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 rollouts 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], [3]. Reinforcement Learning (RL) [4], [5] has since expanded this potential, enabling policies that jointly handle contact decisions and sensor feedback [6], [7], [8]. 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 [9], [10], [11], [12], [13], [14].

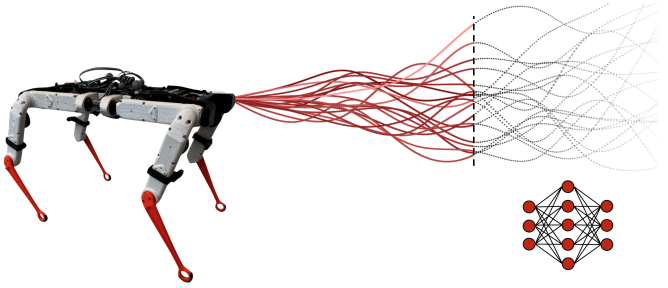

Fig. 1. The proposed TD-CD-MPPI controller enables the use of reduced control horizons by using a learned terminal value function to approximate long-term returns.

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.

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] or by leveraging different sampling strategies [22], [23], [24]. While being more sample efficient, such methods alone 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. Most, if not all, robotics tasks require constraints such as joint limits, safety boundaries, or collision avoidance [25]. Using soft constraints leads to no guarantee of satisfaction, and requires more extensive parameters tuning to ensure convergence and maintain scalability. Some earlier works have focused on enforcing constraint satisfaction in sampling-based methods using either projection

This work is supported by the European Union - Next Generation EU, Mission 4 Component 2 - CUP E53D23001130006 (STARLIT), the European project AGIMUS (under GA no.101070165) and ANITI (ANR-19-P3IA-0004).

[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
Digital Object Identifier: 10.1109/LRA.2025.3632612

techniques [26], probabilistic constraint satisfaction [27] or by learning a distribution that promotes sampling of feasible trajectories [28], [29]. 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. The second approach, mainly designed for collision avoidance in crowded environment, lacks generalization to generic constraints. The third approach, while effective, is usually complex to implement, computationally expensive, and scales badly [30].

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), i.e., one-step TD update), 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. Previous work leveraged offline path planning methods to estimate the VF [31]. Unlike heuristic terminal costs [15] or offline path planning, 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 [32], [8]. 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 minimization of costs instead of a maximization of rewards [19].

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 cumulative reward $R(U^{(i)})$ 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 reward:

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

where $\lambda > 0$ is a temperature parameter that controls the selectivity of the update, and the reward is related to the cost by $J(U^{(i)}) = -R(U^{(i)})$, recovering the standard MPPI form $\exp(-J(U^{(i)})/\lambda)$.

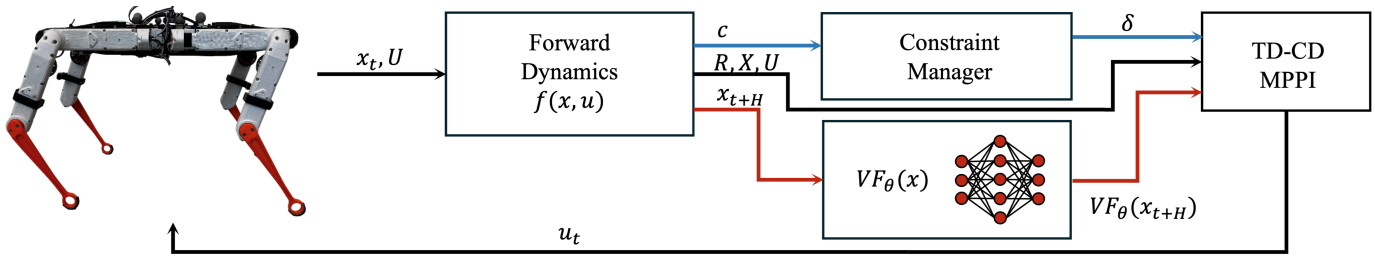

Fig. 2. TD-CD-MMPI controller structure, based on MPPI sample based optimization using the forward dynamics computed by a contact simulator. A terminal value function (VF), learned offline via temporal-difference learning, enables long-term reasoning with short roll-outs. Constraint manager modulate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

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, 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 [32]. 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.

