

Learning to Plan, Planning to Learn: Adaptive Hierarchical RL-MPC for Sample-Efficient Decision Making

Toshiaki Hori
Jonathan DeCastro
Deepak Gopinath
Avinash Balachandran^{*}
Guy Rosman

Toyota Research Institute, Cambridge, MA 02139, USA

TOSHIAKI.HORI@TRI.GLOBAL
JONATHAN.DECASTRO@TRI.GLOBAL
DEEPAK.GOPINATH@TRI.GLOBAL
AVINASH.BALACHANDRAN@TRI.GLOBAL
GUY.ROSMAN@TRI.GLOBAL

Editors: G. Sukhatme, L. Lindemann, S. Tu, A. Wierman, N. Atanasov

Abstract

We propose a new approach for solving planning problems with a hierarchical structure, fusing reinforcement learning and MPC planning. Our formulation tightly and elegantly couples the two planning paradigms. It leverages reinforcement learning actions to inform the MPPI sampler, and adaptively aggregates MPPI samples to inform the value estimation. The resulting adaptive process leverages further MPPI exploration where value estimates are uncertain, and improves training robustness and the overall resulting policies.

This results in a robust planning approach that can handle complex planning problems and easily adapts to different applications, as demonstrated over several domains, including race driving, modified Acrobot, and Lunar Lander with added obstacles. Our results in these domains show better data efficiency and overall performance in terms of both rewards and task success, with up to a 72% increase in success rate compared to existing approaches, as well as accelerated convergence ($\times 2.1$) compared to non-adaptive sampling.

1. Introduction

Reinforcement learning (RL) has enjoyed widespread success in constructing control policies for embodied applications, including robotics [Makoviychuk et al. \(2021\)](#), fully- / partially-autonomous vehicles [Kiran et al. \(2021\)](#), and board games [Silver et al. \(2017\)](#). Nonetheless, learning RL-based controllers can be challenging when faced with environments and physical embodiments where it is costly or unsafe to interact with the target environment. In domains where exploration data can be acquired in abundance, e.g., those with highly parallelizable simulators such as robotics, policies can be trained to solve very complex tasks. However, in domains where data is more limited, e.g., domains with expensive-to-collect human data, uncertain or unmodeled behaviors requiring more data to estimate, safety concerns limiting data collection, or those requiring extensive computation, more sample-efficient learning is needed.

In contrast to RL which seeks to solve for a globally-optimal policy, model-predictive control (MPC), and its sampling-based variants (e.g., MPPI), aim to solve a finite-horizon optimal control problem online by reasoning explicitly over local system behavior, usually using simplified models of the environment. They are well-suited to optimizing for short-term objectives, though they are often extended with a terminal cost that can capture effects beyond the planning horizon. Numerous recent works have blended model-free RL and MPC to achieve data efficiency and safety

^{*} Los Altos, CA, 94022, USA

using a variety of techniques [Romero et al. \(2023a\)](#); [Reiter et al. \(2024\)](#); [Bhardwaj et al. \(2020a, 2019\)](#); [Shin et al. \(2021\)](#); [Hansen et al. \(2024\)](#). However, such methods usually focus on specific cost formulations, couple RL and MPC in specific ways, or do not exploit RL and MPC problem structure.

An underexplored idea involves treating an MPPI controller as a *structured sampler*, which offers similar benefits to data augmentation – it can cheaply generate plausible trajectories from an approximate model that can augment the RL training data. The sampler’s longer horizon affords more accurate value function learning, and the hierarchical decomposition offers clear locations to separate the role of the RL and MPC algorithms. Rather than optimizing each component separately through cost function coupling, we aim to explicitly reuse sampled MPPI trajectories as approximate, virtual rollouts to guide both value function learning and policy updates. Ultimately, drawing such samples reduces the number of real environment steps needed to learn a performant policy, without resorting to additional heuristics (e.g. reward shaping, curricula, feature selection). This is crucial for settings where actual environment queries are extremely expensive or unsafe to collect. Such a framework lends to a principled solution for balancing model-based supervision for achieving desired control performance and model-free exploration for achieving faster convergence and more efficient data utilization. The key challenge is that the distribution mismatch between the approximate model and the true environment, combined with off-policy biases from the MPPI sampler, can introduce systematic errors in value function estimates.

In this work, we propose an integrated RL–MPC framework that couples a high-level RL policy with a sample-based MPC controller via shared value function updates. Our contributions are threefold: (1) we introduce a bi-directional training pipeline that allows sample-based rollouts to improve value function learning as well as leverages improved policy estimates to tune the samples in a general setting, (2) we provide a means for adjusting the mixing of datasets and discuss how this maps to curriculum that initially explores safe, local rollouts, while eventually extending exploration to longer-horizon rollouts, and (3) we provide empirical evidence that coupling value-aware exploration with MPC-based rollouts significantly enhances sample efficiency and safety.

2. Related Work

Our work improves on the intersection of two main research threads in planning approaches. One thread is reinforcement learning ([Sutton and Barto, 2020](#); [Wang et al., 2024](#)), more specifically, model-based reinforcement learning [Moerland et al. \(2023\)](#). The other main thread is that of MPC techniques [Schwenzer et al. \(2021\)](#), and MPPI [Williams et al. \(2017\)](#); [Bhardwaj et al. \(2020b\)](#) based approaches within them.

Approaches for model-based reinforcement learning (MBRL) (e.g., ([Hafner et al., 2020](#); [Janner et al., 2019](#))) have demonstrated the benefit of learning latent world models that enable planning in imagination. However, such methods are costly to train, as a necessary condition for good policy performance is a well-trained model. In contrast, hybrid RL–MPC methods ([Nagabandi et al., 2018](#); [Okada and Taniguchi, 2020](#); [Li and Chen, 2025](#); [Mundheda et al., 2025](#)) allow for more flexibility, and learned dynamics into MPC loops, but typically without a feedback mechanism from MPC to the RL updates (in the actor or critic).

With hierarchical RL showing promise for sample efficiency [Wen et al. \(2020\)](#), recent work on blending MPC and value function learning have been proposed ([Hong et al., 2019](#); [Bhardwaj et al., 2020b,a](#); [Romero et al., 2023b](#); [Reiter et al., 2024](#); [Álvaro Serra-Gomez et al., 2025](#)). Each focus on

new strategies for blending, and the interface between the components for achieving better sample efficiency and to better balance exploration vs. modeling the environment.

In [Bhardwaj et al. \(2020a\)](#), the authors aim to balance the estimates from a MPC-derived local Q-function with a RL-estimated value function. Our approach differs by closing this loop; sampled MPC rollouts act as structured exploration priors that inform value function and policy updates to accelerate high-level learning. Some approaches leveraged RL and MPC by leveraging RL to improve the MPC estimates [Wang et al. \(2025\)](#), or leverage MPPI samples to help robustify the RL training [Mundheda et al. \(2025\)](#), but not within a single coherent co-training framework. Several works have also looked at specific complex domains where RL and MPC can be combined, along with domain-specific insights [Nguyen et al. \(2025\)](#); [Kotecha et al. \(2025\)](#).

A related thread of research focuses on safe RL ([Berkenkamp et al., 2017](#); [Garcia and Fernández, 2015](#)), where constraints are enforced via Lyapunov conditions or shielding mechanisms. Our approach offers an alternative safety-critical reasoning to the MPC layer while using RL to shape high-level intent. In human-interactive domains, several works (e.g. [Sadigh et al. \(2018\)](#); [Rudenko et al. \(2020\)](#)) address prediction and intent inference; the proposed RL–MPC coupling provides a mechanism to integrate such predictive models into closed-loop decision-making.

3. Background and Problem Statement

We assume access to an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, r, P, \gamma, \mu \rangle$, where $s_t \in \mathcal{S}$ is the state/observation at discrete time t , $a_t \in \mathcal{A}$ be the action taken at t , $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is a reward function, $P : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$ is a transition probability, $\gamma \in [0, 1]$ is a discount factor, and $\mu \in \Delta(\mathcal{S})$ is the initial state distribution. The general RL objective is to find an optimal policy π_ϕ^* (under parameters ϕ) that maximize the infinite-time discounted return

$$\pi_\phi^* = \arg \max_{\pi} \mathbb{E}_{\mathcal{M}, \pi, s_0 \sim \mu} \left[\sum_{\tau=t}^{\infty} \gamma^\tau r(s_\tau, a_\tau) \right]. \quad (1)$$

Model-predictive path integral (MPPI, [Williams et al. \(2017\)](#)) is a sample-based algorithm that solves an optimal control problem expressed by a cost function and a model $\hat{\mathcal{M}} = \langle \mathcal{X}, \mathcal{U}, \hat{r}, \hat{P}, H, \hat{\mu} \rangle$ that *approximates* \mathcal{M} , where $x_t \in \mathcal{X}$ is the system state, $u_t \in \mathcal{U}$ is the control input, and \hat{P} is the dynamical system $\hat{P} : \mathcal{X} \times \mathcal{U} \mapsto \Delta(\mathcal{X})$ (e.g. equations of motion, a neural approximator, etc.) $\hat{r} : \mathcal{X} \times \mathcal{U} \mapsto \mathbb{R}$ is an approximate reward function, $H > 0$ is a integer-valued finite planning horizon, and $\hat{\mu} \in \Delta(\mathcal{X})$ is an approximation to μ . We assume the existence of a state interface map $\psi : \mathcal{X} \mapsto \mathcal{S}$ that converts MPPI states to RL observations (e.g., projecting a full physical state to a lower-dimensional observation vector).

