TD-CD-MPPI: Temporal-Difference Constraint-Discounted Model Predictive Path Integral Control

Pietro Noah Crestaz, Ludovic de Matteis, Elliot Chane-Sane, Nicolas Mansard, Andrea Del Prete

TD-CD-MPPI: Temporal-Difference Constraint-Discounted Model Predictive Path Integral Control

Pietro Noah Crestaz[1,2], Ludovic De Matteis[2], Elliot Chane-Sane[2], Nicolas Mansard[2,3], and Andrea Del Prete[1]

***Abstract*— Path Integral methods have demonstrated remarkable capabilities for solving non-linear stochastic optimal control problems through sampling-based optimization. However, their computational complexity grows linearly with the prediction horizon, limiting long-term reasoning, while constraints are merely enforced through handcrafted penalties. In this work, we propose a unified and efficient framework for enabling long-horizon reasoning and constraint enforcement within Model Predictive Path Integral (MPPI) control. First, we introduce a practical method to incorporate a terminal value function, learned offline via temporal-difference learning, to approximate the long-term cost-to-go. This allows for significantly shorter roll-outs while enabling infinite-horizon reasoning, thereby improving computational efficiency and motion performance. Second, we propose a discount modulation strategy that adjusts the return of sampled trajectories based on constraint violations. This provides a more interpretable and effective mechanism for enforcing constraints compared to traditional cost shaping. Our formulation retains the flexibility and sampling efficiency of MPPI while supporting structured integration of long-term objectives and constraint handling. We validate our approach on both simulated and real-world robotic locomotion tasks, demonstrating improved performance, constraint-awareness, and generalization under reduced computational budgets.**

I. Introduction

Legged locomotion in unstructured environments poses significant challenges, requiring real-time reasoning over complex dynamics, contact interactions, and sensor uncertainty. Whole-Body Model Predictive Control (WB-MPC) first demonstrated the potential of planning coordinated full-body motion in a receding horizon framework [1], [2]. Reinforcement Learning (RL) [3], [4] has since expanded this potential, enabling policies that jointly handle contact decisions and sensor feedback [5], [6], [7]. RL has become the dominant approach for locomotion, showing strong performance across diverse terrains and gaits. However, RL lacks several key strengths of MPC: it struggles with hard constraint enforcement, requires extensive training, and generalizes poorly outside its training domain. These limitations have driven interest in hybrid approaches that combine the structure of MPC with the adaptability of learning-based methods [8], [9], [10], [11], [12], [13], [14].

Path Integral control methods have emerged as a robust and efficient alternative to RL for non-linear stochastic optimal control, particularly in contact-rich tasks [15], [16], allowing efficient exploration, online adaptation and local trajectory improvement. These approaches rely on a Monte-Carlo approximation of the optimal solution, providing the advantages of sampling-based methods, such as easy parallelization and direct applicability to non-smooth problems. However, most Path Integral approaches suffers from two structural limitations that hinder their scalability and adaptability in modern control settings.

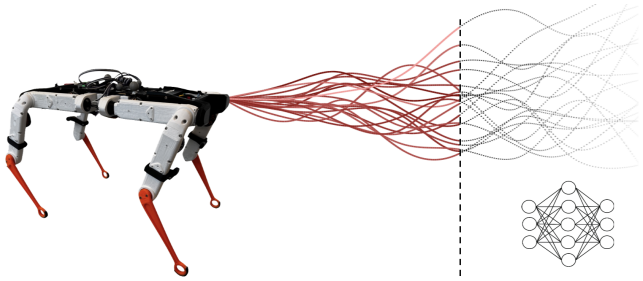

Fig. 1. The proposed TD-CD-MPPI controller enables the use of reduced control horizons by using a learned terminal value function to approximate long-term returns.

First, MPC often requires to plan over a long time horizon to approximate the infinite-horizon behavior, a property critical for many tasks such locomotion, obstacle avoidance, or goal-oriented reasoning [17], [18].

Yet, maintaining both a proper coverage and an efficient local search in sampling-based methods requires to increase the sampling, resulting in increased computational cost and increased sensitivity to model errors [19]. Several methods have been proposed to improve the sample efficiency of the Model Predictive Path Integral (MPPI) approach by adding covariance adaptation strategies [20], [19], [21], but still struggle to extend to longer time-horizons.

Second, constraint handling in sampling-based methods typically involves hand-crafted penalty functions or reward shaping, which introduces a brittle design bottleneck and deteriorates interpretability [22]. Most, if not all, robotics tasks require constraints such as joint limits, safety boundaries, or collision avoidance [23]. Using soft constraints leads to no guarantee of satisfaction, and poor scalability. Some earlier works have focused on enforcing constraint satisfaction in sampling-based methods using either projection techniques [24], [25] or by learning a distribution that promotes sampling of feasible trajectories [26], [27]. However the first approach requires projecting all constraints for all trajectories, increasing the computational burden, and tends to accumulate the trajectories near the constraint boundaries, leading to convergence toward these boundaries.

[1]Industrial Engineering Department, University of Trento, Trento, Italy.
[2]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France.
[3]Artificial and Natural Intelligence Toulouse Institute (ANITI), Toulouse.
⋆Corresponding author: pietronoah.crestaz@unitn.it

The second approach, while effective, is usually complex to implement, computationally expensive, and scales badly [28]

In this work, we address both challenges through a value-augmented formulation of MPPI. Building upon the Dial-MPC framework [19], which improves sampling efficiency via a diffusion-inspired mechanism, we extend the approach by introducing a learned *value function* (VF), trained offline via temporal-difference learning (TD(0)), to approximate the infinite-horizon cost-to-go at the end of the planning horizon. This value estimates the long-term return and allows the controller to maintain high-quality performance even with significantly shorter horizons. Unlike heuristic terminal costs, our VF is learned directly from the control policy's own behavior, leading to a consistent and data-driven extension of MPPI's cost structure. Additionally, the JAX-based implementation by [19] enables seamless integration into high-performance GPU-based control pipelines.

