2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Reference-Free Sampling-Based Model Predictive Control

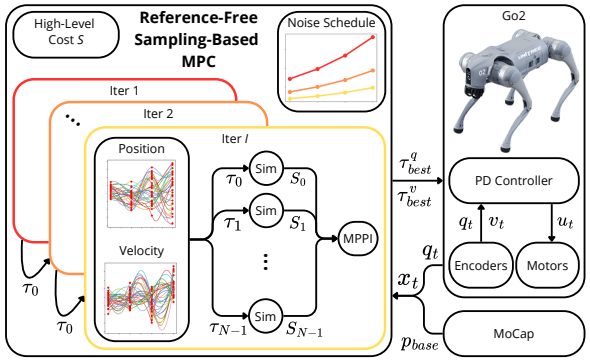
Fabian Schramm[1], Pierre Fabre[1], Nicolas Perrin-Gilbert[2] and Justin Carpentier[1]

***Abstract*— We present a sampling-based model predictive control (MPC) framework that enables emergent locomotion without relying on handcrafted gait patterns or predefined contact sequences. Our method discovers diverse motion patterns, ranging from trotting to galloping, robust standing policies, jumping, and handstand balancing, purely through the optimization of high-level objectives. Building on model predictive path integral (MPPI), we propose a cubic Hermite spline parameterization that operates on position and velocity control points. Our approach enables contact-making and contact-breaking strategies that adapt automatically to task requirements, requiring only a limited number of sampled trajectories. This sample efficiency enables real-time control on standard CPU hardware, eliminating the GPU acceleration typically required by other state-of-the-art MPPI methods. We validate our approach on the Go2 quadrupedal robot, demonstrating a range of emergent gaits and basic jumping capabilities. In simulation, we further showcase more complex behaviors, such as backflips, dynamic handstand balancing and locomotion on a Humanoid, all without requiring reference tracking or offline pre-training.**

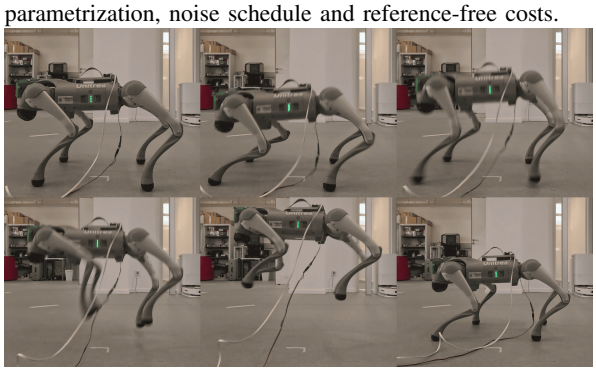
I. Introduction

Online robot control presents a trade-off between solution quality and computational efficiency. While learning-based methods such as reinforcement learning (RL) can generate impressive movements on complex robots, they require extensive offline training. They may also fail to generalize or adapt to new environments or tasks not seen during training [1]. Despite ongoing efforts to improve sample efficiency [2], RL remains orders of magnitude too slow for online adaptation, and simplified models used to accelerate training can hinder sim-to-real transfer [3], [4]. Typically, RL methods rely heavily on engineered rewards, such as phase clocks, air-time penalties, or foot-clearance objectives, to enforce structured contact behavior [5], [6], [7].

On the other hand, trajectory optimization (TO) relies on derivative information to find high-quality solutions, but it requires accurate and informative gradients. This may be unavailable or unreliable in contact-rich scenarios, leading most TO methods to use predefined contact sequences [8], [9]. Contact-implicit TO removes the need for predefined contact schedules [10] and has enabled advanced demonstrations of complex whole-body behaviors [11], [12]. However, many formulations rely on simplified contact models or approximations to remain tractable, which can introduce a mismatch with real dynamics and produce only approximate solutions in practice. These methods also necessitate handcrafted cost functions to obtain good contact sequences [12].



(a) Overview of the framework showing the Hermite spline parametrization, noise schedule and reference-free costs.



(b) Jumping sequence where the robot crouches and leaps to a height of 0.55 m.

Fig. 1: Our reference-free sampling-based MPC framework enables emergent jumping motion experimentally achieved on the Go2 robot without any guiding reference.

Sampling-based methods offer an attractive alternative by providing derivative-free optimization that is well-suited to parallel computation. This makes them particularly attractive for trajectory optimization problems with non-smooth dynamics. However, naive sampling approaches, such as random search, suffer from poor sampling efficiency and slow convergence, and may fail to converge to accurate solutions. Advances in sampling-based control over the past few years, particularly Model Predictive Path Integral (MPPI) control [13], despite their relative simplicity, have demonstrated promising results by incorporating more advanced sampling strategies and trajectory averaging schemes.

