	3
MPPI iterations	Adroit: 8, others: 6
Interaction steps	500,000
Replay buffer capacity	300,000
Sampling	Uniform
Demo. sampling ratio	50%
Eval episodes	10
Num of parallel envs	16
Steps per update	4
Reward prediction coef.	0.1
Value prediction coef.	0.1
Joint-embedding coef.	20
Temporal coef. λ	0.5
Latent state dim	512
Batch size	256
Optimizer	Adam
Learning rate	3×10^{-4}
Encoder learning rate	1×10^{-4}
Discount factor γ	0.95

B.2 Environment Details

In this section, we provide a comprehensive overview of the four benchmark environments utilized in our experiments: *ManiSkill3*, *Meta-World*, *Adroit*, *Robosuite*, and *Real World* experiments. The key characteristics and configurations of these environments are summarized in the following Table 4.

Table 4: Summary of environment details for *ManiSkill3*, *Meta-World*, *Adroit*, *Robosuite*, and *Real World* experiments. Prop.state stands for proprioceptive state of the robotic arm.

	ManiSkill3	Meta-World	Adroit	Robosuite	Real World
Time Horizon	100	100	100	100	Variable
Image Size	128×128	224×224	224×224	128×128	128×128
Observations	RGB×2 + Prop.state	RGB + Prop.state	RGB×2 + Prop.state	RGB×2	RGB×2 + Prop.state
Cameras	Hand + Front	Front	Front	Hand + Front	Hand + Side
Action Repeat	2	2	2	1	1
Action Dim	7	4	Door: 28, Hammer: 26	7	4

B.3 Demonstrations

The demonstration data for all simulation environments used in our experiments are collected by training a TD-MPC2 agent with access to dense rewards and full state observations. The real-world demonstration data was collected by humans using a SpaceMouse remote control. Each demonstration consists of a fixed-length trajectory generated by the expert policy. The specific number of demonstrations employed for each task is summarized in Table 5.

Table 5: Number of demonstration trajectories used for each task in our experiments.

Domain	Task	Number of Demonstrations
ManiSkill3	Peg Insertion	100
	Pick Place	100
	Stack Cube	25
	Poke Cube	5
	Lift Peg Upright	5
Meta-World	Assembly	5
	Peg Insert Side	5
	Stick Push	5
	Stick Pull	5
	Hand Insert	5
	Pick Out of Hole	5
	Pick Place Wall	5
	Push Back	5
	Push	5
	Shelf Place	5
Adroit	Door	5
	Hammer	5
Robosuite	Lift	5
	Door	10
	Stack Blocks	100
Real-World Experiments	Press Button	1
	Push Cube	1
	Grasp Cube	10

B.4 Real-Robot Implementation Details

Hardware Setup. As shown in Fig.9, we use a Realman RM75-B 7-DoF robotic arm equipped with a gripper. The visual perception system consists of two RGB cameras: one is mounted on the wrist (eye-in-hand) to provide egocentric views, while the other is fixed at the side to offer a third-person perspective. Both cameras capture RGB images, which are subsequently cropped to a resolution of 128×128.

Action Space and Controller. The policy outputs a 4-dimensional action vector $a_t \in \mathbb{R}^4$, consisting of the end-effector's relative displacement $(\Delta x, \Delta y, \Delta z)$ and the gripper status.

- **Remote Control:** We employ a 3Dconnexion SpaceMouse for teleoperation. The human operator controls the end-effector's translational velocity.
- **Low-level Control:** The target Cartesian pose is converted to joint positions using an analytical Inverse Kinematics (IK) solver. The commands are sent to the robot at a frequency of 10Hz, with a safety layer that clips joint velocities and confines the end-effector within a predefined workspace boundaries.

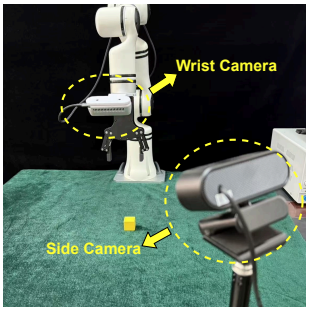


Figure 9: Real-robot setup.

- **Smoothed Action:** To ensure smooth robot motion and mitigate high-frequency jitter, we apply an Exponential Moving Average (EMA) filter to the policy's output actions. Let $a_t = [a_{t,\text{trans}}, a_{t,\text{grip}}]$ denote the raw action at time t, where $a_{t,\text{trans}} \in \mathbb{R}^3$ represents the translational displacement and $a_{t,\text{grip}} \in \mathbb{R}^1$ is the gripper command. The smoothed action $\bar{a}_t$ is computed as:

$$\bar{a}_{t,\text{trans}} = \alpha \cdot a_{t,\text{trans}} + (1-\alpha) \cdot \bar{a}_{t-1,\text{trans}}, \quad \bar{a}_{t,\text{grip}} = a_{t,\text{grip}}$$

where $\alpha \in [0,1]$ is the smoothing factor. This formulation dampens control noise in the trajectory while preserving the immediate responsiveness of gripper actuation.

Hybrid Action Space. The robotic manipulation task involves a hybrid action space $\mathcal{A} = \mathcal{A}_{\text{arm}} \times \mathcal{A}_{\text{grip}}$, where the arm motion $\mathcal{A}_{\text{arm}} \subseteq \mathbb{R}^3$ is continuous, but the gripper actuation $\mathcal{A}_{\text{grip}} \in \{-1, 1\}$ is discrete (binary). Standard MPPI assumes a fully continuous space with Gaussian noise, which is ill-suited for binary controls. We implement a specific strategy to bridge this gap:

- **Policy Discretization via STE:** To enable end-to-end training of the policy network π_θ despite the non-differentiable binary output, we utilize the Straight-Through Estimator (STE). During the forward pass, the gripper action is binarized using a hard threshold function $a_{\text{grip}} = \text{sign}(a_{\text{raw}})$. During the backward pass, gradients are propagated through the continuous a_{raw} (using the identity function for the derivative), effectively treating the threshold as a transparent layer.
- **Quantized MPPI Sampling:** During the planning phase, we modify the sampling procedure of the Model Predictive Path Integral (MPPI). While we sample candidate trajectories from a Gaussian distribution $\mathcal{N}(\mu, \Sigma)$, we enforce a hard quantization on the gripper dimension immediately after sampling:

$$a^{(i)}_{t,\text{grip}} \leftarrow \text{sign}(a^{(i)}_{t,\text{grip}}), \quad \forall i \in \{1, \dots, N\}$$

  This ensures that the latent dynamics model predicts future states conditional on valid discrete gripper commands, reducing prediction error caused by out-of-distribution continuous values (e.g., 0.0).
- **Distribution Projection:** When updating the sampling distribution parameters via the Cross-Entropy Method (CEM), we project the updated mean of the gripper dimension back to the binary manifold. Specifically, after computing the weighted mean μ_{new} from elite trajectories, we apply $\mu_{\text{new,grip}} \leftarrow \text{sign}(\mu_{\text{new,grip}})$. This prevents the sampling distribution from collapsing towards zero (ambiguous state) and maintains distinct modes for "open" and "close" actions throughout the planning horizon.