To further improve modularity and interpretability in behavior shaping, we propose a discount factor modulation scheme that allows trajectory-wise adjustments of the effective horizon based on constraint satisfaction, similar to the approach proposed in [29], [7]. Specifically, rather than enforcing soft constraints using penalties, we modulate the trajectory discount factor to decay more rapidly for trajectories that violate the constraints. This formulation does not enforce hard constraints in the classical sense, but offers a lightweight and interpretable way to prioritize feasible trajectories without directly modifying the cost and without deteriorating exploration. This can be interpreted as a soft mechanism to "truncate" undesirable behaviors, inspired by the notion of terminal states in constrained reinforcement learning.

Our contributions can be summarized as follows:

- We present a novel formulation of MPPI that integrates a terminal value function learned via TD(0), enabling short-horizon control with improved performance and long-term reasoning.
- We propose a principled method for discount factor modulation as a function of constraint violations, providing a modular alternative to cost shaping for constraint-aware planning.
- We demonstrate that our method enables constraint-sensitive and sample-efficient control in challenging locomotion tasks, including obstacle avoidance and contact-rich behaviors.
- We validate our approach in real-world conditions on the Solo12 quadruped robot, showing that the method transfers effectively from simulation to hardware for locomotion.

II. Background

A. Optimal control problem

In this work, we consider finite-horizon optimal control problems (OCP), which can be expressed as:

$$\begin{aligned} \max_{u_{t:t+H}} \quad & \sum_{h=0}^{H} r_h(x_{t+h}, u_{t+h}) + r_{H+1}(x_{t+H+1}) \\ \text{s.t.} \quad & x_{t+h+1} = f_{t+h}(x_{t+h}, u_{t+h}), \quad \forall h = 0, \dots, H \\ & x_{t:t+H+1} \in \mathcal{X}, \quad u_{t:t+H} \in \mathcal{U} \end{aligned} \tag{1}$$

where $r_h : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ denotes the stage reward, $r_{H+1} : \mathcal{X} \to \mathbb{R}$ the terminal reward, $f_h : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ the known system dynamics, and $\mathcal{X}$, $\mathcal{U}$ the admissible state and control spaces, respectively. The control sequence $u_{t:t+H}$ is optimized over a fixed horizon H, starting from the current state x_t. Note that this problem can be reformulated equivalently using a maximization of rewards instead of a minimization of costs.

B. Model Predictive Path Integral (MPPI)

To solve this OCP, we adopt the MPPI control framework. The main idea behind Path Integral approaches is to rewrite the Hamilton-Jacobi-Bellman equation as a linear partial differential equation and to use the Feynman-Kac lemma to express the solution to this equation as an expectation over randomized trajectories. In practice, this is implemented using a Monte-Carlo approach. The MPPI algorithm approximates the optimal control sequence by iteratively refining a nominal control trajectory $U = u_{t:t+H}$.

First, N perturbations $\{w^{(i)}\}_{i=1}^{N}$ are drawn from a zero-mean Gaussian distribution:

$$w^{(i)} \sim \mathcal{N}(0, \Sigma_{t:t+H}), \quad i = 1, \dots, N, \tag{2}$$

where $\Sigma_{t:t+H}$ is the covariance matrix governing the perturbations [19]. For each perturbed control sequence $U^{(i)} = U + w^{(i)}$, the cost $J(U^{(i)})$ is computed by rolling out the system dynamics:

$$R(U^{(i)}) = \sum_{h=0}^{H} r_h(x_{t+h}^{(i)}, u_{t+h}^{(i)}) + r_{H+1}(x_{t+H+1}^{(i)}). \tag{3}$$

Next, the nominal control sequence is updated using a weighted average of the perturbations, where the weight for each perturbation is determined by its corresponding cost:

$$U \leftarrow U + \frac{\sum_{i=1}^{N} \exp\left(\frac{R(U^{(i)})}{\lambda}\right) w^{(i)}}{\sum_{j=1}^{N} \exp\left(\frac{R(U^{(j)})}{\lambda}\right)}, \tag{4}$$

where $\lambda > 0$ is a temperature parameter that controls the selectivity of the update.

III. Method

To enable both value function (VF) learning and constraints handling, we reformulate the objective function of OCP (1) using a discounted cost (or reward). Specifically,

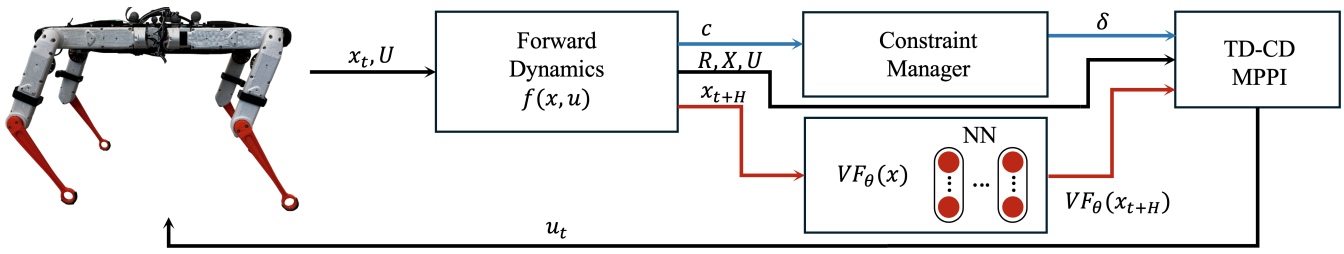

Fig. 2. TD-CD-MMPI controller structure, based on MPPI sample based optimization using the forward dynamics computed by a contact simulator. A terminal value function (VF), learned offline via temporal-difference learning, enables long-term reasoning with short roll-outs. Constraint manager modulate trajectory discounts based on violation, enforcing feasibility without penalty shaping.

we introduce a discount factor $\gamma_0 \in (0, 1]$ and express the reward functional as:

$$\sum_{h=0}^{H} \gamma_0^h r(x_{t+h}, u_{t+h}) + \gamma_0^{H+1} r_t(x_{t+H+1}). \tag{5}$$

This discounted formulation allows for the natural incorporation of stochastic termination criteria, as will be discussed in the following section.