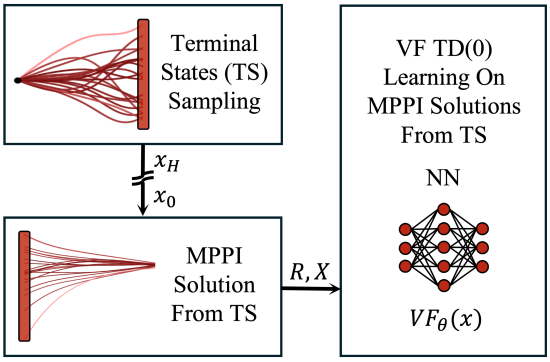

Fig. 3. Overview of the TD-learning data pipeline. (1) Terminal states are collected from MPPI rollouts. (2) From each terminal state, an optimal trajectory is generated under the constraint-discounted MPPI formulation. (3) The resulting transitions form a dataset for Temporal-Difference learning of the value function $V(x)$, later used as a terminal reward in TD-CD-MPPI.

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.

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 obtain a set of terminal states $x_{t+H+1}^{(j)}$ corresponding to each trajectory. These states do not correspond to any optimal behavior but constitute a representative set of reachable states for the given horizon (Fig. 3, Step 1).

Then, we generate a set of optimal trajectories using DIAL-MPC, as motivated in the beginning of this section, starting from the collected terminal states $x_{t+H+1}^{(j)}$, recording all transitions (x_t, u_t, r_t, x_{t+1}) along each trajectory (Fig. 3, Step 2). The reward r_t represents the immediate reward obtained at each time step of the rollout. Finally, the resulting dataset of

state-action-reward transitions is used to perform Temporal-Difference Learning (TD(0)) [33], training an approximate value function $V(x_t)$ that predicts future returns from state x_t (Fig. 3, Step 3). The standard TD update rule is

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

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:

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

Here, the factor $(1 - \delta_t)$ down-weights both immediate and future rewards when constraints are active, effectively penalizing constraint violations in the value estimate..

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

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

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

Algorithm 1 outlines the update of the value function with constraint-discounted returns, where minibatches of transitions are used to compute modified TD targets and update parameters via gradient descent with a soft target network. Algorithm 2 describes the full TD-CD-MPPI algorithm: perturbed action sequences are sampled, their rewards and constraint violations are evaluated, and the control sequence is updated using the corresponding constraint-discounted returns and value function estimates.

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. 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 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.

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

Input: Dataset $\mathcal{D} = \{(x_t, x_{t+1}, r_t, \delta_t)\}$, 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* $(x_t, x_{t+1}, r_t, \delta_t)$ **do**

    $V \leftarrow V_\theta(x_t), \quad V' \leftarrow V_{\theta'}(x_{t+1})$

    $y \leftarrow (1 - \delta_t) \cdot (r_t + \gamma V')$

    $\mathcal{L}(\theta) \leftarrow \frac{1}{B} \sum (V - y)^2$

    $g \leftarrow \nabla_\theta \mathcal{L}(\theta)$

    $\theta \leftarrow \theta - \alpha g$

    $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$

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**

    $\delta_h^{(j)} \leftarrow \max_i \left(p_i^{\max} \cdot \text{clip} \left(\frac{c_h^{(j)}[i]}{c_i^{\max}}, 0, 1 \right) \right)$

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

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

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}

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 [34] via the MJX physics engine, leveraging JAX [35] 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 [36], 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. We use a full-order dynamics model with control inputs $u \in \mathbb{R}^{12}$ and a full-state representation. To ensure fair comparisons, all controllers share the same parameter settings within each scenario. Specifically, the temperature parameter is set to $\lambda_{flat} = 0.2$ for flat terrain locomotion, and to $\lambda_{incline/steps} = 0.05$ for incline and step terrain. The covariance matrices follow the formulation in [19], with parameters $\beta_1 = 0.5$ and $\beta_2 = 0.9$. Metrics are computed as the mean across 20 episodes for each scenario.

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. In the constraint-discounted setting, the base discount factor $\gamma_0 = 0.9$ provides a balance between long-term reasoning and the stability of 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 in a velocity tracking task, 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 MPPI [15] and DIAL-MPC [19] ($H = 16$) with our VF-augmented controller ($H = 8$), showing that TD-CD-MPPI is able to consistently match the average reward of MPPI and 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. With the previously detailed hardware, the baseline controller requires an average of 15.3 ms per step, whereas our method achieves 7 ms.