Task Settings. We design three sparse-reward manipulation tasks of increasing complexity to evaluate the efficacy of our method in real-world scenarios.

- `Press Button(Easy)`: The goal is to control the end-effector to press a specific button on the table. It is a *sparse-reward* task where the agent receives a success signal ($r = 1$) only when the button is fully depressed. This task requires precise end-effector positioning and force application.



- `Push Cube(Medium)`: The agent must push a target cube from a randomized starting position to a designated goal region marked on the table. The reward is binary, provided only when the object successfully enters the target zone. This task challenges the agent's ability to reason about object dynamics and contact physics.



- `Grasp Cube(Hard)`: The objective is to pick up a cube from the table. A success reward is granted only when the object is lifted to a predefined height threshold. This is the most challenging task as it involves precise grasping, stability control during lifting, and handling potential slippage.



Reset Mechanism. At the beginning of each episode, the robotic arm moves to a fixed reset pose. Subsequently, the task scene is manually restored by a human operator. To ensure the robustness of the learned policy, the operator randomizes the object's initial position and orientation within a predefined workspace during each reset.

Reward Function. Due to the absence of ground-truth state information in the real world, we rely on sparse reward signals to indicate task completion. Generally, there are two common approaches to obtain these signals: training a visual binary classifier to predict success probabilities, or employing manual human labeling. In this work, we adopt the latter approach, where a human operator verifies the task success status during policy execution.

Training Process. We adopt the HIL-SERL [38] paradigm for real-world training, implemented based on the official LeRobot codebase[3]. A key component of HIL-SERL is the active human intervention mechanism. During the agent's autonomous rollout, a human operator monitors the execution and can instantaneously intervene via the SpaceMouse to correct deviations or guide the agent out of unpromising states, thereby facilitating efficient exploration.

Instead of a single replay buffer, HIL-SERL maintain two distinct replay buffers to manage the data effectively: an *online RL buffer* that stores transitions from the agent's autonomous policies, and a *demonstration buffer* that houses pre-collected expert demonstrations and human intervention data. For policy updates, we construct training batches by sampling from both buffers simultaneously with a **1:1** ratio. This balanced sampling strategy allows the agent to learn robustly from a diverse distribution of expert guidance and self-generated experiences.

Hyperparameters. We provide the detailed hyperparameters for the environment setup and algorithm training in Table 6.

Table 6: Hyperparameters for Real-Robot Experiments.

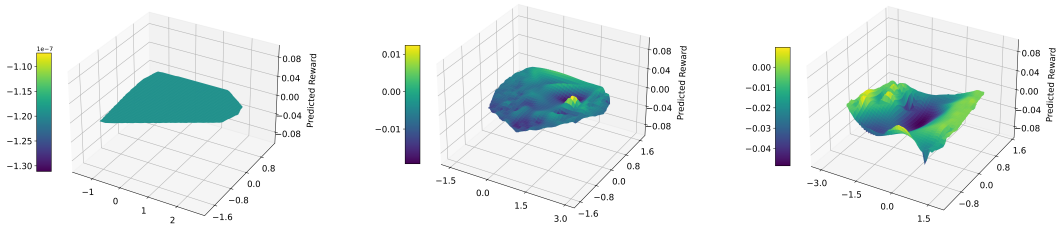
Hyperparameter	Value
Environment & General	
Image resolution	128×128
Proprioceptive state dim	4
Control frequency	10 Hz
Action dimension	4
Batch size	256
Total steps	15,000
SAC	
Actor/Critic Learning rate	3×10^{-4}
Discount factor γ	0.97
UTD ratio	2
Hidden dimension	256
Target smoothing τ	0.005
Online buffer capacity	30,000
MoDem (SLOPE)	
Planning horizon H	5
CEM iterations	6
Q ensemble size	5
Latent dimension	50
MLP dimension	512
Discount factor γ	0.99
Consistency coefficient	20.0
Value coefficient	0.1
Pi coefficient	0.5
Temporal decay coefficient	0.5
Target model momentum	0.995
Smoothing factor α	0.4

[3] https://huggingface.co/docs/lerobot/hilserl

C Additional Experimental Results

C.1 Visualization and Analysis of Reward Landscapes

We visualize the reward landscapes of TD-MPC2, MoDem, and SLOPE separately during rollout. As shown in Fig. 10, TD-MPC2 yields an almost flat reward landscape, indicating that its reward model fails to provide meaningful guidance for policy evaluation. Compared to MoDem, SLOPE generates a more discriminative reward landscape, while MoDem remains relatively flat. This suggests that SLOPE offers clearer optimization signals, enabling more effective credit assignment and policy improvement under sparse rewards.



(a) TD-MPC2 (b) MoDem (c) **SLOPE(Ours)**

Figure 10: Visualization of reward landscape. We collect trajectory data during rollout as the basis for visualization. All methods are evaluated on the same set of states, using their respective reward models to predict rewards. The high-dimensional state representations are projected into a 2D space using Principal Component Analysis (PCA), and predicted rewards are plotted over a mesh grid to construct the reward surfaces. To avoid distortion from extreme values caused by successful states (which yield disproportionately high predicted rewards), we exclude them and visualize only states with ground-truth rewards of zero.

C.2 Visualization of Shaped Reward Evolution in GridWorld

To further investigate the mechanism of our Bootstrapped Potential Construction, we visualize the temporal evolution of the shaped reward landscape during the training process on the 10×10 GridWorld task introduced in Section 4.1. Fig.11 displays the heatmaps of the shaped reward $\widetilde{r}(\mathbf{s}, \mathbf{a})$ at varying training iterations (iteration 10, 20, 30, 40, and 100).