To improve the performance of MPPI we adopt the diffusion-inspired annealing method (DIAL-MPC) [19]. DIAL-MPC refines the exploration-exploitation trade-off by gradually denoising the control sequence distributions. This annealing approach is realized through a sequence of diffusion stages, where each stage introduces increasingly smaller noise to the perturbations.

A. Constraint Handling via Stochastic Terminations

To account for constraints during control without severely limiting exploration, we adopt the Constraints as Terminations (CaT) approach introduced in [29]. This method interprets constraint violations as probabilistic terminations of the trajectory, allowing us to smoothly penalize violations without harsh truncation. Given a set of constraints:

$$c_i(x, u) \leq 0 \qquad \forall i \in \mathcal{I}, \tag{6}$$

we compute a stochastic termination signal $\delta_h^{(j)} \in [0, 1]$ at each step of a sampled trajectory j, based on the normalized constraint violations:

$$\delta_h^{(j)} = \max_{i \in \mathcal{I}} \left(p_i^{\max} \cdot \text{clip} \left(\frac{[c_h^{(j)}]_i}{[c_i^{\max}]}, 0, 1 \right) \right). \tag{7}$$

The hyperparameter $p_i^{\max} \in [0, 1]$ controls the sensitivity to violations, while $c_i^{\max}$ is an exponential moving average of the maximum violation observed for constraint i over all trajectories, which we update at each iteration as:

$$c_i^{\max} \leftarrow \tau_c \cdot \hat{c}_i^{\max} + (1 - \tau_c) \cdot c_i^{\max}, \tag{8}$$

where $\hat{c}_i^{\max} = \max_{j,h} [c_h^{(j)}]_i$ is the maximum violation observed so far for constraint i, and $\tau_c \in [0, 1]$ is a hyperparameter.

These time-varying discounts scale down future rewards and the terminal value contribution, producing a modified reward for trajectory j:

$$R^{(j)} = \sum_{h=0}^{H} \left(\prod_{k=0}^{h} \gamma_0 (1 - \delta_k^{(j)}) \right) r_h^{(j)} \tag{9}$$

This soft termination mechanism allows learning to recover from constraint violations rather than completely discarding affected trajectories.

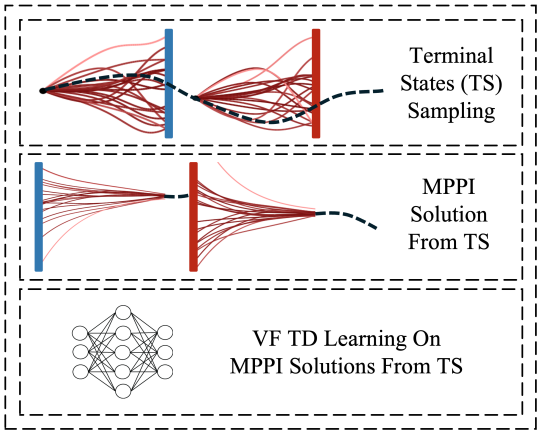

Fig. 3. TD-Learning dataset creation and learning pipeline. Terminal states are sampled from rollout trajectories, and later used as starting conditions for computing an optimal trajectory using the Constraint-Discounted formulation of MPPI. The created dataset constitutes the base for the temporal difference learning.

B. Value Function learning through Temporal-Difference

To further improve the controller, we integrate Temporal-Difference (TD) Learning within the MPPI framework by learning a value function that approximates the long-term cost-to-go from the terminal state of the trajectories generated during the MPPI rollout process.

First, we sample several control trajectories, following the standard MPPI flow, and rollout these trajectories to get a set of terminal states $x_{t+H+1}^{(i)}$ corresponding to each trajectory i. These states do not correspond to any optimal behavior but constitute a representative set of the reachable states for the given time horizon.

Then, we generate a set of optimal trajectories using DIAL-MPC, starting from the collected terminal states $x_{t+H+1}^{(i)}$, and record all transitions (x_t, u_t, r_t, x_{t+1}) along

this trajectory. The reward r_t represents the immediate reward obtained at each time step of the rollout. This procedure gives us a dataset consisting of state-action pairs and rewards for optimal trajectories starting from a representative set of states.

We then apply Temporal-Difference Learning (TD(0)) [30] on this dataset to learn an approximate value function $V(x_t)$ that predicts the future reward starting from state x_t. The standard TD update rule is

$$V(x_t) \leftarrow V(x_t) + \alpha \left(r_t + \gamma_0 V(x_{t+1}) - V(x_t) \right), \quad (10)$$

where $\alpha \in [0, 1]$ is the learning rate.

Because of the constraint handling procedure presented above, we need to consider the updated version of the reward function, reported in (9). This equation can be written in a recursive manner to be included in the TD learning procedure:

$$\begin{aligned} V(x_0) &= \sum_{t=0}^{\infty} \left(\prod_{t'=0}^{t} \gamma_0 (1 - \delta_{t'}) \right) r_t && \text{(11a)} \\ &= \gamma_0 (1 - \delta_0) r_0 + \sum_{t=1}^{\infty} \left(\prod_{t'=0}^{t} \gamma_0 (1 - \delta_{t'}) \right) r_t && \text{(11b)} \\ &= \gamma_0 (1 - \delta_0) r_0 \\ &\quad + \gamma_0 (1 - \delta_0) \sum_{t=1}^{\infty} \left(\prod_{t'=1}^{t} \gamma_0 (1 - \delta_{t'}) \right) r_t && \text{(11c)} \\ &= \gamma_0 (1 - \delta_0) r_0 + \gamma_0 (1 - \delta_0) V(x_1) && \text{(11d)} \\ &= \gamma_0 (1 - \delta_0) \left(r_0 + V(x_1) \right) && \text{(11e)} \end{aligned}$$

Although in standard formulations the discount factor typically applies only to future rewards, here it is applied from the current timestep to account for constraint-based discounting.