MPPI draws N input sequences around a nominal $u_{t:t+H-1}$, then samples additive control noise according to $\varepsilon_\tau^i \sim \mathcal{N}(0, \Sigma)$. The objective evaluates, for each sample $k \in 1, \dots, K$, the cost

$$J_k^{\hat{P}}(u_{t:t+H-1}, a_t) = \mathbb{E}_{\hat{P}} \left[\sum_{\tau=t}^{t+H-1} J(\hat{x}_\tau^k, u_\tau^k + \varepsilon_\tau^k; a_t) + \phi(\hat{x}_{t+H}^k; a_t) \right] \quad (2)$$

to be minimized. The k -th state is updated according to $\hat{x}_{\tau+1}^k \in \hat{P}(\hat{x}_\tau^k, u_\tau^k + \varepsilon_\tau^k)$, with $\hat{x}_t^k = x_t$. The MPPI update rule for the k -th candidate is defined by:

$$u_t \leftarrow u_t + \sum_{j=1}^K \tilde{w}_j \varepsilon_t^j, \quad w_k \propto \exp(-J_k^{\hat{P}} / \lambda_{\text{temp}}), \quad \tilde{w}_k = \frac{w_k}{\sum_{j=1}^K w_j}, \quad (3)$$

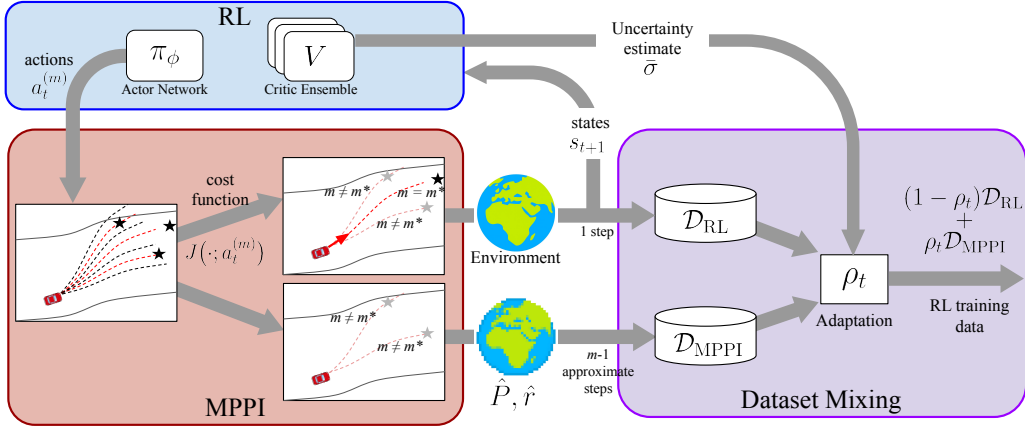


Figure 1: Diagram of the combined approach. Samples from the RL policy generates actions that are fed to MPPI, which then generates a set of candidates m_0, m_1, \dots . One m^* is selected and applied to the real environment. The remainder are stored in a buffer \mathcal{D}_{MPPI} . Data from the two buffers is sampled from a convex combination on the parameter ρ_t , defined by the uncertainty as estimated by a critic ensemble. The data is passed to RL for value and policy iteration.

where $\lambda_{\text{temp}} > 0$ is a temperature parameter.

In (2), we allow the running cost $J(\cdot, \cdot; a_t)$ and terminal cost $\phi(\cdot; a_t)$ (accumulating costs at all times $t \geq H$) to couple hierarchically via the RL action a_t . These costs take general forms in a_τ , e.g. parametric forms Romero et al. (2023b), terminal value Hansen et al. (2024), or target tracking Wang et al. (2023). Note that these are defined independently of the approximate reward \hat{r} , due to any domain-specific adjustments when applying state-action tuples to the experience buffer.

3.1. Problem Statement

Consider the case where we are given an MDP \mathcal{M} describing a physical system embedded in an environment, but also have access to a surrogate $\hat{\mathcal{M}}$, which is used for MPPI and assumed fast to sample from, but bounded according to Assumption 1.

Assumption 1 We assume the error between the approximate and true transition function is bounded according to $\alpha_P := \sup_{s,a,x,u} \left\| P(\cdot | s, a) - \hat{P}(\cdot | x, u) \right\|_{TV}$, where $\|\cdot\|_{TV}$ is the total variation norm. We further assume the error between the surrogate and true reward functions is bounded according to $\alpha_r := \sup_{s,a,x,u} |r(s, a) - \hat{r}(x, u)|$.

Our aim is to unify RL and MPC in a way that improves sample efficiency over existing RL and MPC combinations, while eliminating bias due to the distribution shift when sampling from $\hat{\mathcal{M}}$. We assume domains with reasonable complexity, that is, those exhibiting a combination of (hard) physical and (soft) task-related constraints with non-trivial transition functions (i.e. vehicle, aircraft, robot dynamics, etc.).

4. Technical Approach

We adopt a two-layer design, as shown in Fig. 1, where we assume an RL policy that outputs *high-level tactical parameters* used by an MPPI cost function (e.g., target states, cost weights that

penalize all state and action features, etc.), while an MPPI planning algorithm performs *low-level operational control* on the system. We posit that sampled rollouts from MPPI, which we call *virtual rollouts*, can offer benefits as an additional, structured data source for value function learning. This, in turn, offers overall benefits to learning a high-level RL policy. Additionally, we posit that the learned policy can be used to sample a diverse set of behaviors that benefit MPPI. Hence, the focus of this paper is in *value-based* RL methods.

We propose a commonly-used hierarchical controller setup where, at the high level, a value-based RL scheme outputs a set of actions while, at the low level, MPPI computes a set of locally optimal low-level input sequences for each high-level action. Of the candidate solutions, one optimal solution is applied to the environment; the remaining solutions are scored by a reward model and stored in an MPPI buffer. Crucially, for each high-level action we store only MPPI-feasible trajectories (not arbitrary samples) in an isolated buffer, so that the buffer reflects the controller’s optimality and keeps the data effectively on-policy with respect to the hierarchical controller.

At each RL training iteration, we mix RL buffer data with MPPI buffer data, and use this augmented data to update the value function and RL policy. Biases can be incurred in value learning due to any mismatches in the approximate model that MPPI uses for runtime optimization and any off-policy biases incurred due to the MPPI sampler. To allow our approach to have zero bias with respect to modeling approximations, we propose a tunable influence ratio to allow the algorithm to control for uncertainty, and hence recover zero asymptotic bias.

4.1. Coupling RL and MPPI

Several value-based RL approaches have seen widespread use, including proximal policy optimization (PPO), soft actor-critic (SAC), Advantage Actor Critic (A2C), etc. We choose PPO because the critic target considers future rewards, and hence our trajectory sampling scheme is amenable to that setup. On the other hand, common SAC algorithms only use a one-step lookahead as a critic target. The overall objective for both approaches is a composite of an actor loss, a critic loss and an entropy loss to encourage exploration. PPO is an *on-policy* algorithm, meaning that experience is accumulated using the policy at the current training iteration. In PPO’s experience buffer, we can extend and diversify future policy behavior by augmenting with MPPI rollouts without executing them on the actual environment.

Procedure Our algorithm is presented in Alg. 1. At each time t we draw M high-level candidate actions

$$a_t^{(m)} \sim \pi_\phi(\cdot | s_t), \quad m = 1, \dots, M. \quad (4)$$

We then sample a *single* set of K rollout perturbation sequences $\{\varepsilon_{t:t+H-1}^k\}_{k=1}^K$ and, from it, generate the state-action trajectories $(\hat{x}_{t:t+H}^k, u_{t:t+H-1}^k)$ using our approximate MPPI dynamics $\hat{x}_{\tau+1}^k \in \hat{P}(\hat{x}_\tau^k, u_\tau^k)$, with $\hat{x}_t^k = x_t$. For each m , we apply the cost $J_k^{\hat{P}}(u_{t:t+H-1}, a_t^{(m)})$ in (2), which is conditioned on the RL action $a_t^{(m)}$ that, when coupled with cost in this way, serves as a high-level knob that reshapes the MPPI objective. For each m , we execute the control update step in (3). Sampling actions in (4) in this way allows for a diverse set of feasible rollout options to enter the replay buffer.

Selected candidate m^* We select one candidate m^* among $m = 1, \dots, M$ to apply to the environment. Although it is possible to optimally select among available candidates, in order not to

Algorithm 1: RL + MPPI

Input: dynamics model \hat{P} , reward model \hat{r} , policy π_θ , value heads $\{V_d\}_{d=1}^D$, buffers $\mathcal{D}_{\text{RL}}, \mathcal{D}_{\text{MPPI}}$
Parameters: episodes N , steps T , horizon H , MPPI candidates per action M , minibatch B , update period N_{upd} , EMA λ , init ρ_0

```

for  $i = 1$  to  $N$  do
    for  $t = 1$  to  $T$  do
         $(a_t^{(1)}, \dots, a_t^{(m)}) \leftarrow \pi_\theta(\cdot | s_t)$ 
         $\{u_{t:t+H}^{(m)}\}_{m=1}^M, \{s_{t:t+H}^{(m)}\}_{m=1}^M \leftarrow \text{MPPI}$ 
        for  $m = 1, \dots, M$  do
            if  $m = m^*$  then
                 $\mathcal{D}_{\text{RL}} \leftarrow$  1-step data
            else
                 $V_{\text{target}}(s_t) \leftarrow$  GAE value target Schulman et al. (2015)
                 $\mathcal{D}_{\text{MPPI}} \leftarrow$  1-step data includes  $V_{\text{target}}(s_t)$ 
            end
        end
    end
    if  $i \bmod N_{\text{upd}} = 0$  then
        Sample minibatches  $\mathcal{B}_{\text{RL}}^{(B)} \subset \mathcal{D}_{\text{RL}}, \mathcal{B}_{\text{MPPI}}^{(B)} \subset \mathcal{D}_{\text{MPPI}}$ 
        Calculate  $\rho_t$  via Alg. 2
         $L \leftarrow (1 - \rho_t)L_{\text{RL}} + \rho_t L_{\text{MPPI}}$ 
        Update  $\theta \leftarrow \theta - \alpha \nabla_\theta L$ 
    end
end
return  $\theta$  (or best on validation)
    
```

introduce selection bias on our high-level policy π_ϕ , we sample m uniformly from the set, then apply it to the real environment. We store the resulting tuple in the environment buffer, \mathcal{D}_{RL} :

$$\left(s_t, a_t^{(m^*)}, u_t^{(m^*)}, r_t^{(m^*)}, s_{t+1}^{(m^*)}, \log \pi_{\text{old}}(a_t^{(m^*)} | s_t) \right). \quad (5)$$

Here, π_{old} denotes the PPO policy used to generate the sampled high-level action before the update.

Sampled candidates m We re-score samples from the remaining candidates $m \neq m^*$ before storing to a buffer. To diversify the buffer, we do not include the samples for the selected candidate. As confirmed by experimentation, including m^* did not alter the results significantly. We first define virtual transitions using a reward approximator $\hat{r}(x, u)$ and MPPI dynamics model \hat{P} . Next, we assume the existence of an state interface map, $\psi : \mathcal{X} \mapsto \mathcal{S}$, which allows conversion from MPPI states to RL states (e.g. mapping a set of physical states to a set of lower-dimensional observations). We then use this map to accumulate samples

$$\hat{r}_t^{(m)} = \hat{r}(x_t, u_t^{(m)}), \quad \hat{s}_{t+1}^{(m)} = \psi \left(\hat{P}(x_t, u_t^{(m)}) \right). \quad (6)$$

Finally, we store in an MPPI buffer, $\mathcal{D}_{\text{MPPI}}$, tuples of the form

$$\left(s_t, a_t^{(m)}, u_t^{(m)}, \hat{r}_t^{(m)}, \hat{s}_{t+1}^{(m)}, \log \pi_{\text{old}}(a_t^{(m)} | s_t) \right). \quad (7)$$

4.2. Dataset Mixing

Given that our data are maintained in two buffers, we can utilize both for sampling training batches as well as for defining a composite loss function, with each component derived from one of the buffers. In the limit of large sample sizes, these two formulations are equivalent.