As shown in Fig.11, in the early training stage (iteration 10), high rewards are limited to the area around the goal state (bottom-right corner). The rest of the state space provides little information, similar to the original sparse reward setting. However, as training continues (from iteration 20 to 40), the reward signal propagates backward from the goal towards the start state. By iteration 100, the reward landscape extends across the entire state space. Meanwhile, the color transition becomes smoother, establishing a clear directional trend that effectively guides the agent towards the goal.

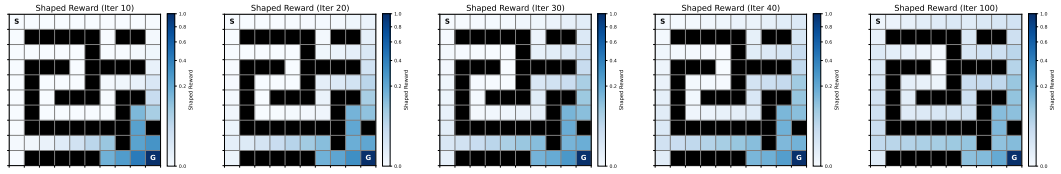


Figure 11: Visualization of the shaped reward landscape evolution on the 10×10 GridWorld. Darker blue regions indicate higher reward values. As training progresses, the dense reward signal propagates from the goal state (bottom-right) backwards to the initial state (top-left), progressively forming a global gradient field.

C.3 Detailed Performance on Each Task

Fig.12 visualizes the aggregate performance of SLOPE against baseline algorithms on all 20 evaluated tasks. This extensive evaluation covers distinct domains (*ManiSkill3*, *Meta-World*, *RoboSuite*, *Adroit*)

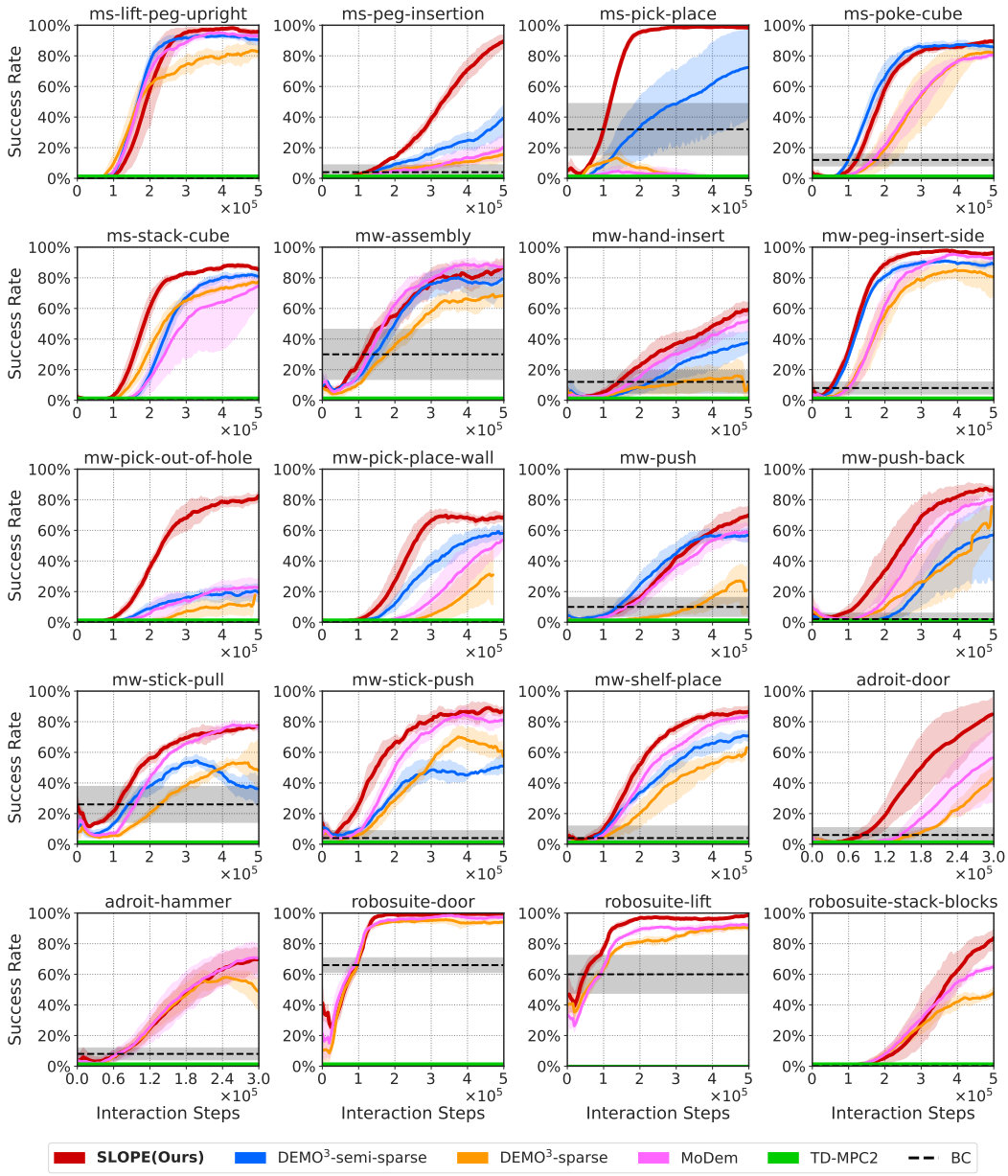


Figure 12: Success rate comparison on 20 tasks. We conduct all experiments using 5 distinct random seeds $\{1, 2, 3, 4, 5\}$.

and varying reward densities (sparse and semi-sparse). As evidenced by the curves, SLOPE not only converges faster but also exhibits lower variance across different seeds, highlighting its robustness and generalizability in solving diverse manipulation problems.

C.4 Evaluation on Dense Reward Tasks

While the primary contribution of SLOPE lies in resolving sparse reward challenges, we further investigate its efficacy in standard dense reward settings to assess its generality. We evaluate SLOPE on the DMC, selecting 8 representative locomotion tasks with state-based inputs. We compare our method against three baselines: **TD-MPC2** (our backbone algorithm), **Dreamerv3**, and **SAC**. The results of SAC, TD-MPC2 and Dreamerv3 are downloaded from this link [4].