Once the value function is learned, we incorporate it into the MPPI framework as a terminal reward:

$$\begin{aligned} R^{(j)} = &\sum_{h=0}^{H} \left(\prod_{k=0}^{h} \gamma_0 (1 - \delta_k^{(j)}) \right) r_h^{(j)} \\ &+ \left(\prod_{k=0}^{H+1} \gamma_0 (1 - \delta_k^{(j)}) \right) V(x_{t+H+1}^{(j)}). \end{aligned} \quad (12)$$

This formulation assumes all reward terms $r_h^{(j)}$ to be non-negative to ensure consistency with the discounted formulation.

Algorithm 1 outlines the Constraint-Discounted TD-Learning procedure, while Algorithm 2 presents the full TD-CD-MPPI algorithm.

C. Benefits of the Value Function Approximation

The main advantage of integrating the value function approximation into the MPPI framework is that it allows the controller to consider longer-term costs, which improves the overall performance. By approximating the value function, we can reduce the control horizon H for the MPPI algorithm while maintaining similar control performance.

Algorithm 1: TD Learning for Value Function Approximation with Constraint-Discounted return

Input: Dataset $\mathcal{D} = \{(x, x', r, \delta)\}$, parameters θ, target θ', discount γ_0, soft update $\tau \in [0, 1]$, learning rate α, batch size B, number of epochs E

for *epoch* $= 1$ **to** E **do**
  foreach *minibatch* $(x_b, x_b', r_b, \delta_b)$ **do**
    $V \leftarrow V_\theta(x_b), \quad V' \leftarrow V_{\theta'}(x_b')$
    $y \leftarrow (1 - \delta_b) \cdot (r_b + \gamma V')$
    $\mathcal{L}(\theta) \leftarrow \frac{1}{B} \sum (V - y)^2$
    $g \leftarrow \nabla_\theta \mathcal{L}(\theta)$
    $\theta \leftarrow \theta - \alpha g$
    $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$

return θ

Algorithm 2: Temporal Difference Constraint Discounted MPPI (TD-CD-MPPI)

Input: Current state x_t, control sequence $U = u_{t:t+H}$, horizon H, samples N, temperature λ, value function $V_\theta(\cdot)$

for $j = 1$ **to** N **do** `// Sample N rollouts`
  for $h = 0$ **to** H **do**
    Sample $w_h^{(j)} \sim \mathcal{N}(0, \Sigma_{t+h})$
    $u_{t+h}^{(j)} \leftarrow u_{t+h} + w_h^{(j)}$
    $x_{t+h+1}^{(j)} \leftarrow f(x_{t+h}^{(j)}, u_{t+h}^{(j)})$
    $r_h^{(j)} \leftarrow \ell(x_{t+h}^{(j)}, u_{t+h}^{(j)}), \quad c_h^{(j)} \leftarrow C(x_{t+h}^{(j)}, u_{t+h}^{(j)})$

foreach $i \in I$ **do**
  $\hat{c}_i^{\max} \leftarrow \max_{j,h} c_h^{(j)}[i]$
  $c_i^{\max} \leftarrow \alpha \cdot \hat{c}_i^{\max} + (1 - \alpha) \cdot c_i^{\max}$

for $j = 1$ **to** N **do** `// Discounted returns`
  for $h = 0$ **to** H **do**
    $\delta_h^{(j)} \leftarrow \max_i \left(p_i^{\max} \cdot \text{clip} \left(\frac{c_h^{(j)}[i]}{c_i^{\max}}, 0, 1 \right) \right)$

$$\begin{aligned} R^{(j)} = &\sum_{h=0}^{H} \left(\prod_{k=0}^{h} \gamma_0 (1 - \delta_k^{(j)}) \right) r_h^{(j)} \\ &+ \left(\prod_{k=0}^{H+1} \gamma_0 (1 - \delta_k^{(j)}) \right) V_\theta(x_{t+H+1}^{(j)}) \end{aligned}$$

Update control sequence:

$$U \leftarrow U + \frac{\sum_j \exp\left(R^{(j)}/\lambda \right) W^{(j)}}{\sum_j \exp\left(R^{(j)}/\lambda \right)}$$

return U

This reduction in horizon size leads to significantly lower computational cost, enabling the controller to use an higher number of samples or to be deployed on hardware with limited resources, such as smaller GPUs. Additionally, the value function helps guide the decision-making process by estimating the future rewards, even if the current trajectory is not optimal, and thus encourages more informed and robust control actions.

Moreover, the inclusion of the learned value function enables the controller to benefit from long-term reasoning computed offline, while still retaining an online MPC

TABLE I
TD LEARNING HYPERPARAMETERS

Parameter	Value
Hidden layers (neurons)	[512, 256, 128]
Activation function	ELU
Discount factor γ_0	0.90
Batch size	512
Learning rate	3×10^{-4}
Target update rate τ	1×10^{-3}

component that can reactively compensate for external disturbances or model-environment mismatches. This hybrid design leverages the strengths of both offline learning and online optimization.

IV. EXPERIMENTS

In this section, we evaluate the proposed TD-CD-MPPI controller both in simulation and on the real Solo12 quadruped robot. We begin by presenting the implementation details, including the simulation environment, neural network setup and hardware configuration used for the real Solo12 quadruped robot experiments.