Let $\rho \in [0, 1]$ be a sample-based influence ratio between a buffer of real data \mathcal{D}_{RL} and virtual data $\mathcal{D}_{\text{MPPI}}$. At each step t , we sample $z_t \sim \text{Bernoulli}(\rho)$ such that

$$d_t = \begin{cases} d_{\text{env}} \sim \mathcal{D}_{\text{RL}} & z_t = 0 \\ d_{\text{virt}} \sim \mathcal{D}_{\text{MPPI}} & z_t = 1 \end{cases} \quad (8)$$

This induces a unified replay buffer with convex sampling distribution

$$s, a \sim (1 - \rho)\mathcal{D}_{\text{RL}} + \rho\mathcal{D}_{\text{MPPI}}. \quad (9)$$

The equivalent PPO loss can be expressed as:

$$L_{\text{PPO}}(\phi) = (1 - \rho) \mathbb{E}_{d_t \sim \mathcal{D}_{\text{RL}}} [L_{\text{PPO}}(\phi; d_t)] + \rho \mathbb{E}_{d_t \sim \mathcal{D}_{\text{MPPI}}} [L_{\text{PPO}}(\phi; d_t)]. \quad (10)$$

In Alg. 1, we employ loss function weighting of (10). Experiments in the Supplement confirm that the direct mixing scheme of (8) is largely consistent with the loss-based weighting in (10).

As opposed to the current PPO practice of filling a buffer of on-policy data before invoking state updates, our scheme allows for a more frequent updating process. Consider the case where policy updates occur after an experience buffer of N_B is filled. In standard PPO, this requires N_B environment steps. In our approach, assuming M MPPI samples are obtained per RL step, mixing virtual rollouts can allow for an effective reduction to N_B/M environment steps.

4.3. Bias in RL-MPC

Under the proposed approach, we bound the gap between the optimal value $V_{\pi^*, \mathcal{M}}$ and the estimated value $\hat{V}_{\pi, \mathcal{M}_\rho}$ of the idealized compositional policy under the mixed MDP, measuring the optimality under the approach, with proof in Appendix A.

Theorem 1 *Let π be the policy of the combined RL-MPC approach and π^* be the optimal policy under the true MDP, and take the bounds from Assumption 1. Let \hat{V} be a value function estimate. We can bound the value function error using the proposed approach to the value obtained from the optimal policy in the true domain according to:*

$$\begin{aligned} \left\| V_{\pi^*, \mathcal{M}}(s) - \hat{V}_{\pi, \mathcal{M}_\rho}(s) \right\|_\infty &\leq \rho \alpha_P H \gamma \frac{1 - \gamma^{H-1}}{1 - \gamma} \frac{\text{span}(r)}{2} + \rho \alpha_r \frac{1 - \gamma^H}{1 - \gamma} \\ &+ \gamma^H \varepsilon_V + \frac{4R_{\max} (1 - \gamma^H)}{(1 - \gamma)^2} \cdot \max_s D_{TV}(\pi^*(\cdot | s) \| \pi(\cdot | s)) \end{aligned} \quad (11)$$

where $\text{span}(r) := \sup(r) - \inf(r)$, $\varepsilon_V := \sup_s |V(s) - \hat{V}(s)|$ and $R_{\max} := \sup_{s,a} |r(s, a)|$.

The result highlights three separable contributors to error: model mismatch due to \mathcal{M}_ρ , value approximation error from the terminal bootstrap, and a policy sub-optimality gap. The first vanishes via appropriate adaptation of ρ , while the remaining terms vanish with training. Moreover, as discussed in Corollary 3 (Appendix A), finite MPPI samples introduce an additional resampling bias that does not decay with ρ ; this further motivates annealing ρ to reduce the weight placed on MPPI-derived estimates.

Algorithm 2: Online weight update from ensemble uncertainty

Input: $V \in \mathbb{R}^{B \times D}$ (batch \times ensemble), previous EMA $\Omega_t \geq 0$, previous weight $\rho_t \geq 0$
Parameters: ensemble size D , batch size B , smoothing factor $\lambda \in [0, 1)$, ρ_0, Ω_0

```

 $\mu_b \leftarrow \frac{1}{D} \sum_{d=1}^D V_{:,d}$  // per-sample mean across ensemble
 $\sigma_b^2 \leftarrow \frac{1}{D} \sum_{d=1}^D (V_{:,d} - \mu_b)^2$  // per-sample ensemble variance ( $\in \mathbb{R}^B$ )
 $\bar{\sigma}^2 \leftarrow \frac{1}{B} \sum_{b=1}^B \sigma_b^2$  // aggregate uncertainty over batch
 $\omega \leftarrow \frac{1}{1 + \bar{\sigma}^2}$  // confidence score
 $\Omega_{t+1} \leftarrow \lambda \Omega_t + (1 - \lambda) \omega$  // EMA
 $\eta \leftarrow (1 - \lambda) \Omega_{t+1}$  // decay rate
 $\rho_{t+1} \leftarrow \max(0, \rho_t(1 - \eta))$  // non-negativity clamp
return  $\Omega_{t+1}, \rho_{t+1}$ 

```

4.4. Adaptive Influence Ratio

While virtual data can improve exploration and data efficiency, the model terms in Theorem 1 scale linearly with the influence ratio ρ , so we wish to drive ρ to zero. We adapt the influence ratio dynamically in Alg. 2 by using ensemble disagreement as a proxy for when the critic has learned enough that virtual data is no longer needed.

Similar to Chen et al. (2017); Peer et al. (2021), we estimate value uncertainty $\bar{\sigma}$ from an ensemble of value functions and construct a bounded confidence score $\omega_t = \frac{1}{1 + \bar{\sigma}_t^2}$. This score is smoothed via an exponential moving average (EMA) when applied to ρ , so that, as value-function uncertainty decreases, ρ_t is annealed toward zero. The monotonic decay ensures that the algorithm eventually relies entirely on real data for the model error terms in Theorem 1 to vanish. The smoothing parameter λ controls the decay rate: larger $\lambda \rightarrow 1$ yields slower, stable annealing that filters out transient fluctuations in ensemble variance, while smaller $\lambda \rightarrow 0$ makes ρ more responsive. The initial value ρ_0 sets the level of trust in the virtual data during early training. Higher ρ_0 encourages more aggressive use of MPPI rollouts when real data is scarce, at the cost of initial bias if the surrogate model is poor (Table 2 captures this tradeoff).

5. Experiments and Results

We evaluate control performance in three environments: (i) Acrobot (Sutton, 1995), (ii) Lunar Lander (Brockman et al., 2016), and (iii) Racing (CARLA) (DeCastro et al., 2025). In Acrobot, the environment dynamics and reward are intentionally simple, and the dynamics/reward models used by our method match the ground-truth environment exactly. In Lunar Lander, which is built on Box2D physics, our method uses approximate dynamics and reward models. In Racing (CARLA), we adopt an even coarser (more misspecified) dynamics and reward model. This setup creates a spectrum of model mismatch—from exact (Acrobot) to approximate (Lunar Lander) to highly-mismatched (Racing)—allowing us to assess robustness. Specifically, the RL action injected into MPPI is defined as target angles in Acrobot, and as target velocities in Lunar Lander and Racing.

Across all environments, we introduce a *Danger Zone* to increase task difficulty: these are randomly-generate unsafe regions that yield a large negative reward upon entry. See the supplement for further details (e.g., state / action spaces, rewards, and tasks).

We structure the analysis in two stages. First, we compare four methods to highlight inter-method differences: (i) PPO (baseline), (ii) SAC (baseline), (iii) PPO-MPPI with no virtual data

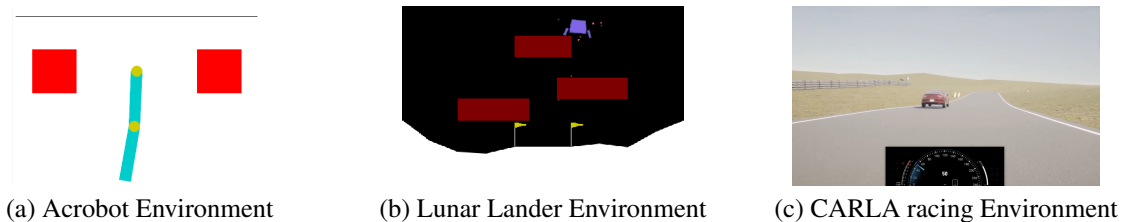


Figure 2: Experimental environments.

Table 1: Evaluation across four methods. Entries are *mean* \pm *std* over 50 episodes per each method. \uparrow / \downarrow indicate whether higher / lower equates to better performance, with **bold** being best.

Method	Acrobot			Lunar Lander			Racing		
	Success [%]	Step [\downarrow]	Reward [\uparrow]	Success [%]	Dist. to Goal [\downarrow]	Reward [\uparrow]	Finish [%]	Coll./Off-track [\downarrow]	Reward [\uparrow]
PPO	0.40	332.6	-347.1	0.00	1.30	-141.9	0.24	0.76	2710.2
	± 0.49	± 205.1	± 188.4	± 0.00	± 0.35	± 84.9	± 0.43	± 0.43	± 2693.8
SAC	0.00	500.0	-500.0	0.16	0.89	-64.3	0.30	0.50	1177.5
	± 0.00	± 0.0	± 0.0	± 0.37	± 0.40	± 188.4	± 0.46	± 0.50	± 4484.1
PPO-MPPI ($\rho = 0.0$)	1.00	82.6	-92.6	0.36	0.43	-8.3	0.14	0.86	2856.3
	± 0.00	± 22.9	± 40.1	± 0.48	± 0.29	± 227.5	± 0.35	± 0.35	± 1465.9
PPO-MPPI ($\rho = 0.3$)	1.00	90.7	-103.1	0.46	0.40	44.2	0.74	0.18	4512.9
	± 0.00	± 24.3	± 40.0	± 0.50	± 0.35	± 215.5	± 0.44	± 0.38	± 2516.0

($\rho = 0.0$), and (iv) PPO-MPPI with a fixed virtual-data mixing ratio ($\rho = 0.3$). Second, within the proposed method, we study the effect of the mixing ratio by varying ρ across fixed values and by using the adaptation scheme defined in Algorithm 2. This two-step design isolates the contributions of planning and virtual-data mixing, allowing us to examine how the choice of ρ influences outcomes under different levels of model mismatch.

5.1. Comparison Result of Methods

We observe a consistent trend across three environments (Table 1). The baseline **PPO** and **SAC** frequently explores within the Danger Zone during early training and subsequently collapses to behaviors that remain near the initial state. As a result, task success—measured by success/finish rates—remains low, indicating entrapment in local optimal. Augmenting with **MPPI** ($\rho = 0.0$) allows us to introduce soft constraints in the planner, leading from the outset to exploratory trajectories that repeatedly avoid the Danger Zone; in Acrobot, this yields a 100% success rate. However, in the remaining environments, success rates drop dramatically. By contrast, applying virtual-data mixing at $\rho = 0.3$ allows for exploration along the contour and within the interior of the Danger Zones. The results confirm that collecting multiple samples per step increases coverage of boundary states, alleviates data imbalance, and leads to higher success/finish rates than the other methods.

5.2. Comparison Result of Influence Ratio ρ

We next present results when the influence ratio is adapted, as detailed in Table 2. In the Acrobot domain, the final performance was nearly indistinguishable across settings (i.e., performance saturates nearly equally across all setups / parameters). In the Lunar Lander domain, however, the best result was achieved with the fixed setting $\rho = 0.5$, with performance degrading as ρ is increased to $\rho = 0.8$. In Racing, the best results was achieved with the *adaptive influence ratio* $\rho_0 = 0.3$, $\lambda = 0.98$, yielding better results than any fixed setting. This suggests that, in Racing,

Table 2: Evaluation across five influence ratios. Entries are *mean* \pm *std* over 50 episodes per each method. \uparrow / \downarrow indicate whether higher / lower equates to better performance, with **bold** being best.