Implementation Details for Dense Rewards. It is worth noting that for these dense reward experiments, we modified the SLOPE configuration. Since the environment provides continuous and informative feedback, the exploration challenge is significantly mitigated compared to sparse settings. Therefore, we disabled the Optimism-Driven Landscape Shaping component and relied solely on the Bootstrapped Potential Construction. In this configuration, SLOPE essentially acts as TD-MPC2 augmented with a dynamic PBRS mechanism.

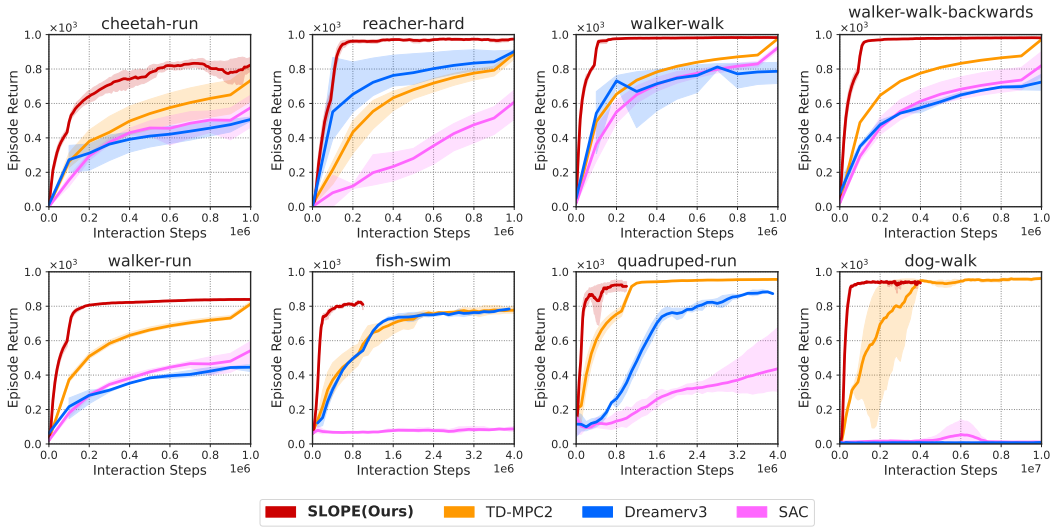


Figure 13: Performance comparison on 8 tasks from the DMC with dense rewards. We compare SLOPE (Ours) against TD-MPC2, Dreamerv3, and SAC. All results are averaged over 5 random seeds, with shaded regions indicating 95% CIs. SLOPE consistently achieves faster convergence and higher asymptotic performance compared to the baselines, demonstrating that bootstrapped potential shaping accelerates learning even in dense reward settings.

Results and Analysis. The results on the DMC tasks are illustrated in Fig.13. First, compared to the backbone TD-MPC2 (orange curves), SLOPE (red curves) demonstrates significantly faster convergence speeds across nearly all tasks. This acceleration suggests that shaping the reward with the learned value function creates a more favorable optimization landscape, allowing the planner to identify high-reward trajectories more efficiently. Furthermore, in terms of asymptotic performance, SLOPE consistently matches or outperforms both the model-free baseline (SAC) and the model-based baseline (DreamerV3). These results collectively confirm that the benefits of Bootstrapped Potential Construction extend beyond sparse reward domains, acting as an effective regularizer that efficiently guides the planner within dense reward landscapes.

C.5 Generalization to Other MBRL Algorithm: Dreamerv3

To assess the universality of our framework beyond the specific architecture of TD-MPC2, we integrated SLOPE into Dreamerv3 [5], a leading MBRL algorithm known for its robust performance across diverse domains.

[4] https://github.com/nicklashansen/tdmpc2/tree/main/results

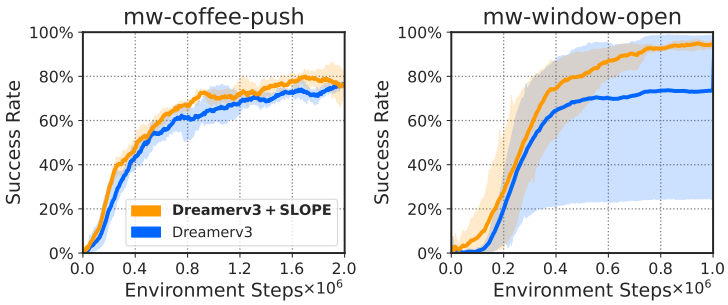


Figure 14: Performance comparison between Dreamerv3 + SLOPE and vanilla Dreamerv3 on *Meta-World* sparse reward tasks. By injecting the potential-based shaping signal into the critic's target computation during latent imagination, SLOPE significantly accelerates learning and improves final performance compared to the strong baseline.

Implementation Details. The integration of SLOPE into Dreamerv3 requires a distinct strategy due to the architectural differences in how reward models are utilized.

- **Difference in Backbone:** TD-MPC2 employs the learned reward function primarily for inference-time planning (i.e., evaluating trajectory return via MPPI). In contrast, Dreamerv3 decouples representation learning from Actor/Critic learning. The world model learns to predict rewards to shape the latent representation, while the Actor and Critic learn behaviors via latent imagination.
- **Integration Strategy:** To preserve the fidelity of the world model's representation learning, we do not alter the training target of the reward predictor in the world model. The reward predictor continues to learn the ground-truth environmental reward $r(\mathbf{s}, \mathbf{a})$. Instead, we intervene during the Critic learning phase. Specifically, during the latent imagination rollouts used to calculate the value targets, we augment the predicted rewards $\hat{r}_t$ with our potential-based shaping term:

$$\tilde{r}_t = \hat{r}_t + \gamma\Phi(\hat{\mathbf{s}}_{t+1}) - \Phi(\hat{\mathbf{s}}_t), \tag{10}$$

  where $\Phi(\cdot)$ is the bootstrapped potential. This ensures that the Critic learns a value function that incorporates global directional guidance, subsequently guiding the Actor, while the World Model remains an accurate simulator of the environment.

Results and Analysis. We evaluated this integration on two sparse-reward tasks from the *Meta-World* benchmark: `Coffee-Push` and `Window-Open`. As shown in Figure 14, Dreamerv3 combined with SLOPE (Orange curves) significantly outperforms the vanilla Dreamerv3 (Blue curves). These results confirm that the benefits of potential landscape construction are algorithm-agnostic and can also effectively enhance policy optimization in decoupled MBRL architectures.