A. Software Stack and Experimental Setup

Simulation experiments are carried out using MuJoCo [31] via the MJX physics engine, leveraging JAX [32] for just-in-time compilation and efficient parallelization of trajectory rollouts. This infrastructure enables scalable and fast sampling, which is critical for both real-time control and value function updates. The same JAX-based simulation stack is also used during real-robot experiments.

The neural network used to approximate the value function is implemented and trained using Flax [33], a high-performance neural network library built on top of JAX, which enables efficient batching and gradient computations. Our implementation of TD-CD-MPPI builds upon the DIAL-MPC codebase [19], which provides a modular and extensible framework for real-time sampling-based controller implementation.

The function approximator used for the value function is a multilayer perceptron (MLP) consisting of three fully connected hidden layers. Each hidden layer is followed by an Exponential Linear Unit (ELU) activation and Layer Normalization to improve training stability and convergence. The network outputs a single scalar value representing the estimated value function. The network is trained by minimizing the mean squared error (MSE) loss between the predicted value and the target computed via temporal difference learning. Table I summarizes the key hyperparameters used during training.

On the real system, the controller runs offboard on an external computer equipped with a 13th Gen Intel Core i9-13900K and an NVIDIA RTX 4090 GPU. Control inputs are computed at 50 Hz and transmitted to the robot via a low-latency Ethernet connection.

The robot's state is estimated using a Qualisys motion capture system operating at 300 Hz. These measurements are fused with onboard IMU data through a filtering process to obtain accurate and low-delay state feedback for control.

B. Results and Analysis

TD-CD-MPPI is evaluated across several dimensions, focusing on its efficiency, generalization capabilities and transfer to real hardware under diverse and challenging scenarios. Each result is individually illustrated in the companion video.

a) Effectiveness: On flat terrain, TD-CD-MPPI achieves similar locomotion quality compared to the baseline DIAL-MPC, while requiring significantly less computational resources. Fig. 5 shows the average reward per time step over 8-second flat terrain episodes as a function of the MPC horizon and number of samples. DIAL-MPC fails to maintain stable behavior with short horizons (it typically needs $H \geq 10$), while the VF-augmented controller demonstrates a more stable performance degradation, confirming the effectiveness of the terminal value function. While the rollout horizon is critical for MPC deployment, our contribution allows long-time reasoning for practical run-time computations.

In Fig. 5 bottom, we compare DIAL-MPC (H=16) with our VF-augmented controller (H=8), showing that TD-CD-MPPI is able to consistently match the average reward of DIAL-MPC even under reduced computational resources.

This improvement leads to a direct reduction in GPU memory usage and computational cost. As a result, TD-CD-MPPI achieves comparable control performance while lowering the GPU memory footprint by up to 30%. Our nominal setting is $H = 8$ and 512 rollouts, while DIAL-MPC uses $H = 16$ and 1024 to 2048 rollouts, i.e. we use 5 to 10 times less simulation steps than the baseline.

b) Generalization: A key strength of MPC is the capability to generalize at runtime by relying on on-edge optimization. We validated TD-CD-MPPI beyond flat ground walking and show that the value function does not need specialization to non-flat terrains. We demonstrated successful locomotion on staircases, characterized by abrupt discontinuities in elevation and contact dynamics, as well as on inclined surfaces that introduce sustained slope-induced disturbances. First, we assess the performance of TD-CD-MPPI against the baseline DIAL-MPC. Both controllers are informed of the terrain shape: the MPPI rollouts are performed on the correct terrain model, with stairs and slope. Yet, we challenged TD-CD-MPPI by using the VF trained on flat terrain (no retraining). An overview of the behaviour on stairs is given in Fig. 4, also shown in the video. TD-CD-MPPI can easily traverse stairs and slopes, without requiring any contact planning nor retraining of the VF. Surprisingly, TD-CD-MPPI outperforms DIAL-MPC in both environments as reported in Table II. We hypothesize that it is due to the combination of properly handling the constraints and enabling long-horizon reasoning with a reduced number of rollouts.

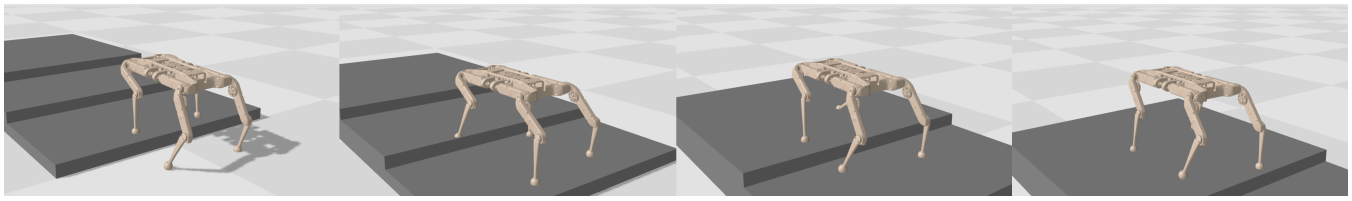

Fig. 4. Sequence of snapshots showing the quadruped climbing a three-step staircase using the proposed TD-CD-MPPI controller. The controller successfully handles terrain configurations not seen during training of the value function, leveraging real-time optimization to adapt online, while strictly enforcing dynamic and style-related constraints throughout the motion.

TABLE II

PERFORMANCE COMPARISON OF TD-CD-MPPI AND PENALTY-BASED CONSTRAINED DIAL-MPC-REWARD WITH SHORT (SH-10 STEPS) AND LONG (LH-20 STEPS) HORIZONS. SUCCESS RATE AND NORMALIZED CONSTRAINT VIOLATION ARE AVERAGED OVER 20 EPISODES PER CONDITION.