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

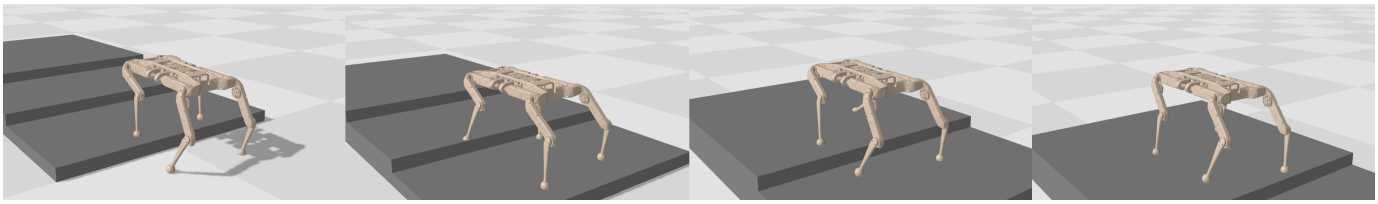

Fig. 4. Sequence of snapshots showing the quadruped climbing a three-step staircase using the proposed TD-CD-MPPI controller. The controller successfully handles terrain configurations not seen during training of the value function, leveraging real-time optimization to adapt online, while strictly enforcing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

TABLE II
TD-CD-MPPI VS. DIAL-MPC-REWARD WITH SHORT (SH-10) AND LONG (LH-20) HORIZONS, AVERAGED OVER 20 EPISODES.

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%**

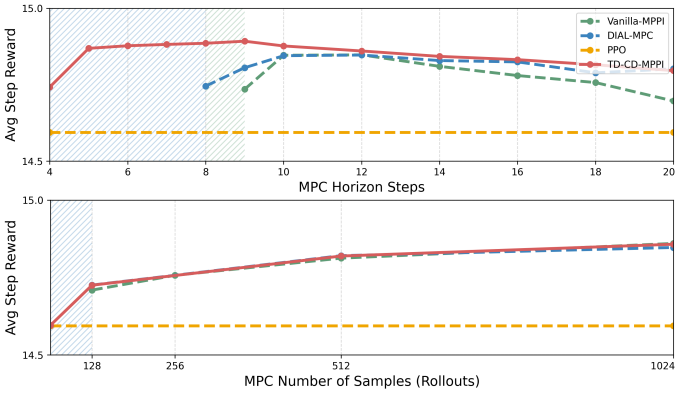

Fig. 5. Comparison of the average step reward obtained by Vanilla MPPI, DIAL-MPC and our proposed TD-CD-MPPI variant as a function of the MPC horizon length (Top) and number of samples (Bottom). The hatched green/blue region highlights the range where MPPI/DIAL-MPC fails to ensure convergence: the controller 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 reduced planning horizon and number of samples. For reference, the horizontal yellow line marks the mean reward achieved by the PPO baseline.

is due to the combination of properly handling the constraints and enabling long-horizon reasoning with a reduced number of rollouts.

We also emphasized the interest of MPC against RL by comparing TD-CD-MPPI with the same reward optimized by PPO [32]. 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:

- **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.
- **PPO**: a PPO implementation with encoded constraints via reward discounting [32].

TABLE III
CONSTRAINT ENCODING HAA/HEIGHT.

Reward-shaped Constraint	Constraint
$r = e^{-\frac{1}{\alpha}(\max(x_z - h_{\max}, 0))^2}$	$c = \frac{x_z - h_{\max}}{h_{\max}} \leq 0$
$r = e^{-\frac{1}{\alpha}(\max(\lvert q_{\text{HAA}}\rvert - q_{\lim}, 0))^2}$	$c = \frac{\lvert q_{\text{HAA}}\rvert - q_{\lim}}{q_{\lim}} \leq 0$

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%
CaT-PPO	0.9%	**0.01%**
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. 7 illustrates style and collision-avoidance constraint enforcement during flat terrain locomotion: joint angle limits on the hip abduction–adduction 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. We additionally benchmark against PPO [32], observing comparable enforcement of HAA joint limits and marginally improved body-height constraint satisfaction, although with increased safety margins.