C.6 Integration with LS-Imagine

To evaluate SLOPE beyond continuous-control robotics, we integrate it with LS-Imagine [39] and evaluate on its native MineDojo task, *Harvest water with bucket*. We compare the original LS-Imagine with LS-Imagine augmented by SLOPE over one million environment steps. This experiment examines whether SLOPE provides complementary benefits within an existing framework for long-horizon exploration.

As shown in Fig. 15, incorporating SLOPE accelerates learning, particularly during the first half of training. The augmented agent reaches higher success rates substantially earlier, whereas the original LS-Imagine requires more interactions to achieve comparable performance. The performance gap narrows later in training, but adding SLOPE still yields a higher average

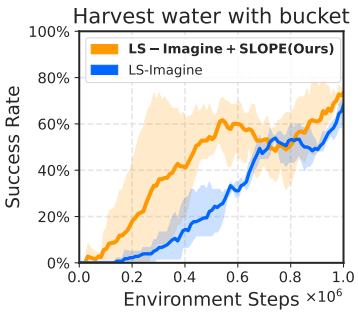


Figure 15: Success-rate of LS-Imagine with and without SLOPE on *Harvest water with bucket*.

final success rate. The learning curves therefore indicate a substantial benefit in learning efficiency, accompanied by a modest improvement in final performance.

These results suggest that SLOPE's value-based shaping complements LS-Imagine's long-horizon imagination mechanism. By providing additional task-directed guidance through the learned reward model, SLOPE improves learning on this open-world task, extending its demonstrated applicability beyond robotic manipulation.

C.7 Long-Horizon Hard Exploration Tasks

We evaluate SLOPE in two complementary long-horizon settings: high-dimensional continuous control in HumanoidBench and discrete-action exploration in Montezuma's Revenge. All agents learn from scratch without demonstrations or pre-collected successful trajectories.

HumanoidBench. We evaluate on `h1hand-hurdle` and `h1hand-maze` from HumanoidBench [40], using the Unitree H1 humanoid with dexterous hands. Both tasks have a 1,000-step episode horizon, and training runs for two million interaction steps. The benchmark authors identify hurdle as a hard-exploration problem, where agents learn forward locomotion but fail to discover the jumping behavior needed to overcome obstacles. As shown in Fig. 16, SLOPE achieves substantially higher returns than TD-MPC2 on hurdle. Its return continues to improve throughout training, whereas the baseline makes limited progress. On maze, SLOPE also improves returns earlier and achieves higher final performance, although the final gap is smaller. The larger variability on hurdle indicates that learning remains challenging despite the substantial improvement in average return.

Montezuma's Revenge. To examine a canonical hard-exploration domain, we integrate SLOPE with DreamerV3 to accommodate the discrete action space and train for ten million environment steps. Figure 16 shows that DreamerV3+SLOPE achieves higher average scores during most of training and finishes above vanilla DreamerV3.

Together, these experiments show that SLOPE's benefits extend beyond demonstration-supported manipulation. It improves learning from scratch across continuous and discrete action spaces with different MBRL backbones. The gains vary by task, with a pronounced improvement on hurdle and more moderate improvements on maze and Montezuma's Revenge. These results support its applicability to the evaluated long-horizon exploration settings.

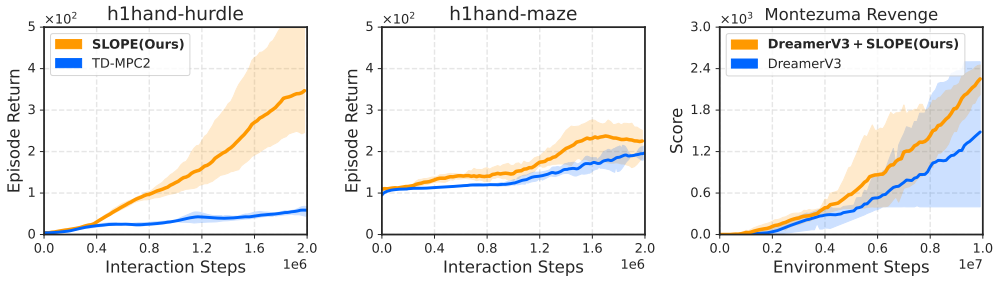


Figure 16: Learning curves on long-horizon exploration tasks. We evaluate SLOPE with DreamerV3 on Montezuma's Revenge over ten million environment steps and with TD-MPC2 on HumanoidBench over two million interaction steps. Both HumanoidBench tasks have a 1,000-step episode horizon.

C.8 Standalone Performance in Zero-Demonstration Settings

To provide a more comprehensive evaluation of SLOPE's foundational capabilities, we assess its performance in the absence of expert demonstrations within fully sparse environments. As shown in Fig. 17, SLOPE effectively learns from scratch and achieves task success without any prior guidance. In contrast, the baseline TD-MPC2 completely fails to overcome the severe exploration bottleneck, consistently yielding near-zero returns.

Despite this standalone effectiveness, we emphasize the highly complementary relationship between our shaping framework and expert demonstrations. While demonstrations provide critical high-value initial states, SLOPE structurally propagates these localized signals into a dense, informative landscape. This synergy maximizes the utility of limited data, significantly enhancing sample efficiency for real-world applications where unguided exploration is prohibitively expensive.

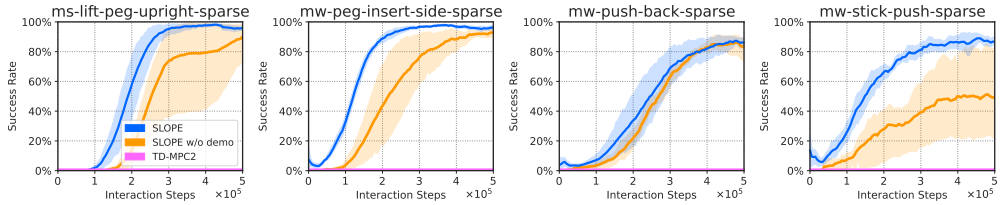


Figure 17: Performance of SLOPE without demonstrations in fully sparse reward settings.