Method	Acrobot			Lunar Lander			Racing		
	Success Rate [\uparrow]	Step [\downarrow]	Reward [\uparrow]	Success Rate [\uparrow]	Dist. to Goal [\downarrow]	Reward [\uparrow]	Finish Rate [\uparrow]	Coll/Off-track [\downarrow]	Reward [\uparrow]
PPO-MPPI ($\rho = 0.3$)	1.00	90.7	-103.1	0.46	0.40	44.2	0.74	0.18	4512.9
	± 0.00	± 24.3	± 40.0	± 0.50	± 0.35	± 215.5	± 0.44	± 0.38	± 2516.0
PPO-MPPI ($\rho = 0.5$)	1.00	88.3	-99.3	0.72	0.15	149.8	0.50	0.40	3774.9
	± 0.00	± 27.1	± 43.5	± 0.45	± 0.16	± 132.3	± 0.50	± 0.49	± 2669.2
PPO-MPPI ($\rho = 0.8$)	1.00	82.5	-96.4	0.06	0.16	-232.3	0.44	0.28	2114.9
	± 0.00	± 21.7	± 42.7	± 0.24	± 0.11	± 155.3	± 0.50	± 0.45	± 3350.9
PPO-MPPI ($\rho_0 = 0.3, \lambda = 0.99$ $\lambda = 0.98$ in Racing)	1.00	83.8	-90.3	0.42	0.54	25.9	0.84	0.16	5251.5
	± 0.00	± 23.7	± 36.0	± 0.49	± 0.52	± 202.6	± 0.37	± 0.37	± 2045.8
PPO-MPPI ($\rho_0 = 0.5, \lambda = 0.98$ $\lambda = 0.95$ in Racing)	1.00	96.6	-115.3	0.34	0.48	31.0	0.60	0.34	4336.8
	± 0.00	± 37.6	± 62.5	± 0.47	± 0.30	± 162.7	± 0.49	± 0.47	± 2130.6

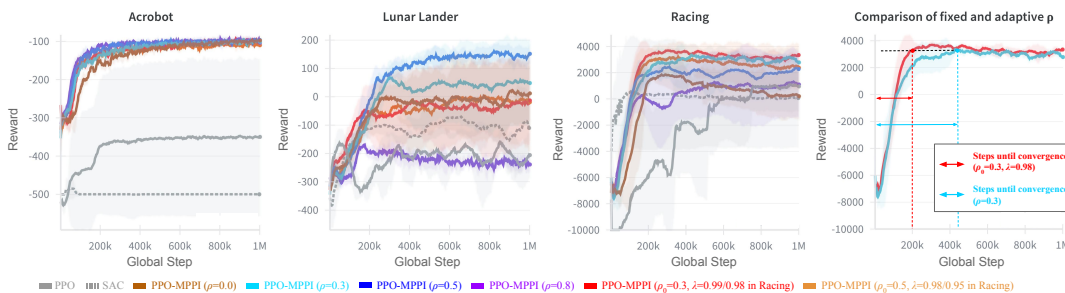


Figure 3: Three figures on the left: Episode reward of each environment. averaged over 5 seeds. Rightmost figure: Episode reward for PPO-MPPI($\rho = 0.3$ v.s. $\rho_0 = 0.3, \lambda = 0.98$) in Racing environment.

where model mismatch is pronounced, adapting ρ mitigates model errors and improves performance. Compared to the fixed setting ($\rho = 0.3$), an adaptive schedule initialized at $\rho_0 = 0.3$ converged faster than the remainder of the cases, reaching the highest reward achieved by the fixed setting in $2.1\times$ fewer training steps. In turn, when dynamics/reward models are misspecified, an environment-dependent optimal ρ exists. See Fig. 3 for training curves of each of the methods.

6. Conclusion

We introduce a shared training framework that unifies the high-level reasoning capabilities of value-based RL and low-level, trajectory-based reasoning of MPPI. We develop training scheme that leverages a bi-directional information flow between the two strategies: the RL component guides sampling, while MPPI-generated rollouts accelerate value learning and policy improvement. We establish a bound on the value-function error that explicitly accounts for model approximation and sampling-induced bias, and propose a training algorithm featuring an adaptive influence ratio that dynamically regulates the contribution of real and virtual data based on ensemble uncertainty. Our results across various environments underscore the role of adaptivity in mitigating model bias.

Promising future extensions could be in exploring how mixtures of real and imagined rollouts can benefit model-based or latent-world model settings (e.g., Dreamer, TD-MPC) in terms of sample efficiency and bias. Additionally, use of uncertainty-aware constraints could benefit human-interactive and safety-critical settings more generally.

Acknowledgments

Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of TRI or any other Toyota entity.

References

- Felix Berkenkamp, Matteo Turchetta, Angela P Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in Neural Information Processing Systems*, 2017.
- M. Bhardwaj, Ankur Handa, D. Fox, and Byron Boots. Information theoretic model predictive q-learning, 2019.
- M. Bhardwaj, Sanjiban Choudhury, and Byron Boots. Blending MPC & value function approximation for efficient reinforcement learning, 2020a.
- Mohak Bhardwaj, Ankur Handa, Dieter Fox, and Byron Boots. Information theoretic model predictive q-learning. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pages 840–850. PMLR, 10–11 Jun 2020b.
- Greg Brockman, Vicki Cheung, Ludwig Pettersson, et al. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.
- Richard Y Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. Ucb exploration via q-ensembles. *arXiv preprint arXiv:1706.01502*, 2017.
- Jonathan DeCastro, Andrew Silva, Deepak Gopinath, Emily Sumner, Thomas Matrai Balch, Laporsha Dees, and Guy Rosman. Dreaming to assist: Learning to align with human objectives for shared control in high-speed racing. In *Conference on Robot Learning*, pages 2599–2628. PMLR, 2025.
- Javier Garcia and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.
- Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Dreamer: Reinforcement learning with latent world models. In *International Conference on Learning Representations*, 2020.
- Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR) 2024, Spotlight*, 2024. Preprint arXiv:2310.16828.
- Zhang-Wei Hong, Joni Pajarinen, and Jan Peters. Model-based lookahead reinforcement learning. *arXiv preprint arXiv:1908.06012*, 2019.
- Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019.
- Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the nineteenth international conference on machine learning*, pages 267–274, 2002.
- B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE transactions on intelligent transportation systems*, 23(6):4909–4926, 2021.
- Prakrut Kotecha, Ganga Nair B, and Shishir Kolathaya. Real-time gait adaptation for quadrupeds using model predictive control and reinforcement learning. *arXiv preprint arXiv:2510.20706*, 2025.

- Yankai Li and Mo Chen. Unifying model predictive path integral control, reinforcement learning, and diffusion models for optimal control and planning. *arXiv preprint arXiv:2502.20476*, 2025.
- Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac Gym: High performance GPU-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.
- Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 16(1):1–118, 2023. ISSN 1935-8237. doi: 10.1561/22000000086.
- Vedant Mundheda, Zhouchonghao Wu, and Jeff Schneider. Teacher-guided off-road autonomous driving. In *ML4AD*, Philadelphia, PA, USA, 2025.
- Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model predictive control. In *IEEE International Conference on Robotics and Automation*, 2018.
- Khang Nguyen, Khai Nguyen, An T Le, Jan Peters, Manfred Huber, Ngo Anh Vien, and Minh Nhat Vu. Td-grpc: Temporal difference learning with group relative policy constraint for humanoid locomotion. *arXiv preprint arXiv:2505.13549*, 2025.
- Masashi Okada and Tadahiro Taniguchi. Variational inference MPC for Bayesian model-based reinforcement learning. In *Conference on robot learning*, pages 258–272. PMLR, 2020.
- Oren Peer, Chen Tessler, Nadav Merlis, and Ron Meir. Ensemble bootstrapping for q-learning. In *International conference on machine learning*, pages 8454–8463. PMLR, 2021.
- Philip Polack, Florent Alché, Brigitte d’Andréa Novel, and Arnaud de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE intelligent vehicles symposium (IV)*, pages 812–818. IEEE, 2017.
- Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL <http://jmlr.org/papers/v22/20-1364.html>.
- Rudolf Reiter, Andrea Ghezzi, Katrin Baumgärtner, Jasper Hoffmann, Robert D. McAllister, and Moritz Diehl. Ac4mpc: Actor-critic reinforcement learning for nonlinear model predictive control, 2024.
- Angel Romero, Elie Aljalbout, Yunlong Song, and Davide Scaramuzza. Actor-critic model predictive control: Differentiable optimization meets reinforcement learning. *arXiv preprint arxiv:2306.09852*, 2023a. URL <https://arxiv.org/abs/2306.09852>.
- Angel Romero, Yunlong Song, and D. Scaramuzza. Actor-critic model predictive control, 2023b.
- Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: a survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020.
- Dorsa Sadigh, Shankar Sastry, Sanjit A Seshia, and Anca D Dragan. Planning for autonomous cars that leverage interactions with humans. In *Robotics: Science and Systems*, 2018.
- John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

- Max Schwenzer, Muzaffer Ay, Thomas Bergs, and Dirk Abel. Review on model predictive control: An engineering perspective. *The International Journal of Advanced Manufacturing Technology*, 117(5):1327–1349, 2021.
- Jaek Shin, A. Hakobyan, Mingyu Park, Yeoneung Kim, Gihun Kim, and Insoon Yang. Infusing model predictive control into meta-reinforcement learning for mobile robots in dynamic environments, 2021.
- David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.
- Richard S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1995.
- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2020.
- Xu Wang, Sen Wang, Xingxing Liang, Dawei Zhao, Jincui Huang, Xin Xu, Bin Dai, and Qiguang Miao. Deep reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):5064–5078, 2024. doi: 10.1109/TNNLS.2022.3207346.
- Yubin Wang, Zengqi Peng, Yusen Xie, Yulin Li, Hakim Ghazzai, and Jun Ma. Learning the references of on-line model predictive control for urban self-driving. *CoRR*, abs/2308.15808, 2023. Also arXiv:2308.15808.
- Yuhang Wang, Hanwei Guo, Sizhe Wang, Long Qian, and Xuguang Lan. Bootstrapped model predictive control, 2025.
- Zheng Wen, Doina Precup, Morteza Ibrahimi, Andre Barreto, Benjamin Van Roy, and Satinder Singh. On efficiency in hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 6708–6718, 2020.
- Grady Williams, Nolan Wagener, Brian Goldfain, P. Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information theoretic mpc for model-based reinforcement learning, 2017.
- Álvaro Serra-Gomez, Daniel Jarne Ornia, Dhruva Tirumala, and Thomas Moerland. A kl-regularization framework for learning to plan with adaptive priors. *arXiv preprint*, arXiv:2510.04280, 2025.