For a broader comparison among MPPI based controllers, Table II reports a quantitative 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

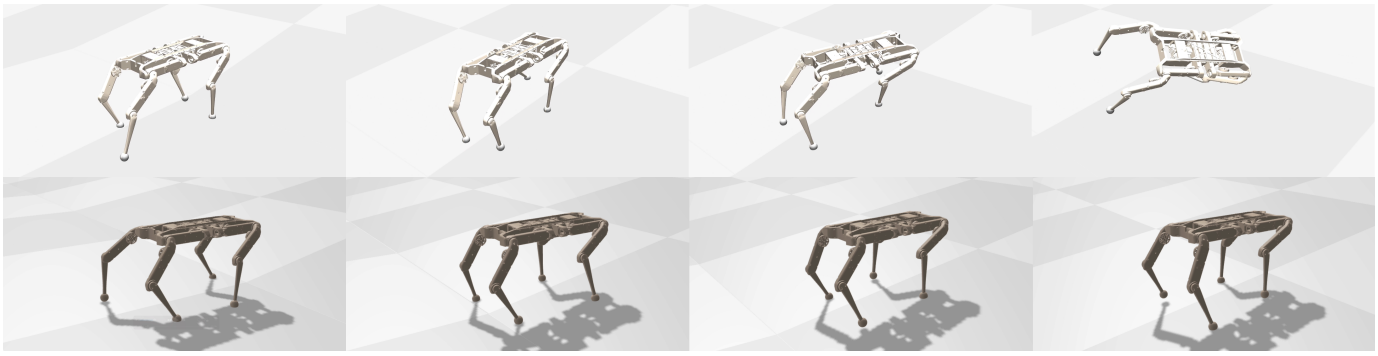

Fig. 6. Comparison on inclined terrain outside the training distribution between PPO (**top**) and TD-CD-MPPI (**bottom**), both using a lower-bound constraint on foot air time as gait shaping constraint. The policy trained with PPO becomes unstable under the out-of-distribution conditions, while TD-CD-MPPI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

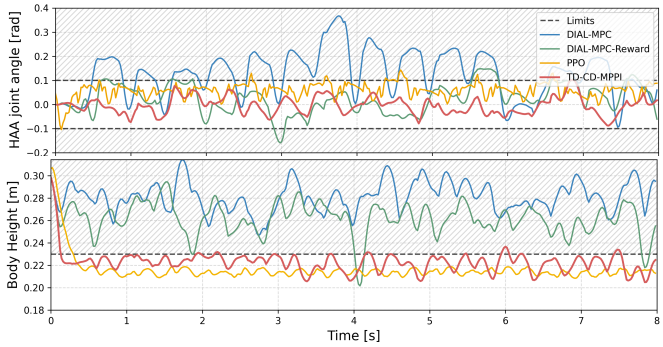

Fig. 7. Comparison of constraint violations during flat terrain locomotion. **Top:** Hip-roll joint angle limits. **Bottom:** Body height upper bound. We compare DIAL-MPC without constraints, DIAL-MPC with handcrafted reward penalties, PPO with encoded constraints, and TD-CD-MPPI.

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. While prior work [27] demonstrates that constraint satisfaction can be achieved with careful formulation and tuning, in our experiments the handcrafted penalty DIAL-MPC proved difficult to balance and the reported configuration reflects the best trade-off obtained without sacrificing efficiency. As an additional ablation, we also evaluated TD-CD-MPPI without the terminal value function. In this case, the controller achieves a similar level of constraint satisfaction as the full algorithm, but loses the advantages of long-horizon reasoning, exhibiting the same limitations as baseline methods when operating with shorter horizons.

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 [37]. 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 $0.5\,\text{kg}$ (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. 8, the controller generalizes and transfers to the real HW, maintaining the desired gait pattern and enforcing the physical constraints defined in the control formulation. Fig. 9 shows HAA joint constraint violations on the real hardware for DIAL-MPC-Reward and TD-CD-MPPI, with TD-CD-MPPI demonstrating minimal violations in the sim-to-real transfer. 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 for legged robots that combines long-term reasoning with constraint enforcement. By learning a terminal value function via temporal-difference learning, we approximate the infinite-horizon cost-to-go, reducing required rollout lengths and enabling efficient MPC under real-time constraints. Constraint violations are embedded into the discount factor, providing a practical mechanism for constraint enforcement without reward shaping. Together, these contributions improve numerical stability, reduce computational load, and allow deployment on smaller GPUs. Our experiments demonstrate strong generalization to out-of-distribution scenarios, highlighting the benefit of integrating value-driven learning with sampling-based MPC. Future work will explore more complex loco-manipulation tasks and deeper integration with reinforcement learning to further enhance control efficiency and scalability.