C.9 Performance on Semi-Sparse Tasks

We also evaluate the performance of SLOPE on semi-sparse tasks, where sparse-reward environments are augmented with handcrafted stage rewards through manual task decomposition. As shown in Fig. 18 (Left), SLOPE achieves significantly better performance in the semi-sparse setting compared to the fully sparse rewards. This improvement demonstrates that while SLOPE is not tailored for semi-sparse tasks, it still benefits from the structured rewards and achieves competitive results.

C.10 Comparison with Other Reward Shaping Methods

We compare SLOPE with two alternative reward shaping methods: similarity-based and entropy-based potentials, both designed to encourage exploration. The similarity-based potential $\Phi_{\text{sim}}(\mathbf{s}) = \|\mathbf{z}' - \texttt{stop_grad}(h_\theta(\mathbf{s}'))\|_2^2$ drives the agent toward novel states, while the entropy-based potential $\Phi_{\text{entroy}}(\mathbf{s}) = \mathcal{H}(\pi(\cdot|\mathbf{s}))$ promotes diverse behavior through higher policy entropy.

As shown in Figure 18 (Right), both methods achieve reasonable performance on relatively simple tasks, such as `ms-poke-cube-sparse`, where exploratory behaviors are easier to reward. However, as the task complexity increases, the performance of both similarity-based and entropy-based shaping methods degrades significantly. In contrast, SLOPE consistently maintains strong performance across these harder tasks. This suggests that in complex sparse-reward environments, general exploration incentives are insufficient, and agents benefit more from reward signals that are explicitly aligned with task progression.

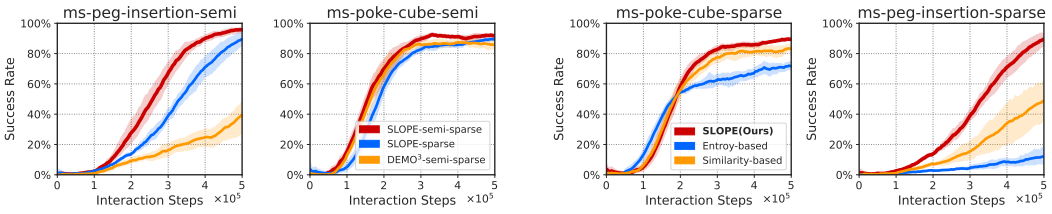


Figure 18: **Left**: Performance evaluation of SLOPE in semi-sparse reward settings. **Right**: Comparison of different reward shaping methods.

C.11 Comparison with Reward Smoothing: DreamerSmooth

To provide a more comprehensive evaluation of SLOPE, we compare it against DreamSmooth [4], a representative MBRL algorithm designed to mitigate reward prediction bottlenecks. Its methodology utilizes temporal relaxation to predict a temporally-smoothed reward instead of the exact scalar at a specific timestep, thereby addressing prediction ambiguity.

Core Implementation. Unlike SLOPE, which is designed for purely sparse-reward environments, DreamSmooth is typically applied to tasks with existing dense reward structures augmented by high-magnitude sparse signals at key stage points. We reimplemented this approach by applying a Gaussian smoothing function f to rewards r during replay buffer storage:

$$\tilde{r}_t \leftarrow f(r_{t-L:t+L}) = \sum_{i=-L}^{L} f_i \cdot r_{t+i} \quad \text{s.t.} \quad \sum f_i = 1 \tag{11}$$

Our implementation adheres to the following core principles:

- **Temporal Relaxation**: By diffusing the sparse reward signal over a temporal horizon L, the model can predict rewards several timesteps earlier or later without incurring large losses.
- **Sum Preservation**: The smoothing function is normalized to ensure $\sum \tilde{r} = \sum r$, preserving policy optimality under the original reward objective.
- **Gaussian Kernel**: We utilize a Gaussian kernel ($\sigma = 8$) to provide the planner with a smooth, directional gradient toward the goal, transforming the sparse rewards into dense.

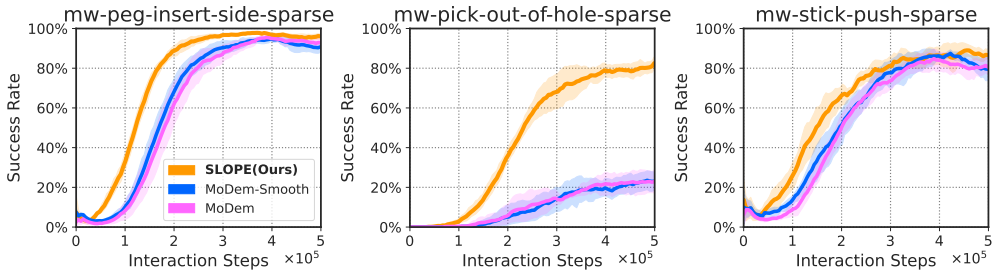


Figure 19: Comparative analysis between SLOPE and Reward Smoothing.

Results and Analysis. As illustrated in Fig. 19, SLOPE consistently outperforms the reward smoothing baseline (MoDem-Smooth) across multiple *Meta-World* tasks, achieving both faster convergence and higher asymptotic success rates. Notably, the integration of temporal smoothing into MoDem yields no significant performance gain over the original sparse baseline. While temporal relaxation in DreamSmooth eases the reward prediction bottleneck by diffusing signals over a temporal window L, it remains fundamentally limited by the scarcity of the underlying scalar signal. In fully sparse-reward environments, the absence of dense supervision or stage-wise priors means that temporal diffusion merely attenuates the infrequent success signals rather than constructing an informative gradient. Consequently, reward smoothing fails to bridge the exploration gap in challenging tasks like `pick-out-of-hole`. In contrast, SLOPE's potential-based reward provides precise directional guidance by leveraging structured potential landscapes, offering a more robust planning signal than simple Gaussian diffusion ($\sigma = 8$) for complex robotic manipulation.