Appendix A. Proof of Theorem 1 and Additional Analyses

We note that there are two sources of bias; bias introduced by resampling rollouts from an MPPI buffer, and a bias due to resampling and re-scoring the rollouts.

First let the compositional policy $\tilde{\pi}(u_t | s_t) := \int \pi_{\text{MPPI}}(u_t | s_t; a) \pi_\phi(a | s_t) da$. We assume some environment model \mathcal{M} and some virtual model used in MPPI $\hat{\mathcal{M}}$. Given the interface map ψ and MPPI policy $\pi_{\text{MPPI}}(u_t | s_t; a)$, with slight abuse of notation, we can recast $\hat{r} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ and $\hat{P} : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$. Let $\pi : \mathcal{S} \mapsto \mathcal{A}$ be any policy. We first introduce a Simulation Lemma to bound the error in value due to the approximations \hat{r} and \hat{P} .

Lemma 2 (*H*-Step Simulation Lemma) *Assume ρ_t is a time-varying parameter controlling data sampled from the real environment and virtual samples $\mathcal{D}_t = (1 - \rho_t)\mathcal{D}_{\text{RL},t} + \rho_t\mathcal{D}_{\text{MPPI},t}$,*

$$\|V_{\pi, \mathcal{M}}(s) - V_{\pi, \mathcal{M}_\rho}(s)\|_\infty \leq \rho\alpha_P H \gamma \frac{1 - \gamma^{H-1}}{1 - \gamma} \frac{\text{span}(r)}{2} + \rho\alpha_r \frac{1 - \gamma^H}{1 - \gamma} \quad (12)$$

Proof Let

$$\begin{aligned} r_\rho(s, a) &:= (1 - \rho)r(s, a) + \rho\hat{r}(s, a), \\ P_\rho(\cdot | s, a) &:= (1 - \rho)P(\cdot | s, a) + \rho\hat{P}(\cdot | s, a) \end{aligned} \quad (13)$$

We obtain trajectories, each containing H samples. MPC approximates the truncated-horizon value function as follows:

$$V_{\pi, P}(s) = \mathbb{E}_{\pi, P} \left[\sum_{i=0}^{H-1} \gamma^i r(s_i, a_i) + \gamma^H V(s_H) \middle| s_0 = s \right] \quad (14)$$

We define an MDP $\mathcal{M}_\rho = \langle \mathcal{S}, \mathcal{A}, P_\rho, r_\rho, H, \gamma, \mu \rangle$. Assuming the actual state transitions come from the real environment \mathcal{M} , we can factor r_ρ, P_ρ independently as follows:

$$\begin{aligned} V_{\pi, \mathcal{M}}(s) - V_{\pi, \mathcal{M}_\rho}(s) &= V_{\pi, \mathcal{M}}(s) - \mathbb{E}_{\pi, P_\rho} \left[\sum_{i=0}^{H-1} \gamma^i r_\rho(s_i, a_i) + \gamma^H V(s_H) \right] \\ &= V_{\pi, \mathcal{M}}(s) - \mathbb{E}_{\pi, P_\rho} \left[\sum_{i=0}^{H-1} \gamma^i r(s_i, a_i) + \gamma^H V(s_H) \right] \\ &\quad + \mathbb{E}_{\pi, P_\rho, \mu} \left[\sum_{i=0}^{H-1} \gamma^i r(s_i, a_i) + \gamma^H V(s_H) \right] - \mathbb{E}_{\pi, P_\rho, \mu} \left[\sum_{i=0}^{H-1} \gamma^i r_\rho(s_i, a_i) + \gamma^H V(s_H) \right] \end{aligned} \quad (15)$$

Applying the triangle inequality yields:

$$\begin{aligned} &\|V_{\pi, \mathcal{M}}(s) - V_{\pi, \mathcal{M}_\rho}(s)\|_\infty \\ &\leq \left\| \sum_{s \in \{s_0, \dots, s_{H-1}\}} (P(s) - P_\rho(s)) \left(\sum_{i=0}^{H-1} \gamma^i r(s_i, a_i) \right) + \gamma^H V(s_H) \right\|_\infty \\ &\quad + \left\| \mathbb{E}_{\pi, P_\rho, \mu} \left[\sum_{i=0}^{H-1} \gamma^i (r(s_i, a_i) - r_\rho(s_i, a_i)) \right] \right\|_\infty \\ &\leq \rho\alpha_P H \gamma \frac{1 - \gamma^{H-1}}{1 - \gamma} \frac{\text{span}(r)}{2} + \rho\alpha_r \frac{1 - \gamma^H}{1 - \gamma} \end{aligned} \quad (16)$$

Where we apply the uniform bounds $|r(s, a) - r_\rho(s, a)| \leq \alpha_r$ and $|P(s, a) - P_\rho(s, a)| \leq \alpha_P$, and $\text{span}(f) := \sup(f) - \inf(f)$. ■

Next, we prove Theorem 1.

Proof Our goal is to bound the following triangle inequality:

$$\left\| V_{\pi^*, \mathcal{M}} - \hat{V}_{\tilde{\pi}, \mathcal{M}_\rho} \right\|_\infty \leq \underbrace{\left\| V_{\pi^*, \mathcal{M}} - V_{\pi^*, \mathcal{M}_\rho} \right\|_\infty}_{\text{Lemma 2}} + \underbrace{\left\| V_{\pi^*, \mathcal{M}_\rho} - \hat{V}_{\pi^*, \mathcal{M}_\rho} \right\|_\infty}_{\text{Value approximation error}} + \underbrace{\left\| \hat{V}_{\pi^*, \mathcal{M}_\rho} - \hat{V}_{\tilde{\pi}, \mathcal{M}_\rho} \right\|_\infty}_{\text{policy gap}} \quad (17)$$

We find bounds for the last two terms.

Value approximation error. We first bound the approximation error of the terminal value function \hat{V} to the true value function V .

Let $\varepsilon_V := \sup_s |V(s) - \hat{V}(s)|$

$$\begin{aligned} V_{\pi, \mathcal{M}}(s) - \hat{V}_{\pi, \mathcal{M}_\rho}(s) &= V_{\pi, \mathcal{M}}(s) - \mathbb{E}_{\pi, P_\rho} \left[\sum_{i=0}^{H-1} \gamma^i r_\rho(s_i, a_i) + \gamma^H \hat{V}(s_H) \right] \\ &= \gamma^H \left| \mathbb{E} \left[V(s_H) - \hat{V}(s_H) \right] \right| \leq \gamma^H \varepsilon_V \end{aligned} \quad (18)$$

Policy gap error. Our goal is to bound the error under due to the difference in policy $\tilde{\pi}$ from the optimal one, π^* .

First, let

$$\hat{V}_{\pi, \mathcal{M}}(s) = \mathbb{E}_{\pi, \mathcal{M}} \left[\sum_{i=0}^{H-1} \gamma^i r_\rho(s_i, a_i) + \gamma^H \hat{V}(s_H) \mid s_0 = s \right] \quad (19)$$

and define the advantage function as

$$\hat{A}_{\pi, \mathcal{M}}(s, a) = \hat{Q}_{\pi, \mathcal{M}}(s, a) - \hat{V}_{\pi, \mathcal{M}}(s) \quad (20)$$

where

$$\hat{Q}_{\pi, \mathcal{M}}(s, a) = r_\rho(s, a) + \gamma \mathbb{E}_{s' \sim P_\rho(\cdot | s, a)} \left[\hat{V}_{\pi, \mathcal{M}}(s') \right] \quad (21)$$

Using the Performance Difference Lemma [Kakade and Langford \(2002\)](#), we can characterize the difference in value functions of two policies π_1 and π_2 as equivalent to the sum of advantages of π_2 over time when actions are sampled from π_1 :

$$\hat{V}_{\pi_1, \mathcal{M}_\rho}(s_0) - \hat{V}_{\pi_2, \mathcal{M}_\rho}(s_0) = \mathbb{E}_{\pi_1, P_\rho} \left[\sum_{i=0}^{H-1} \gamma^i \hat{A}_{\pi_2, \mathcal{M}_\rho}(s_i, a_i) \mid s_0 = s \right] \quad (22)$$

To find a bound for $\left\| \hat{V}_{\pi^*, \mathcal{M}_\rho}(s) - \hat{V}_{\tilde{\pi}, \mathcal{M}_\rho}(s) \right\|_\infty$, we use the property that the advantage over π has expectation zero when sampled over π , i.e. $\mathbb{E}_{a \sim \pi(\cdot | s)} \left[\hat{A}_{\pi, \mathcal{M}_\rho}(s, a) \right] = 0$. Hence,

$$\left| \mathbb{E}_{a \sim \pi^*(\cdot | s)} \left[\hat{A}_{\tilde{\pi}, \mathcal{M}_\rho}(s, a) \right] \right| = \left| \mathbb{E}_{a \sim \pi^*(\cdot | s)} \left[\hat{A}_{\tilde{\pi}, \mathcal{M}_\rho}(s, a) \right] - \mathbb{E}_{a \sim \tilde{\pi}(\cdot | s)} \left[\hat{A}_{\tilde{\pi}, \mathcal{M}_\rho}(s, a) \right] \right| \quad (23)$$

From the definition of total variation, we have, for bounded f :

$$|\mathbb{E}_{a \sim \pi_1} [f(a)] - \mathbb{E}_{a \sim \pi_2} [f(a)]| \leq 2 \|f(\cdot)\|_\infty \cdot D_{TV}(\pi_1(\cdot | s) \| \pi_2(\cdot | s)) \quad (24)$$

Thus,

$$\left| \mathbb{E}_{a \sim \pi^*(\cdot | s)} [\hat{A}_{\tilde{\pi}, \mathcal{M}_\rho}(s, a)] \right| \leq 2 \left\| \hat{A}_{\tilde{\pi}, \mathcal{M}_\rho}(s, a) \right\|_\infty \cdot D_{TV}(\pi_1(\cdot | s) \| \pi_2(\cdot | s)) \quad (25)$$

Now, expanding the telescoping sum in (22),

$$\begin{aligned} \left\| \hat{V}_{\pi^*, \mathcal{M}_\rho} - \hat{V}_{\tilde{\pi}, \mathcal{M}_\rho} \right\|_\infty &= \left\| \mathbb{E}_{\pi^*, P_\rho} \left[\sum_{i=0}^{H-1} \gamma^i \hat{A}_{\tilde{\pi}, \mathcal{M}_\rho}(s_i, a_i) \middle| s_0 = s \right] \right\|_\infty \\ &\leq \sum_{i=0}^{H-1} \gamma^i \left\| \mathbb{E}_{\pi^*, P_\rho} [\hat{A}_{\tilde{\pi}, \mathcal{M}_\rho}(s_i, a_i) | s_0 = s] \right\|_\infty \\ &\leq 2 \cdot \sum_{i=0}^{H-1} \gamma^i \left\| \hat{A}_{\tilde{\pi}, \mathcal{M}_\rho}(s_i, a_i) \right\|_\infty \cdot D_{TV}(\pi^*(\cdot | s_i) \| \tilde{\pi}(\cdot | s_i)) \\ &\leq 2 \cdot \frac{1 - \gamma^H}{1 - \gamma} \left\| \hat{A}_{\tilde{\pi}, \mathcal{M}_\rho}(s, a) \right\|_\infty \cdot \max_s D_{TV}(\pi^*(\cdot | s) \| \tilde{\pi}(\cdot | s)) \end{aligned} \quad (26)$$