Method	**Incline SH**		**Incline LH**		**Stairs SH**		**Stairs LH**	
	Succ. rate ↑	**Cstr. ↓**	**Succ. rate ↑**	**Cstr. ↓**	**Succ. rate ↑**	**Cstr. ↓**	**Succ. rate ↑**	**Cstr. ↓**
DIAL-MPC-Reward	30%	33.9%	**100%**	44.7%	5%	61.4%	50%	69.7%
TD-CD-MPPI	**100%**	**0.1%**	**100%**	**1.5%**	**85%**	**1.8%**	**85%**	**6.3%**

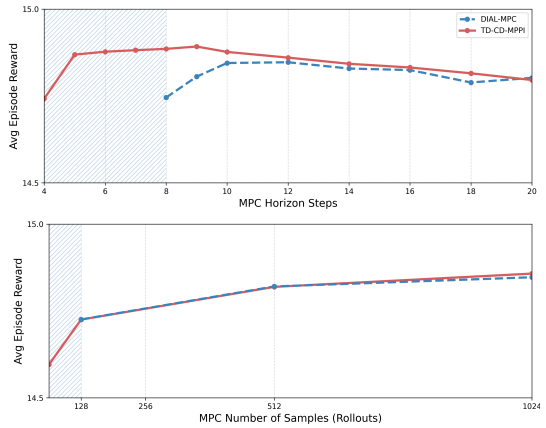

Fig. 5. Comparison of the average episode reward obtained by the DIAL-MPC controller and our proposed TD-CD-MPPI variant as a function of the MPC horizon length (Top) and number of samples (Bottom). The hatched blue region highlights the range where DIAL-MPC fails to ensure convergence: the controller is unable to generate a stable walking behavior, and the robot eventually falls, causing early episode termination. In contrast, TD-CD-MPPI maintains reliable walking behavior even at significantly reduce planning horizon and number of samples.

We also emphasized the interest of MPC against RL by comparing TD-CD-MPPI with the same reward optimized by PPO [29]. Obviously, the trained policy is not expected to generalize to novel situations. In practice, performance exhibits a sharp threshold: PPO succeeds in all episodes with slopes below $8°$, while it completely fails with higher slopes. TD-CD-MPPI maintains a robust performance up to $20°$, beyond which success drops due to insufficient friction in our simulator. An illustration is given in Fig. 6.

c) Constraint satisfaction: We asses the capability of TD-CD-MPPI to enforce constraints by comparing it to two other MPC algorithms:

- **DIAL-MPC** without explicit constraints: standard MPPI without constraint enforcement [19].
- **DIAL-MPC-Reward**: an ad-hoc variation of DIAL-MPC where constraints are incorporated via penalty terms in the reward without value approximation.

TABLE III

CONSTRAINT ENCODING HAA/HEIGHT.

DIAL-MPC-Reward	**TD-CD-MPPI**
$r = \exp\left(-\frac{1}{\alpha}(x_z - h_{\max})^2\right)$	$c = \frac{x_z - h_{\max}}{h_{\max}} \leq 0$
$r = \exp\left(-\frac{1}{\alpha}\left(\max(\lvert q_{\text{HAA}}\rvert - q_{\text{lim}}, 0)\right)^2\right)$	$c = \frac{\lvert q_{\text{HAA}}\rvert - q_{\text{lim}}}{q_{\text{lim}}} \leq 0$

TABLE IV

AVERAGE NORMALIZED CONSTRAINT VIOLATION DURING FLAT TERRAIN LOCOMOTION

Method	**HAA Joint**	**Body Height**
DIAL-MPC	55.5%	22.2%
DIAL-MPC-Reward	2.9%	13.8%
TD-CD-MPPI	**0.01%**	**0.05%**

TD-CD-MPPI enables clear and flexible constraint specification without extensive reward engineering, while reward shaping requires careful manual tuning and may lead to less interpretable formulations.

Reward and constraint formulations are detailed in Table III. Fig. 8 illustrates style and collision-avoidance constraint enforcement during flat terrain locomotion: joint angle limits on the Hip-roll joint (top) and an upper bound on body height simulating collision avoidance (bottom). As shown, the reward-shaped controller occasionally violates constraints, whereas TD-CD-MPPI consistently satisfies all constraints throughout the task. This observation is quantitatively supported by the average normalized constraint violation percentages reported in Table IV, where TD-CD-MPPI achieves violations below 0.1% for both constraints, significantly outperforming the reward-shaped MPPI and baseline controllers.

For a broader comparison, Table II reports a quantita-

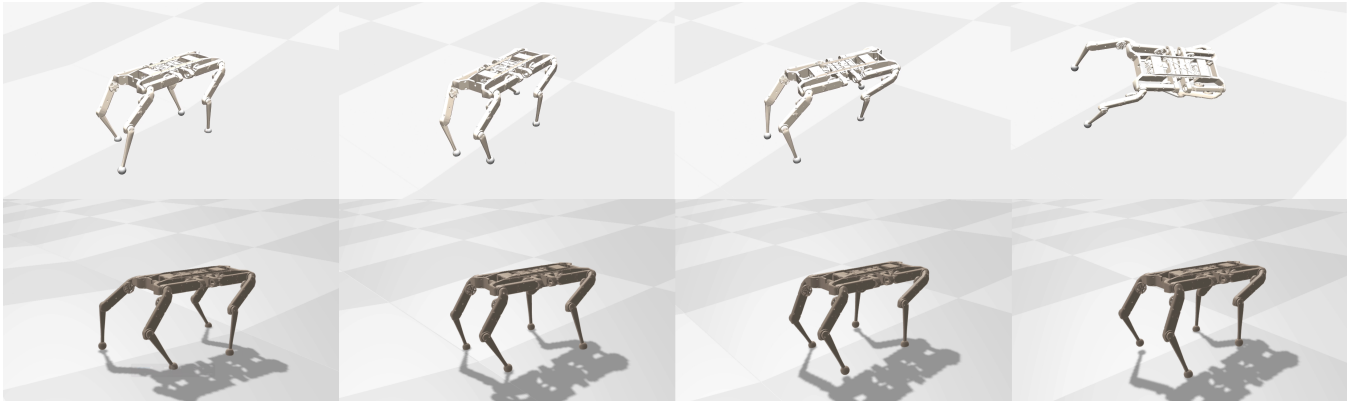

Fig. 6. Comparison on inclined terrain outside the training distribution between PPO (**top**) and TD-CD-MPPI (**bottom**), both using a lower-bound constraint on foot air time as gait shaping constraint. The policy trained with PPO becomes unstable under the out-of-distribution conditions, while TD-CD-MPPI keeps consistent locomotion behavior.