Acknowledgment

The authors thank Rafael Kourdis for providing his personal Solo12 robot and for his support during the hardware experiments, and Thomas Flayols for valuable assistance.

References

[1] M. Neunert, M. Stäuble *et al.*, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters (RAL)*, vol. 3, no. 3, pp. 1458–1465, 2018.

[2] C. Mastalli, R. Budhiraja *et al.*, "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[3] E. Dantec, M. Naveau *et al.*, "Whole-body model predictive control for biped locomotion on a torque-controlled humanoid robot," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2022.

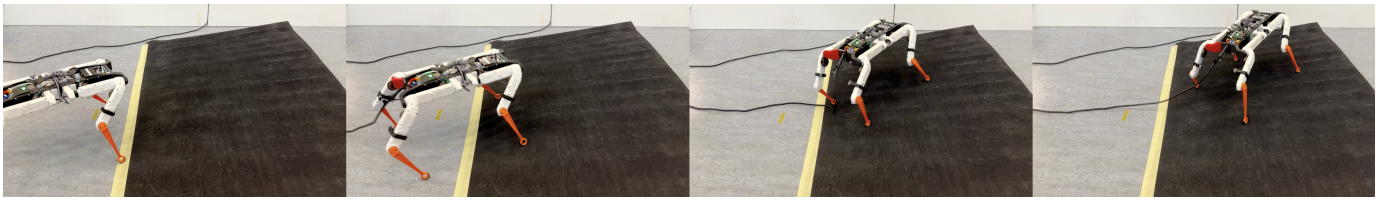

Fig. 8. Real-world evaluations on the Solo-12 quadruped. Deployment on inclined terrain unseen during training, where TD-CD-MPC adapts online while enforcing style and safety constraints.

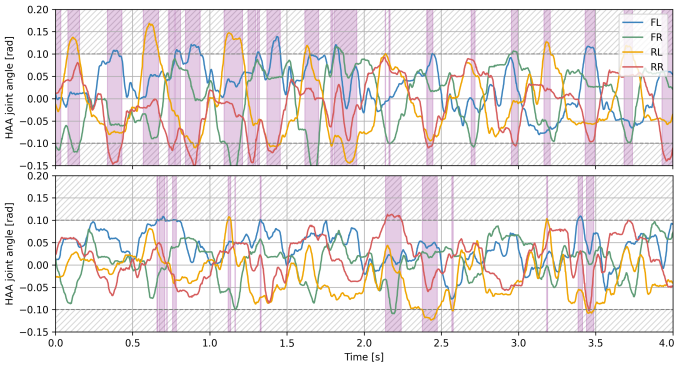

Fig. 9. Comparison from hardware experiments of HAA joint constraint violations during flat-terrain locomotion. **Top:** DIAL-MPC-Reward. **Bottom:** TD-CD-MPPI

[4] S. Gu, T. Lillicrap *et al.*, "Continuous deep q-learning with model-based acceleration," in *International Conference on Machine Learning (ICML)*, 2016.

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

[6] A. Agarwal, A. Kumar *et al.*, "Legged locomotion in challenging terrains using egocentric vision," in *Conference on Robot Learning (CoRL)*, 2023.

[7] D. Hoeller, N. Rudin *et al.*, "Anymal parkour: Learning agile navigation for quadrupedal robots," *Science Robotics*, vol. 9, no. 88, 2024.

[8] E. Chane-Sane, J. Amigo *et al.*, "Soloparkour: Constrained reinforcement learning for visual locomotion from privileged experience," in *Conference on Robot Learning (CoRL)*, 2024.

[9] N. Hansen, H. Su *et al.*, "TD-MPC2: Scalable, robust world models for continuous control," in *International Conference on Learning Representations (ICLR)*, 2024.

[10] R. Reiter, J. Hoffmann *et al.*, "Synthesis of model predictive control and reinforcement learning: Survey and classification," 2025.