To obtain a bound for $\left\| \hat{A}_{\tilde{\pi}, \mathcal{M}_\rho}(s, a) \right\|_\infty$, observe that

$$\left| \hat{A}_{\tilde{\pi}, \mathcal{M}_\rho}(s, a) \right| = \left| \hat{Q}_{\tilde{\pi}, \mathcal{M}_\rho}(s, a) - \hat{V}_{\tilde{\pi}, \mathcal{M}_\rho}(s) \right| \leq \left| \hat{Q}_{\tilde{\pi}, \mathcal{M}_\rho}(s, a) \right| + \left| \hat{V}_{\tilde{\pi}, \mathcal{M}_\rho}(s) \right| \quad (27)$$

Rewards are bounded by R_{\max} , and $\gamma < 1$, any value function satisfies $\|V\|_\infty \leq \frac{R_{\max}}{1-\gamma}$. Therefore,

$$\left\| \hat{V}_{\tilde{\pi}, \mathcal{M}_\rho}(s) \right\|_\infty \leq \frac{R_{\max}}{1-\gamma}, \quad \left\| \hat{Q}_{\tilde{\pi}, \mathcal{M}_\rho}(s, a) \right\|_\infty \leq \frac{R_{\max}}{1-\gamma} \quad (28)$$

Substituting into (26), we arrive at

$$\left\| \hat{V}_{\pi^*, \mathcal{M}_\rho} - \hat{V}_{\tilde{\pi}, \mathcal{M}_\rho} \right\|_\infty \leq \frac{4R_{\max}(1-\gamma^H)}{(1-\gamma)^2} \cdot \max_s D_{TV}(\pi^*(\cdot | s) \| \tilde{\pi}(\cdot | s)) \quad (29)$$

■

Theorem 1 analyzes the idealized compositional policy $\tilde{\pi}$, which assumes exact MPPI solutions. In practice, MPPI uses K finite samples, and the resulting approximation error can be quantified as follows.

Corollary 3 *Let $\tilde{V}_{\pi, \mathcal{M}_\rho}$ be the value that MPPI computes, and let*

$$g := \sum_{\tau=t}^{t+H-1} \gamma^{\tau-t} r(s_\tau, a_\tau) + \gamma^H \hat{V}(s_{t+H}). \quad (30)$$

It follows that:

$$\left\| V_{\pi^*, \mathcal{M}}(s) - \tilde{V}_{\pi, \mathcal{M}_\rho}(s) \right\|_\infty \leq \left\| V_{\pi^*, \mathcal{M}}(s) - \hat{V}_{\pi, \mathcal{M}_\rho}(s) \right\|_\infty + \text{span}(g) D_u \sqrt{\frac{H}{2} \lambda_{\max}(\Sigma^{-1})} \quad (31)$$

where $\lambda_{\max}(\Sigma^{-1})$ is the largest eigenvalue of Σ^{-1} , and $D_u := \sup_{u, u' \in \mathcal{U}} \|u - u'\|$

Proof We apply the triangle inequality. The first term of the bound comes from Theorem 1, with the second term being $\left\| \hat{V}_{\pi, \mathcal{M}_\rho}(s) - \tilde{V}_{\pi, \mathcal{M}_\rho}(s) \right\|_\infty$. To show the second term, we form a distributional / rescaling bound on the samples drawn by MPPI. Given two distributions q and q' and a function g , we can construct a bound

$$\mathbb{E}_q[g] - \mathbb{E}_{q'}[g] \leq \text{span}(g) \sup_{s \in \mathcal{S}} |q(s) - q'(s)| \leq \text{span}(g) \sqrt{\frac{1}{2} D_{KL}(q \| q')} \quad (32)$$

where the inequalities, respectively, follow from total variation and Pinsker's inequality.

Under the sampling rule in (3), a nominal control sequence $\bar{u}_{t:t+H-1}$ is perturbed by noise of the form $\varepsilon_\tau^i \sim \mathcal{N}(0, \Sigma)$. A weighted average of these perturbations are constructed at time t as:

$$u_t^{(m)} = \bar{u}_t + \sum_{k=1}^K \tilde{w}_k^{(m)} \varepsilon_t^{(k)} \quad (33)$$

For the chosen candidate, let the difference of candidate m 's sequence from the nominal sequence be defined as $\Delta u_t^{(m)} := u_t^{(m)} - \bar{u}_t$. Let

$$g_\pi(s_0) := \sum_{i=0}^{H-1} \gamma^i r(s_i, a_i) + \gamma^H \hat{V}(s_H) \quad (34)$$

Define two distributions $u_t \sim q_0$ and $u_t^{(m)} \sim q_m$, respectively, as the nominal distribution for u_t and the MPPI-sampled control $u_t^{(m)}$. Then, (32) gives:

$$\begin{aligned} \mathbb{E}_{q_m}[g_\pi(s_0)] - \mathbb{E}_{q_0}[g_\pi(s_0)] &\leq \text{span}(g_\pi(s_0)) \sqrt{\frac{1}{2} D_{KL}(q_m \| q_0)} \\ &= \text{span}(g_\pi(s_0)) \sqrt{\frac{1}{2} D_{KL}(\mathcal{N}(\Delta u_\tau^{(m)}, \Sigma) \| \mathcal{N}(0, \Sigma))} \\ &= \text{span}(g_\pi(s_0)) \sqrt{\frac{1}{2} \sum_{\tau=t}^{t+H-1} \left(\Delta u_\tau^{(m)} \right)^T \Sigma^{-1} \Delta u_\tau^{(m)}} \end{aligned} \quad (35)$$

Note that the above bound is empirical, requiring a set of MPPI samples. To find a non-empirical bound, we can observe the following property:

$$\left(\Delta u_\tau^{(m)} \right)^T \Sigma^{-1} \Delta u_\tau^{(m)} \leq \lambda_{\max}(\Sigma^{-1}) \|\Delta u_\tau^{(m)}\|^2 \leq \lambda_{\max}(\Sigma^{-1}) D_u^2 \quad (36)$$

where $\lambda_{\max}(\cdot)$ is the largest eigenvalue of a matrix, and diameter $D_u := \sup_{u, u' \in \mathcal{U}} \|u - u'\|$. Hence,

$$\sum_{\tau=t}^{t+H-1} \left(\Delta u_\tau^{(m)} \right)^T \Sigma^{-1} \Delta u_\tau^{(m)} \leq H \lambda_{\max}(\Sigma^{-1}) D_u^2 \quad (37)$$

■

It is worth noting that the bound within Corollary 3 does not decay as modeling error improves. However, the corollary makes it clear that increasing K (more MPPI samples) or decreasing Σ tightens this term, giving concrete knobs to reduce the gap between the idealized and implemented algorithms.

Appendix B. Domain Details

For each domain, we introduce relevant details for the environment and provide further details about approximations and choices used for MPPI.

B.1. Acrobot

Task. Swing up the end effector of a two-link underactuated pendulum to a target height by applying a continuous torque at the actuated joint, while avoiding a pre-specified set of danger zones.

Observation space. A 6-dimensional continuous vector $[\cos \theta_1, \sin \theta_1, \cos \theta_2, \sin \theta_2, \dot{\theta}_1, \dot{\theta}_2]$, where θ_1 is the absolute angle of the first link and θ_2 is the angle of the second link relative to the first. In addition, the observation includes the *DangerZone* parameters (x_D, y_D, s_D) , which specify the center coordinates and the side length of a square danger region (in the same coordinate frame as x, y).

Action space. A scalar continuous torque command at the actuated joint, $u \in [-1, 1]$ N m.

Reward.

- -1 per time step until termination (shorter episodes yield higher return).
- Additional -50 per time step while the agent is inside the *DangerZone*.

Termination.

- Goal height achieved: $-\cos \theta_1 - \cos(\theta_1 + \theta_2) > 1.0$.
- Otherwise, the episode is truncated at a fixed horizon (e.g., 500 steps).

Dynamics Model and reward model for MPPI.

- Dynamics model: exactly same as the environment’s dynamics.
- Reward model: exactly same as the environment’s reward structure.

B.2. LunarLander

Task. Control a lunar lander to achieve a soft touchdown at the center of a landing pad while minimizing fuel use and impact velocity, while avoiding a randomly-selected set of danger zones.

Observation space. The observation consists of the 8-dimensional state $[x, y, \dot{x}, \dot{y}, \phi, \dot{\phi}, \text{leg_left}, \text{leg_right}]$, where the last two entries are binary flags indicating ground contact for the left and right legs, augmented with the *DangerZone* parameters (x_D, y_D, w_D, h_D) that give the center coordinates and the width/height of a rectangular danger region (in the same coordinate frame as x, y). Note that the *DangerZones* in this domain are randomized.

Action space. A 2-D continuous control vector $u = [u_{\text{main}}, u_{\text{lateral}}] \in [-1, 1]^2$. The first element controls the main engine throttle (values ≤ 0 are treated as off), and the second controls the lateral thrusters; its sign indicates left (< 0) or right (> 0) firing.

Reward.

- Increased as the lander gets closer to the center of the landing pad, and decreased as it moves farther away.
- Increased as the lander’s translational velocity becomes slower, and decreased as it moves faster.
- Decreased the more the lander is tilted (i.e., larger absolute tilt angle $|\phi|$).
- Increased by +10 points for each leg in ground contact.
- Decreased by 0.03 points per time step while any side engine is firing.
- Decreased by 0.3 points per time step while the main engine is firing.
- Additional -100 for crashing and $+100$ for a safe landing (episode end).
- Additional -5 per time step while the agent is inside the *DangerZone*.

Termination.

- Crash (lander body contacts the surface).
- Leaving the viewport bounds.
- Lander is not awake (comes to rest).
- Otherwise, the episode is truncated at a fixed horizon (maximum number of steps).

Dynamics Model and reward model for MPPI.

- Dynamics model: We omit Box2D’s complex rigid-body calculations and instead update the vehicle’s position and orientation using fixed constants.
- Reward model: We exclude the terminal reward from the one in the environment. Terminal costs in this domain tend to outweighs the dense reward signals.

B.3. Racing

Task. Drive along a designated track to reach the goal as quickly as possible while avoiding boundary violations and collisions with other vehicles, and avoid pre-specified danger zones.

Observation space. A concatenation of the following components:

- Vehicle state: planar position x, y , speed v (and optionally heading ψ , yaw rate r , etc.).
- TrackState: boundary deviation coefficient b (normalized lateral offset from the track center-line) and heading error $\Delta\psi$ between the vehicle heading and the track tangent.
- TrackPoint: local boundary information (e.g., a fixed-length set of nearby boundary points or left/right distances) expressed in the vehicle frame.
- DangerZone: center coordinates (x_D, y_D) and side length s_D of a square danger region.