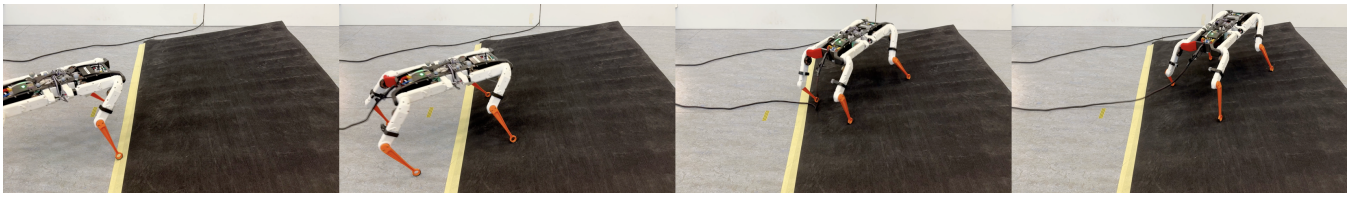

Fig. 7. Real-world evaluations on the Solo-12 quadruped. Deployment on inclined terrain unseen during training, where TD-CD-MPC adapts online while enforcing style and safety constraints.

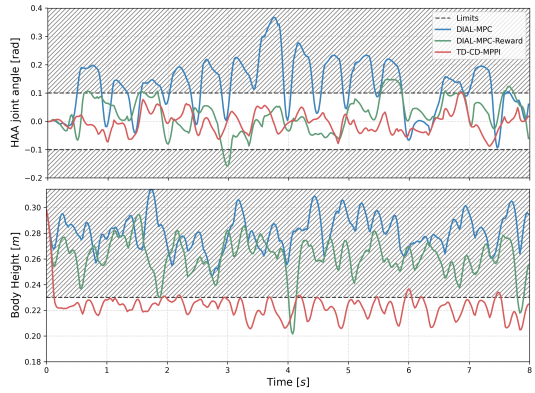

Fig. 8. Comparison of constraint violations during flat terrain locomotion. **Top:** Hip-roll joint angle limits. **Bottom:** Body height upper bound. We compare MPPI without constraints, MPPI with handcrafted reward penalties, and TD-CD-MPPI with encoded constraints.

tive comparison of the reward-shaped DIAL-MPC and TD-CD-MPPI controllers across multiple terrains and varying time horizons. Considered constraints include joint position, velocity, acceleration, and locomotion style. The violation metric quantifies the average normalized constraint violation per episode, while the success rate indicates the percentage of episodes completed without early termination. Notably, tuning the penalty weights for the handcrafted penalty MPPI proved challenging, and this configuration represents the best trade-off found without significantly sacrificing controller efficiency.

d) Transfer to the real robot: To validate the sim-to-real capabilities, TD-CD-MPPI was deployed on a real quadruped robot platform. The experiments are conducted on the Solo12 robot, a lightweight and torque-controlled quadruped developed by the Open Dynamic Robot Initiative [34]. The robot is tested under multiple scenarios, including flat-ground walking, ramp traversal, and staircase climbing. To assess the robustness of the approach under modeling inaccuracies, we tested TD-CD-MPPI with a modified model that includes an additional payload placed on the robot's torso, consisting of two metal blocks for a total added mass of approximately $0.5\,\mathrm{kg}$ (about 20% of the robot's nominal mass). This alters the inertial properties of the robot and introduces a mismatch between training conditions and deployment, described by also altering TD-CD-MPPI underlying dynamic model. As shown in Fig. 7, the controller generalizes and transfers to the real HW, maintaining the desired gait pattern and enforcing the physical constraints defined in the control formulation. Notably, this is achieved without the need for additional fine-tuning or domain adaptation.

V. Conclusions

We presented Temporal-Difference Constraint-Discounted MPPI (TD-CD-MPPI), a novel formulation of Model Predictive Path Integral control that enables real-scale deployment on legged robots with both long-term reasoning and constraint enforcement. By introducing an offline-learned value function via temporal-difference learning, we approximate the infinite-horizon cost-to-go, significantly reducing the roll-out length required during planning. This facilitates more

efficient and tractable MPC under real-time constraints. We also introduced a constraint-handling mechanism in the MPC framework, reformulating constraint violations as trajectory terminations and embedding them into the discount factor. This a pragmatic and efficient way to introduce constraint enforcement into MPPI solvers without the need of cost shaping. Crucially, combining value function approximation with discount-based constraint handling enhances the numerical stability of the optimization. The increased stability reduces the number of required roll-outs per control cycle, lowering the memory footprint of the solver and enabling deployment on smaller-scale GPUs. Beyond efficiency, our results demonstrate strong generalization to out-of-distribution scenarios not seen during training. Surprisingly, the learned value function improves generalization even compared to pure MPPI, likely due to its ability to encode long-horizon reasoning within short-horizon planning.

These findings highlight a promising direction for combining the structure and safety of model-based control with the adaptability and foresight of value-driven learning. Future work will explore extensions to more complex loco-manipulation tasks and deeper integration with reinforcement learning to further enhance sample efficiency and control scalability.