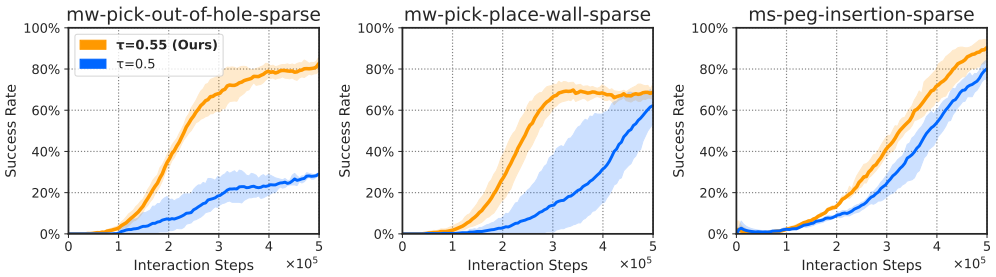


Figure 20: Effect of the quantile coefficient on three sparse-reward tasks. Mild optimistic weighting ($\tau = 0.55$, orange) improves learning speed and average final success rates compared with the symmetric objective ($\tau = 0.5$, blue). The largest final-performance gain occurs on Pick Out of Hole.

C.12 Effect of the Quantile Coefficient

To examine the effect of mild optimism across environments, we compare $\tau = 0.55$ with the symmetric objective ($\tau = 0.5$) on three sparse-reward tasks: Pick Out of Hole, Pick Place Wall, and Peg Insertion.

As shown in Fig. 20, $\tau = 0.55$ improves learning on all three evaluated tasks, although the magnitude of the benefit varies. On Pick Out of Hole, it produces a much steeper improvement during early training and raises average final success. On Pick Place Wall, the optimistic objective reaches high success rates substantially earlier, while the symmetric objective partially catches up near the end of training. The final improvement is therefore smaller on this task. On Peg Insertion, $\tau = 0.55$ also accelerates learning and maintains higher success rates during the later training stages.

C.13 Ablating Potential Shaping and Optimistic Learning

To separate the contributions of potential shaping and optimistic learning, we compare four variants: MoDem, MoDem with potential shaping only, MoDem with optimistic learning only, and SLOPE, which combines both components. We use Pick Place and Peg Insertion, two sparse-reward ManiSkill3 tasks on which MoDem achieves relatively low performance. All variants share the demonstrations, TD-MPC2 backbone, interaction budget, and evaluation protocol.

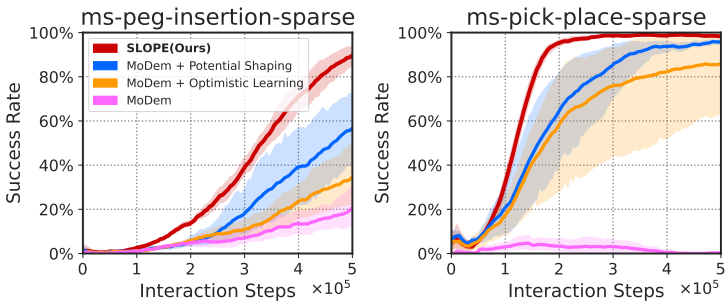


Figure 21: Separate ablations of potential shaping and optimistic learning on Peg Insertion and Pick Place. Potential shaping alone substantially improves over MoDem, while adding optimistic learning further improves final performance on Peg Insertion and accelerates learning on Pick Place.

Figure 21 shows that potential shaping alone substantially improves over MoDem on both tasks and achieves higher final performance than optimistic learning alone. This indicates that shaping accounts for a substantial portion of SLOPE's performance gains.

Optimistic learning provides complementary, task-dependent benefits. On Peg Insertion, removing optimism slows learning and leaves a substantial gap in final performance, showing that shaping alone does not fully explain SLOPE's effectiveness. On Pick Place, shaping alone eventually approaches SLOPE's final success rate, but requires considerably more interactions to reach high performance. Optimism therefore primarily accelerates learning on this task.

These results indicate that potential shaping accounts for much of the performance improvement on the two evaluated tasks. Optimistic learning complements this guidance by placing greater weight on high-return targets, with benefits reflected in higher final performance on Peg Insertion and faster learning on Pick Place. Combining both components achieves the best overall performance.

C.14 Computational Resources

To evaluate the efficiency of our approach, we conducted a comparative experiment using the *ManiSkill3* benchmark to assess the computational resource requirements of SLOPE against various baseline algorithms. To ensure a fair comparison of resource usage, these benchmarking experiments were all conducted on a high-performance computing server equipped with dual Intel Xeon Gold 6240 CPUs (totaling 36 cores and 72 threads, 2.60 GHz) and 10 NVIDIA GeForce RTX 2080 Ti GPUs.

Table 7: Comparison of training resources for different algorithms. The best score is marked with **bold**. The direction of the arrow indicates whether the desired metric is large or small.

Method	Training time ↓	CPU memory ↓	Learnable parameters ↓	Average success rate ↑
TD-MPC2	**7.9h**	**29.53GB**	**5.6M**	0%
MoDem	9.7h	30.48GB	**5.6M**	55.0%
DEMO3-semi-sparse	11.8h	30.48GB	5.7M	74.0%
DEMO3-sparse	11.8h	30.48GB	5.7M	54.8%
SLOPE(Ours)	9.9h	40.37GB	**5.6M**	**92.0%**

As illustrated in Table 7, although SLOPE requires a higher allocation of CPU memory (40.37 GB) compared to the baseline, this additional resource consumption translates into a substantial gain in performance. Specifically, SLOPE achieves a remarkable average success rate of **92.0%**, surpassing the strongest baseline (DEMO^3-semi-sparse) by a margin of 18.0%. The increased memory footprint stems from augmenting the demonstration buffer with successful trajectories encountered during training, a strategy designed to mitigate data imbalance in sparse-reward tasks. Furthermore, despite the increased memory footprint, our method maintains high computational efficiency with a training time of 9.9 hours, which is comparable to MoDem and notably faster than DEMO^3 (11.8 hours).

C.15 Visualization of Real-World Policy Rollouts

In this section, we provide the visualization of the learned policies deployed on the real robot. Figure 22 illustrates the frame-by-frame execution sequences of our method, SLOPE, across the three sparse-reward manipulation tasks. As shown in the snapshots, the agent effectively controls the robotic arm to approach the target objects and perform precise interactions, demonstrating the robustness of the learned behaviors in real-world scenarios.