Action space. Steering and throttle commands $u = [u_{\text{steer}}, u_{\text{throt}}] \in [-1, 1]^2$.

Reward.

- Passing reward: increased with the vehicle’s relative speed to a nearby *OtherVehicle* when overtaking (i.e., larger forward relative velocity yields larger reward).
- Progress reward: proportional to the forward arc-length progress along the track centerline.
- Boundary penalty: decreased according to the magnitude of the boundary deviation coefficient b .
- Additional -100 upon exceeding boundary limits (hard off-track violation).
- Additional -1000 upon collision with another vehicle.
- Additional $+1000$ upon reaching the goal.
- Additional -150 per time step while the agent is inside the *DangerZone*.

Termination.

- Exceeding boundary limits (off-track).
- Collision with another vehicle.
- Goal reached.
- Otherwise, the episode is truncated at a fixed time limit (maximum number of steps).

Dynamics Model and reward model for MPPI.

- Dynamics model: We apply the kinematic bicycle model (Polack et al. (2017)).
- Reward model: We exclude the terminal reward from the one in the environment.

Appendix C. Architecture Details

We adopt Stable-Baselines3 (SB3) Raffin et al. (2021) for our reinforcement learning implementations; the hyperparameters are listed in Tables 3 and 4. For MPPI, we use `pytorch-mppi`; the hyperparameters are listed in Table 5. The MPPI running cost is decomposed as

$$J_i = \sum_{\tau=t}^{t+H-1} \left(w_{\text{RL}} J_{\text{RL}}(\hat{x}_{\tau}^i, u_{\tau}^i; a_t) + w_{\text{D}} J_{\text{danger}}(\hat{x}_{\tau}^i) + J_{\text{other}}(\hat{x}_{\tau}^i, u_{\tau}^i) \right), \quad (38)$$

where $w_{\text{RL}}, w_{\text{D}} \in \mathbb{R}_{\geq 0}$ are fixed weights.

RL term. Three common forms are used for the RL term. (i) is used for the results in Section 5, and (ii), (iii) are used for the results in Section D.

(i) *Target-tracking form:*

$$J_{\text{RL}}^{\text{track}}(\hat{x}_\tau^i; a_t) = \|\hat{x}_\tau^i - x^*(a_t)\| \quad (39)$$

In Acrobot, $x^*(a_t) = [\theta_1^*, \theta_2^*]$, and in Lunar Lander and Racing, $x^*(a_t) = [\dot{x}^*, \dot{y}^*]$.

(ii) *Quadratic (QP-like) form:*

$$J_{\text{RL}}^{\text{quad}}(\hat{x}_\tau^i, u_\tau^i; a_t) = \begin{bmatrix} \hat{x}_\tau^i \\ u_\tau^i \end{bmatrix}^\top Q_\tau(a_t) \begin{bmatrix} \hat{x}_\tau^i \\ u_\tau^i \end{bmatrix} + p_\tau^\top(a_t) \begin{bmatrix} \hat{x}_\tau^i \\ u_\tau^i \end{bmatrix}, \quad (40)$$

where (Q_τ, p_τ) or the reference $x^*(a_t)$ can be viewed as quantities parameterized by the RL output a_t .

(iii) *Value function terminal cost form:*

$$J_{\text{RL}}^{\text{V}}(\hat{x}_H^i) = -V(\hat{x}_H^i) \quad (41)$$

Note that we can combine (iii) with either (i) or (ii).

Danger-zone term. Let the axis-aligned danger zone be the rectangle centered at (x_D, y_D) with width W and height H : $\mathcal{D} = \{(x, y) : |x - x_D| \leq W/2, |y - y_D| \leq H/2\}$ (the square case is $W = H$). We use a binary indicator:

$$J_{\text{danger}}(\hat{x}_\tau^i) = \begin{cases} 1, & (x(\hat{x}_\tau^i), y(\hat{x}_\tau^i)) \in \mathcal{D}, \\ 0, & \text{otherwise.} \end{cases} \quad (42)$$

Other term. Below we specify the instances used in each domain.

Lunar Lander. We penalize altitude from the pad ($y = 0$ at the pad) and control effort:

$$J_{\text{other}}^{\text{LL}}(\hat{x}_\tau^i, u_\tau^i) = w_y (y(\hat{x}_\tau^i))^2 + w_{\text{act}} \|u_\tau^i\|_2^2. \quad (43)$$

Racing. We combine a collision indicator with a boundary-violation penalty. Let $p(\hat{x}) = (x(\hat{x}), y(\hat{x}))$ be the ego position, $p_{\tau, \text{opponent}}^i$ the opponent position, and d_{contact} the contact threshold (accounting for vehicle sizes). Define the collision cost

$$J_{\text{coll}}(\hat{s}_\tau^i) = \begin{cases} 1, & \|p(\hat{x}_\tau^i) - p_{\tau, \text{opponent}}^i\|_2 \leq d_{\text{contact}} \\ 0, & \text{otherwise.} \end{cases} \quad (44)$$

and the boundary proportion $\text{BP}(\hat{x})$ as the radial distance from the lane center normalized by the center-to-edge distance (so $\text{BP} = 1$ at the boundary). The boundary penalty is

$$J_{\text{bound}}(\hat{x}_\tau^i) = [\max(0, \text{BP}(\hat{x}_\tau^i) - 1)]^2. \quad (45)$$

We set

$$J_{\text{other}}^{\text{Race}}(\hat{x}_\tau^i) = w_{\text{coll}} J_{\text{coll}}(\hat{x}_\tau^i) + w_{\text{bound}} J_{\text{bound}}(\hat{x}_\tau^i). \quad (46)$$

where $w_y, w_{\text{act}}, w_{\text{coll}}, w_{\text{bound}} \in \mathbb{R}_{\geq 0}$ are fixed scalar weights.

Table 3: PPO Hyperparameters

Parameter	Default
learning_rate	3×10^{-4}
n_steps	2048
batch_size	64
n_epochs	10
gamma	0.99
gae_lambda	0.95
clip_range	0.2
ent_coef	0.0
vf_coef	0.5

Table 4: SAC Hyperparameters

Parameter	Default
learning_rate	3×10^{-4}
buffer_size	1000000
batch_size	256
tau	0.005
gamma	0.99
train_freq	1
gradient_steps	1
ent_coef	“auto”
target_update_interval	1

Table 5: MPPI Hyperparameters

Parameter	Acrobot	LunarLander	Racing
horizon (state ref)	10	10	10
horizon (QP)	3	3	2
noise_sigma	0.5	0.5	0.5
num_samples	100	100	100
lambda	1.0	1.0	1.0
w_{RL}	50.0	50.0	1.0
w_D	50.0	400.0	300.0
w_{act}	—	20.0	—
w_y	—	10.0	—
w_{bound}	—	—	500.0
w_{coll}	—	—	300.0

Appendix D. Additional Results

We first evaluate alternative cost formulations to show that our methods are not tied to a particular cost form. We further show results using per-sample mixing, as described by (8), to contrast with the loss function weighting approach described by (10).

Alternative cost formulations Fig. 4, 5 report two evaluations, each comparing training *with* vs. *without* MPPI virtual rollout data. Fig. 4 uses the quadratic cost formulation (40), and Fig. 5 uses the RL value function as a terminal cost in MPPI (41), a common way to couple RL and MPPI. Across both formulations and all environments, we observe that adding MPPI virtual rollout data consistently yields higher performance and better sample efficiency.

Per-sample mixing We also show in Fig. 6, 7, 8 that applying the influence ratio ρ to the sampling distribution (8) yields the similar qualitative trend as shown in the main result in Fig. 3 (reproduced here for comparison), which uses loss weighting (10). Table 6 summarizes the results of Welch’s *t*-tests, reporting the *t*-statistics and *p*-values when comparing the baseline setting of ρ against other configurations for both ρ applied to sampling distribution of (8) and ρ applied to

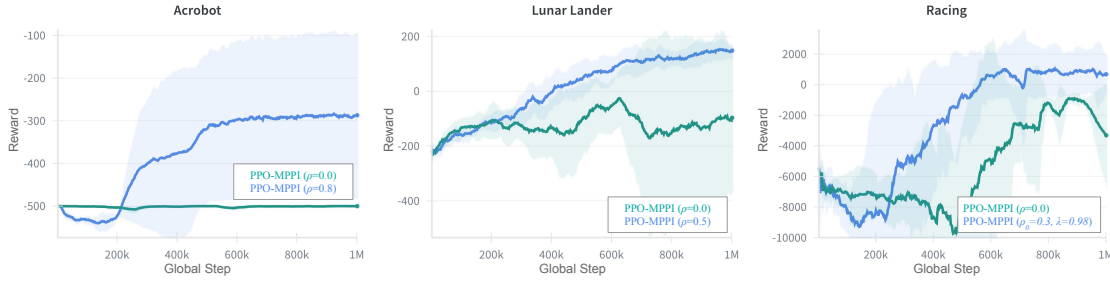


Figure 4: Episode reward under the quadratic (QP) cost formulation averaged over 5 seeds.

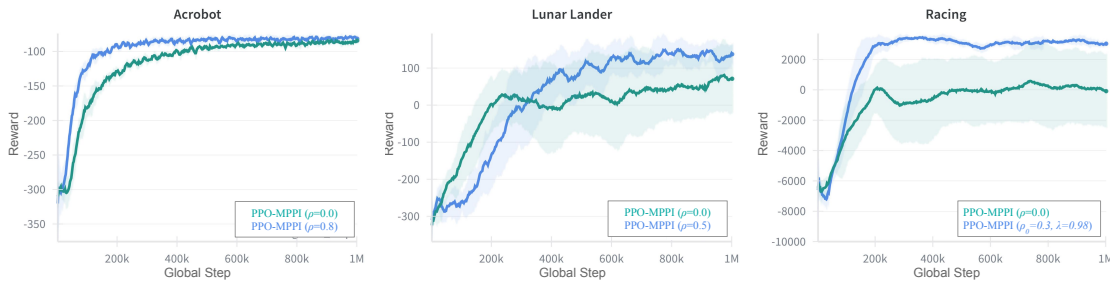


Figure 5: Episode reward under using the RL value function V as the terminal cost averaged over 5 seeds.

loss (10) in each environment. In the Acrobot environment, most of the p -values are sufficiently large, and in particular no significant differences are observed for any setting of ρ for dist. This indicates that performance does not change much with different choices of ρ , and this trend itself is shared by both ρ for dist and ρ for loss. In contrast, in the LunarLander environment, almost all settings exhibit significant differences for both ρ for dist and ρ for loss, and the signs of the t -values coincide across settings. Therefore, the ranking of which ρ values perform better is very similar between the two methods. In the Racing environment, the presence or absence of significance and the signs of the t -values agree between the two metrics for three of the four settings, so the overall trend is broadly similar; however, for the setting $\rho_0 = 0.5, \lambda = 0.95$ the signs of the t -values for ρ for sampling distribution and ρ for loss are reversed, indicating a clear discrepancy in this case. This discrepancy is likely due to differences in both the amount and the bias of the data used for policy updates between ρ for distribution and ρ for loss. Concretely, with ρ for distribution, at each step samples are stored only in one of D_{rl} or D_{mppi} , whereas with ρ for loss, at each step as many samples as the number of RL outputs are stored, with one sample saved in D_{rl} and all remaining samples saved in D_{mppi} . As a result, the total amount of data used for ρ for loss becomes larger. In a simple environment such as Acrobot, the state distribution has small variation and the data bias is limited, so this difference in data volume has only a minor impact on performance, which appears as a common trend that *changing ρ does not substantially affect performance*. In contrast, in a complex environment such as Racing, the state space is larger and the collected data is more prone to bias, so the amount of data accumulated in each buffer has a strong effect on performance. This can explain why, as observed, the trends for ρ for distribution and ρ for loss diverge for some settings. LunarLander can be seen as an environment of intermediate difficulty, where both metrics