[11] A. Jordana, S. Kleff *et al.*, "Infinite-horizon value function approximation for model predictive control," *IEEE Robotics and Automation Letters (RAL)*, 2025.

[12] G. Grandesso, E. Alboni *et al.*, "CACTO: Continuous actor-critic with trajectory optimization—towards global optimality," *IEEE Robotics and Automation Letters (RAL)*, vol. 8, no. 6, pp. 3318–3325, 2023.

[13] E. Alboni, G. Grandesso *et al.*, "CACTO-SL: Using sobolev learning to improve continuous actor-critic with trajectory optimization," in *Learning for Dynamics & Control Conference*, 2024.

[14] K. Lowrey, A. Rajeswaran *et al.*, "Plan online, learn offline: Efficient learning and exploration via model-based control," *arXiv preprint arXiv:1811.01848*, 2018.

[15] G. Williams, N. Wagener *et al.*, "Information theoretic MPC for model-based reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[16] G. Williams, A. Aldrich *et al.*, "Model predictive path integral control using covariance variable importance sampling," *arXiv preprint arXiv:1509.01149*, 2015.

[17] B. Hu and A. Linnemann, "Toward infinite-horizon optimality in nonlinear model predictive control," *IEEE Transactions on Automatic Control (T-AC)*, 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 (T-AC)*, vol. 53, no. 9, pp. 2100–2111, 2008.

[19] H. Xue, C. Pan *et al.*, "Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[20] Z. Yi, C. Pan *et al.*, "CoVO-MPC: Theoretical analysis of sampling-based mpc and optimal covariance design," in *Learning for Dynamics & Control Conference*, 2024.

[21] J. Yin, Z. Zhang *et al.*, "Trajectory Distribution Control for Model Predictive Path Integral Control using Covariance Steering," in *International Conference on Robotics and Automation (ICRA)*, 2022.

[22] I. S. Mohamed, J. Xu *et al.*, "Towards efficient MPPI trajectory generation with unscented guidance: U-MPPI control strategy," *IEEE Transactions on Robotics (T-RO)*, 2025.

[23] I. S. Mohamed, K. Yin *et al.*, "Autonomous Navigation of AGVs in Unknown Cluttered Environments: Log-MPPI Control Strategy," *IEEE Robotics and Automation Letters (RAL)*, vol. 7, no. 4, pp. 10 240–10 247, 2022.

[24] Leon, Yan *et al.*, "Output-Sampled Model Predictive Path Integral Control (o-MPPI) for Increased Efficiency," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[25] A. Haffemayer, A. Jordana *et al.*, "Collision avoidance in model predictive control using velocity damper," *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[26] J. Carius, R. Ranftl *et al.*, "Constrained stochastic optimal control with learned importance sampling: A path integral approach," *The International Journal of Robotics Research (IJRR)*, vol. 41, no. 2, pp. 189–209, Feb. 2022.

[27] I. S. Mohamed, M. Ali *et al.*, "Chance-constrained sampling-based mpc for collision avoidance in uncertain dynamic environments," *IEEE Robotics and Automation Letters (RAL)*, 2025.

[28] J. Sacks and B. Boots, "Learning sampling distributions for model predictive control," in *Conference on Robot Learning (CoRL)*, 2023.

[29] T. Power and D. Berenson, "Variational inference MPC using normalizing flows and out-of-distribution projection," *arXiv preprint arXiv:2205.04667*, 2022.

[30] A. Razmjoo, T. Xue *et al.*, "Sampling-based constrained motion planning with products of experts," *arXiv preprint arXiv:2412.17462*, 2024.

[31] N. Hatch and B. Boots, "The Value of Planning for Infinite-Horizon Model Predictive Control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[32] E. Chane-Sane, P.-A. Leziart *et al.*, "Cat: Constraints as terminations for legged locomotion reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[33] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, no. 1, pp. 9–44, Aug. 1988.

[34] E. Todorov, T. Erez *et al.*, "Mujoco: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[35] J. Bradbury, R. Frostig *et al.*, "JAX: composable transformations of Python+NumPy programs," 2018.

[36] J. Heek, A. Levskaya *et al.*, "Flax: A neural network library and ecosystem for JAX," 2024.

[37] F. Grimminger, A. Meduri *et al.*, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters (RAL)*, vol. 5, no. 2, pp. 3650–3657, 2020.