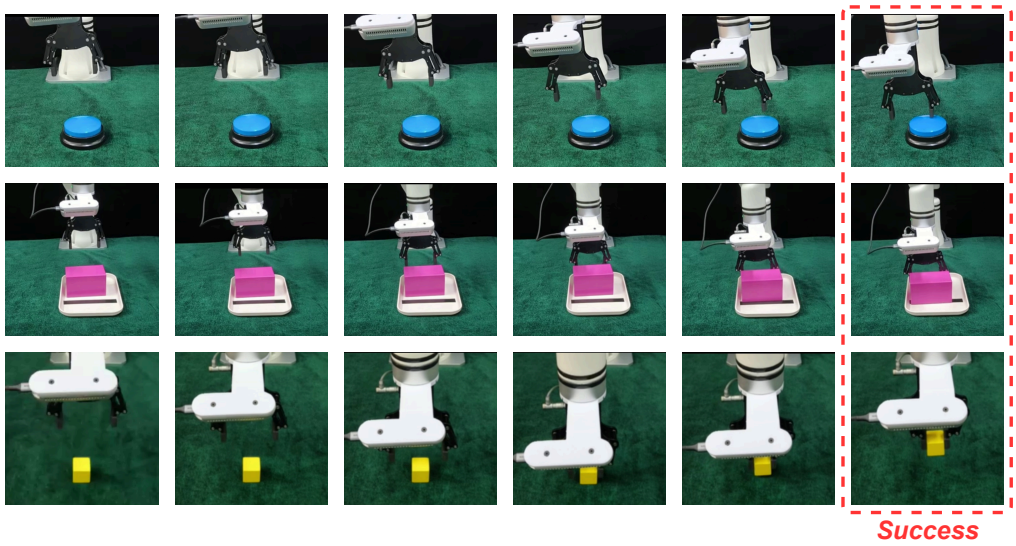


Figure 22: Visualization of successful policy rollouts in the real world. The figure displays chronological snapshots (left to right) of the agent executing the **Press Button** (top row), **Push Cube** (middle row), and **Grasp Cube** (bottom row) tasks. The final column, marked with a red dashed box, highlights the successful completion state for each sparse-reward task.

D Algorithm

The pseudocode of our proposed method is presented in Algorithm 1.

Our implementation of SLOPE is designed to be backbone-agnostic while carefully adapting to the specific architectural constraints of the underlying algorithm. When integrating with TD-MPC2, which utilizes an ensemble of Q-networks, we adopt an asymmetric aggregation strategy to calculate the shaping term $\gamma\Phi(s') - \Phi(s)$. Specifically, we compute the current potential $\Phi(s)$ using the ensemble average, aligning with TD-MPC2's standard practice for low-variance value estimation. However, for the next state potential $\Phi(s')$, we employ the ensemble minimum. As PBRS is particularly sensitive to the next-state term, where any overestimation in $\Phi(s')$ directly translates into an artificially inflated reward signal, utilizing the minimum operator effectively mitigates overestimation bias. This ensures that the shaping signal reflects reliable potential gains rather than epistemic noise, while maintaining a stable baseline through the averaged current potential.

This strategy generalizes to architectures that employ standard explicit target networks, such as Dreamerv3 or our real-world robotic control frameworks. In these settings, SLOPE computes $\Phi(s)$ using the online network and $\Phi(s')$ using the slow-moving target network, strictly adhering to the stability mechanisms of the respective backbones. This flexibility ensures that SLOPE provides consistent, risk-averse guidance across diverse MBRL implementations, whether they rely on ensemble statistics or frozen target parameters.

Algorithm 1 SLOPE: Shaping Potential Landscapes for MBRL

Input: $\mathcal{D}^{\text{demo}}$: demonstration dataset, $\mathcal{B}$: replay buffer (initially empty)
Parameter: θ: model parameters, π_θ: policy, Π_θ: planner, τ: quantile estimation coefficient, η: reward shaping weight
Output: Π_θ: planner, π_θ: trained policy

// Phase 1: Policy Pretraining
for each policy update **do**
  Sample $(s_t, a_t) \sim \mathcal{D}^{\text{demo}}$
  Update h_θ, π_θ using BC loss $\mathcal{L}_{\text{BC}}(\theta)$
end for
// Phase 2: Model Seeding
for each seeding rollout **do**
  Collect rollout $\mathcal{T} \leftarrow \{s_t, a_t, r_t, s_{t+1}\}_{0:T}$ using $a_t \sim \pi_\theta(s_t)$
  Add trajectory to buffer: $\mathcal{D}^{\text{seed}} \leftarrow \mathcal{D}^{\text{seed}} \cup \mathcal{T}$
end for
for each model update **do**
  Sample $(s_t, a_t, r_t, s_{t+1})_{0:T}$ from $\mathcal{D}^{\text{demo}} \cup \mathcal{D}^{\text{seed}}$
  Update $h_\theta, d_\theta, R_\theta, Q_\theta, \pi_\theta$ using TD-MPC loss $\mathcal{L}_{\text{TD-MPC}}(\theta)$
end for
// Phase 3: Interactive Learning with SLOPE Shaping
Initialize buffer: $\mathcal{B} := \mathcal{D}^{\text{seed}}$
while interaction budget not exhausted **do**
  Collect rollout $\mathcal{T} \leftarrow \{s_t, a_t, r_t, s_{t+1}\}_{0:T}$ using $a_t \sim \Pi_\theta(s_t)$
  Add trajectory to buffer: $\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{T}$
  Sample $(s_t, a_t, r_t, s_{t+1})_{t:t+H}$ from $\mathcal{D}^{\text{demo}} \cup \mathcal{B}$
  Compute shaped reward: $\tilde{r}_t = r_t + \Delta r_t$, where $\Delta r_t = \gamma \mathbb{E}_{s'}[\Phi(s')] - \Phi(s_t)$
  Update $h_\theta, d_\theta, R_\theta, Q_\theta, \pi_\theta$ using TD-MPC loss $\mathcal{L}_{\text{TD-MPC}}(\theta)$
end while
return Π_θ

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See Abstract and Introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Conclusion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: See detailed proofs in Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The implementation details are provided in Appendix and the code is linked anonymously in the Abstract.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code is linked anonymously in the Abstract.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: See implementation details in Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The curves and shaded areas in our paper represent the mean and 95% confidence intervals (CIs) over 5 independent runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Section Computational Resources in Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and ensure that our paper complies with them.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper focuses on fundamental algorithmic improvements to MBRL in simulated environments. As foundational research without a direct path to specific, deployable applications, it does not present immediate positive or negative societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cite the original paper that produced the code package or dataset correctly.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The demonstrations used in our paper in released in the anonymized URL.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The LLM is used only for editing and formatting.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.