REFERENCES

[1] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[2] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2536–2542, IEEE, 2020.

[3] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *International conference on machine learning*, pp. 2829–2838, PMLR, 2016.

[4] D. Bertsekas, *Reinforcement learning and optimal control*, vol. 1. Athena Scientific, 2019.

[5] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Conference on robot learning*, pp. 403–415, PMLR, 2023.

[6] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *Science Robotics*, vol. 9, no. 88, p. eadi7566, 2024.

[7] E. Chane-Sane, J. Amigo, T. Flayols, L. Righetti, and N. Mansard, "Soloparkour: Constrained reinforcement learning for visual locomotion from privileged experience," in *Conference on Robot Learning (CoRL)*, 2024.

[8] N. Hansen, X. Wang, and H. Su, "Temporal difference learning for model predictive control," in *ICML*, 2022.

[9] N. Hansen, H. Su, and X. Wang, "Td-mpc2: Scalable, robust world models for continuous control," 2024.

[10] R. Reiter, J. Hoffmann, D. Reinhardt, F. Messerer, K. Baumgärtner, S. Sawant, J. Boedecker, M. Diehl, and S. Gros, "Synthesis of model predictive control and reinforcement learning: Survey and classification," 2025.

[11] A. Jordana, S. Kleff, A. Haffemayer, J. Ortiz-Haro, J. Carpentier, N. Mansard, and L. Righetti, "Infinite-horizon value function approximation for model predictive control," *arXiv preprint arXiv:2502.06760*, 2025.

[12] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan online, learn offline: Efficient learning and exploration via model-based control," 2019.

[13] G. Grandesso, E. Alboni, G. P. R. Papini, P. M. Wensing, and A. Del Prete, "Cacto: Continuous actor-critic with trajectory optimization—towards global optimality," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3318–3325, 2023.

[14] E. Alboni, G. Grandesso, G. P. R. Papini, J. Carpentier, and A. Del Prete, "Cacto-sl: Using sobolev learning to improve continuous actor-critic with trajectory optimization," in *6th Annual Learning for Dynamics & Control Conference*, pp. 1452–1463, PMLR, 2024.

[15] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic MPC for model-based reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1714–1721, May 2017. Read_Status: Read Read_Status_Date: 2025-03-31T11:37:16.267Z.

[16] G. Williams, A. Aldrich, and E. Theodorou, "Model Predictive Path Integral Control using Covariance Variable Importance Sampling," Sept. 2015. Read_Status: Read Read_Status_Date: 2025-03-31T11:37:11.373Z.

[17] B. Hu and A. Linnemann, "Toward infinite-horizon optimality in nonlinear model predictive control," *IEEE Transactions on Automatic Control*, vol. 47, no. 4, pp. 679–682, 2002.

[18] L. Grune and A. Rantzer, "On the infinite horizon performance of receding horizon controllers," *IEEE Transactions on Automatic Control*, vol. 53, no. 9, pp. 2100–2111, 2008.

[19] H. Xue, C. Pan, Z. Yi, G. Qu, and G. Shi, "Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing," 2024.

[20] Z. Yi, C. Pan, G. He, G. Qu, and G. Shi, "CoVO-MPC: Theoretical Analysis of Sampling-based MPC and Optimal Covariance Design," Jan. 2024. arXiv:2401.07369 [cs] Read_Status: Read Read_Status_Date: 2025-03-18T08:24:31.513Z.

[21] J. Yin, Z. Zhang, E. Theodorou, and P. Tsiotras, "Trajectory Distribution Control for Model Predictive Path Integral Control using Covariance Steering," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 1478–1484, 2022. arXiv:2109.12147 [cs] Read_Status: Read Read_Status_Date: 2025-04-09T07:47:28.127Z.

[22] E. Dantec, M. Naveau, P. Fernbach, N. Villa, G. Saurel, O. Stasse, M. Taix, and N. Mansard, "Whole-body model predictive control for biped locomotion on a torque-controlled humanoid robot," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, pp. 638–644, IEEE, 2022.

[23] A. Haffemayer, A. Jordana, L. de Matteis, K. Wojciechowski, F. Lamiraux, and N. Mansard, "Collision avoidance in model predictive control using velocity damper," *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[24] J. Carius, R. Ranftl, F. Farshidian, and M. Hutter, "Constrained stochastic optimal control with learned importance sampling: A path integral approach," *The International Journal of Robotics Research*, vol. 41, pp. 189–209, Feb. 2022. Publisher: SAGE Publications Ltd STM Read_Status: Read Read_Status_Date: 2025-04-22T11:18:54.909Z.

[25] J. Kim, I. Ko, and F. C. P. and, "Randomized path planning for tasks requiring the release and regrasp of objects," *Advanced Robotics*, vol. 30, no. 4, pp. 270–283, 2016.

[26] J. Sacks and B. Boots, "Learning sampling distributions for model predictive control," in *Conference on Robot Learning*, pp. 1733–1742, PMLR, 2023.

[27] T. Power and D. Berenson, "Variational inference mpc using normalizing flows and out-of-distribution projection," 2022.

[28] A. Razmjoo, T. Xue, S. Shetty, and S. Calinon, "Sampling-based constrained motion planning with products of experts," 2024.

[29] E. Chane-Sane, P.-A. Leziart, T. Flayols, O. Stasse, P. Souères, and N. Mansard, "Cat: Constraints as terminations for legged locomotion reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[30] R. S. Sutton, A. G. Barto, *et al.*, *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge, 1998.

[31] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, 2012.

[32] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[33] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee, "Flax: A neural network library and ecosystem for JAX," 2024.

[34] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, J. Fiene, A. Badri-Spröwitz, and L. Righetti, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, 2020.