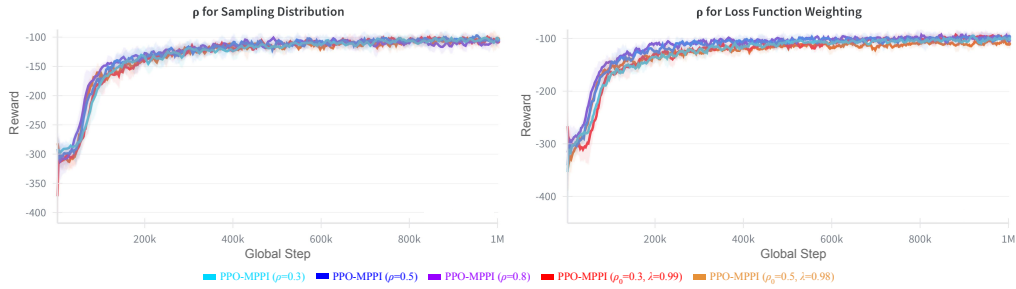


Figure 6: Comparison of applying the influence ratio ρ to the sampling distribution (8) v.s. the loss weighting (10), averaged over 5 seeds in Acrobot environment.

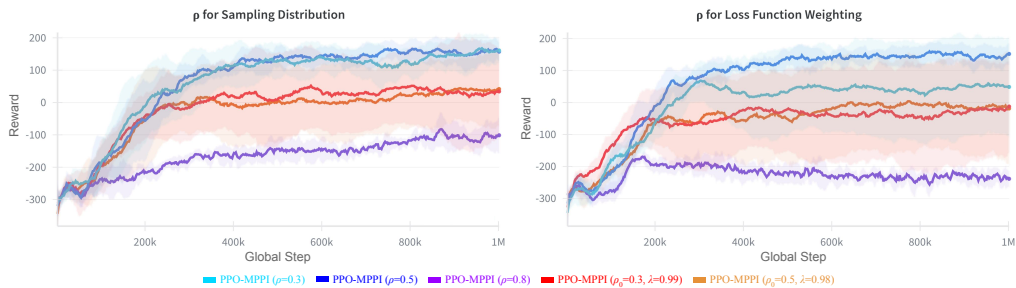


Figure 7: Comparison of applying the influence ratio ρ to the sampling distribution (8) vs. the loss weighting (10), averaged over 5 seeds in Lunar Lander environment.

obtain sufficient data without severe bias, leading to the very similar trends observed between the two methods.

Appendix E. Additional MPPI Results

In this section, we provide additional analyses of the MPPI component used in the proposed method using Lunar Lander environment. Specifically, we investigate (1) the increase in computational cost introduced by MPPI, (2) the sensitivity of performance to key MPPI hyperparameters, and (3) the performance of MPPI when used alone.

Computational cost. We first evaluate the additional computational cost introduced by adding MPPI to the PPO framework. Table 7 compares the average computation time of the pure RL baseline and PPO-MPPI for 1M training steps used in the main results. As expected, incorporating MPPI increases the computational burden because each control step requires sampling-based trajectory optimization.

This result indicates that the performance gain of PPO-MPPI comes at the cost of additional computation. However, as shown in the main results, PPO-MPPI also provides improved control performance, suggesting that this increased cost should be interpreted as a trade-off for better decision quality. Importantly, in our setting, the increase in computation remained within a controllable range and scaled predictably with MPPI hyperparameters such as the prediction horizon and the number of samples.

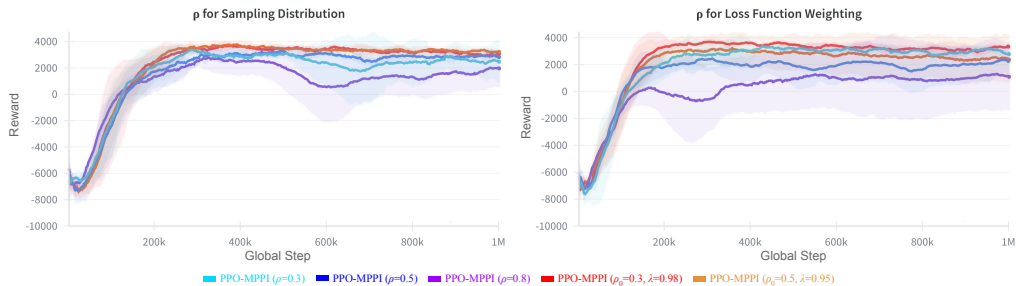


Figure 8: Comparison of applying the influence ratio ρ to the sampling distribution (8) vs. the loss weighting (10), averaged over 5 seeds in Racing environment.

Table 6: Welch t test for all tasks.

Env	Baseline	Setting	ρ for dist		ρ for loss	
			t	p	t	p
Acrobot	$\rho_0 = 0.3, \lambda = 0.99$	$\rho = 0.3$	0.643	0.522	3.31	0.133e-2
		$\rho = 0.5$	0.0318	0.975	0.481	0.632
		$\rho = 0.8$	1.56	0.123	1.11	0.271
		$\rho_0 = 0.5, \lambda = 0.98$	0.156	0.876	7.54	0.276e-10
Lunar-lander	$\rho = 0.5$	$\rho = 0.3$	0.423	0.673	4.74	0.162e-4
		$\rho = 0.8$	35.3	0.469e-53	67.8	0.511e-81
		$\rho_0 = 0.3, \lambda = 0.99$	6.83	0.737e-8	8.65	0.858e-11
		$\rho_0 = 0.5, \lambda = 0.98$	8.00	0.646e-10	7.18	0.233e-8
Racing	$\rho_0 = 0.3, \lambda = 0.98$	$\rho = 0.3$	2.44	0.179e-1	15.1	0.108e-24
		$\rho = 0.5$	2.99	0.374e-2	7.22	0.263e-8
		$\rho = 0.8$	5.59	0.790e-6	6.64	0.235e-7
		$\rho_0 = 0.5, \lambda = 0.95$	-4.47	0.299e-4	3.70	0.541e-3

Sensitivity to MPPI hyperparameters. Next, we investigate the sensitivity of the proposed framework to two key MPPI hyperparameters: the prediction horizon and the number of samples. Fig. 9 overlays the learning curves obtained under different parameter settings with those of the default configuration shown in Table 5.

Prediction horizon. We first examine the effect of the prediction horizon. When the horizon was reduced to half of the default value, performance dropped substantially. This suggests that a short horizon prevents MPPI from sufficiently anticipating future system evolution, thereby limiting its ability to generate effective auxiliary control inputs. In our task, selecting good actions requires looking beyond immediate outcomes, and an insufficient horizon directly degrades performance.

In contrast, when the horizon was doubled, performance improved slightly, but the gain was marginal. This indicates that the default horizon was already long enough to capture most of the useful future information, and that extending it further yields diminishing returns.

Number of samples. We then evaluate the effect of the number of samples. When the number of samples was reduced to half, the performance decreased slightly. On the other hand, when it

Table 7: Comparison of computation time between PPO and PPO-MPPI.

Domain	PPO	PPO-MPPI
Acrobot	33m21s	15h9m58s
Lunar Lander	48m45s	5h53m20s
Racing	5h34m31s	2d2h7m45s

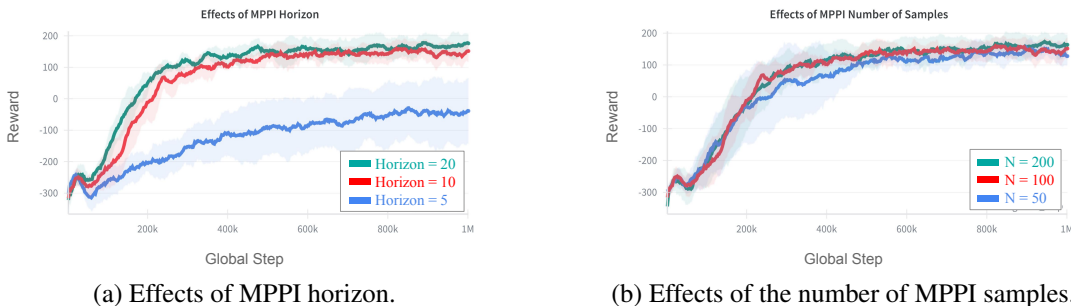


Figure 9: Sensitivity analysis of key MPPI hyperparameters.

was increased to twice the default value, the overall performance remained largely unchanged. This suggests that the default number of samples was already sufficient to provide adequate exploration in the trajectory optimization process, and that performance had already begun to saturate around this range.

Performance of pure MPPI. Finally, we evaluate the performance of MPPI when used alone. In this experiment, MPPI was executed using the same reward model as that used in PPO-MPPI as the cost function, multiplied by -1 to convert reward maximization into cost minimization. This setting allows us to test whether MPPI alone can achieve strong performance when using the same reward model as PPO-MPPI.

The results show that pure MPPI performs significantly worse than PPO-MPPI, even though it has direct access to the same reward model. This finding suggests that knowing the reward function alone is not sufficient for effective control in this task. A key reason is that the reward function is not sufficiently smooth and dense from the perspective of finite-horizon trajectory optimization. When meaningful reward differences do not appear within the prediction horizon, MPPI struggles to distinguish good trajectories from bad ones and therefore cannot reliably generate effective control actions.

This result highlights the difference in roles between RL and MPPI. RL can learn useful long-term behavioral tendencies through repeated interaction and long-horizon credit assignment. MPPI, on the other hand, is effective at finite-horizon local optimization, but its performance can be limited when the reward is sparse, discontinuous, or difficult to exploit within a short prediction window. Therefore, in settings such as ours, it is more effective to combine RL for long-term guidance with MPPI for short-term control refinement.

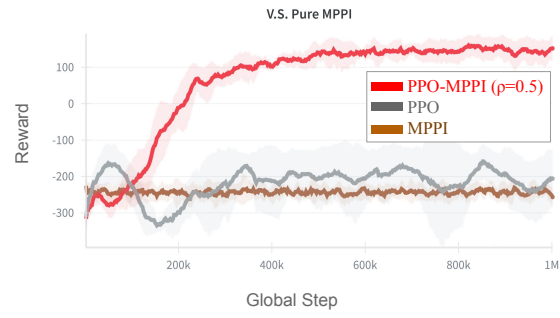


Figure 10: Performance comparison between PPO-MPPI and pure MPPI.