In this paper, we present a reference-free robotic control framework that challenges the paradigm requiring hand-crafted locomotion patterns. While recently introduced sampling-based MPC methods rely on gait references, whether through Raibert heuristics [14], foot swing trajecto-

[1]Inria - Département d'Informatique de l'École normale supérieure, PSL Research University firstname.lastname@inria.fr
[2]Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, ISIR

	DIAL-MPC [16]	RT-Whole-Body MPPI [14]	**Ours**
Samples	2048 - 4096	30	30 - 70
References	Swing foot	Raibert	**None**
Noise	Diffusion-inspired	Fixed	Diffusion-inspired
Hardware	GPU	CPU	CPU
Spline Type	Quadratic	Cubic	Hermite Cubic
Frequency	50 Hz	100 Hz	50 Hz
Simulator	MuJoCo MJX	MuJoCo C++	Simple

TABLE I: Comparison of sampling-based MPC for legged robot control. We combine the noise-annealing strategy with the computational efficiency of prior work while eliminating the need for reference gait requirements.

ries [15], or predefined contact sequences [16], our approach enables the discovery of emergent locomotion purely from high-level goal specifications within a cost function. In summary, the key contributions are:

- A sampling-based MPC framework that enables reference-free motion discovery without reliance on gait priors or offline pre-training.
- A cubic Hermite spline parameterization that jointly samples position and velocity control points to improve exploration and dynamic consistency. Constraining endpoint derivatives allows for bound-preserving rules that prevent overshooting joint limits.
- Demonstration of real-time performance on standard CPU hardware with as few as 30 samples, validated both on a real platform and in a high-fidelity simulator.

II. Related Work

Several works have explored sampling-based methods for robotic control. A common strategy is to refine a nominal action sequence iteratively, typically parameterized as a spline, using random perturbations. Howell et al. [17] evaluate predictive sampling (PS) as a zero-order baseline for comparison against methods like gradient descent, the iterative Linear Quadratic Regulator (iLQR) [18] and MPPI [13]. Despite its simplicity, PS achieves competitive performance and supports real-time tuning. The reported experiments, however, were restricted to torque-space splines in the `MuJoCo` [19] simulator and required workstation-class hardware.

More informative updates arise from leveraging statistics across all sampled trajectories, as in MPPI. This has shown strong performance in real-time control for racing, where parallel rollouts can be combined efficiently [20]. Related work has explored improving MPPI through structured proposal distributions, for example, via feature-based sampling [21]. Turrisi et al. [15] demonstrated MPPI on a 12-DOF quadruped using GPU-accelerated rigid-body dynamics, achieving 10k rollouts for a 12-step horizon in under 20 ms, but with gait frequencies fixed a priori. Extensions to MPPI include DIAL-MPC [16], which introduces a two-level annealing schedule inspired by diffusion models [22]. This method broadens exploration in early iterations and later horizon steps, while gradually refining actions closer to execution. Evaluation relies on GPU-based simulation with thousands of parallel rollouts and reference gait tracking. Building on DIAL-MPC, Crestaz et al. [23] add a constraint-enforcing mechanism and a terminal value function approximation for longer-horizon reasoning.

Recent work has also investigated CPU-based implementations. Alvarez-Padilla et al. [14] propose real-time MPPI in joint space at 100 Hz, with torques generated via a PD controller at 20 kHz. Their system rolls out 30–50 trajectories in `MuJoCo` [19], avoiding fixed contact sequences while still relying on reference heuristics for robust behavior.

In contrast to these approaches, our method emphasizes low-sample efficiency and generality. It operates entirely on the CPU, uses a limited number of rollouts, does not rely on gait priors, and employs cubic Hermite splines to sample both position and velocity targets for the PD controller. Unlike prior joint-space sampling methods, which provide only position references (with velocity targets set to zero), our formulation yields dynamically consistent PD targets and enables richer exploration.

III. Methodology

We formulate our controller within the MPPI framework [13], which provides a sampling-based approach to trajectory optimization in a receding-horizon loop. At each control step, MPPI maintains a nominal control sequence over a finite horizon, perturbs this sequence with Gaussian noise, evaluates the resulting trajectories under a cost objective, and updates the nominal sequence via importance-weighted averaging before executing the first control input.

Building on this foundation and using a diffusion-inspired annealing scheme that structures noise, we introduce several practical enhancements. We use a cubic Hermite spline parameterization that jointly optimizes position and velocity control points to improve exploration and enforce dynamically consistent trajectories, and a reference-free cost formulation that supports emergent locomotion without gait priors. To further improve stability and convergence, we integrate state prediction and warm-starting strategies to maintain temporal consistency across optimization steps. A complete overview of the framework is shown in Fig. 1a and Tab. I presents a comparison with recent MPPI-based methods.

A. Search-space parametrization

Defining the search space is critical for effective random-search optimization. In contrast to approaches that sample directly in torque space, often together with massively parallel GPU simulation [24], we sample reference trajectories in joint space and track them using a PD controller that maps joint-space references to torque commands. This decouples high-level trajectory exploration from low-level torque regulation, enabling efficient CPU-based simulation at high frame rates. In our implementation, this requires only 20–30 parallel trajectories for quadruped control and 60–70 for humanoid control.

Cubic Hermite spline parameterization. For each actuated degree of freedom, the control sequence is parameterized by a cubic Hermite spline with K nodes. The optimization

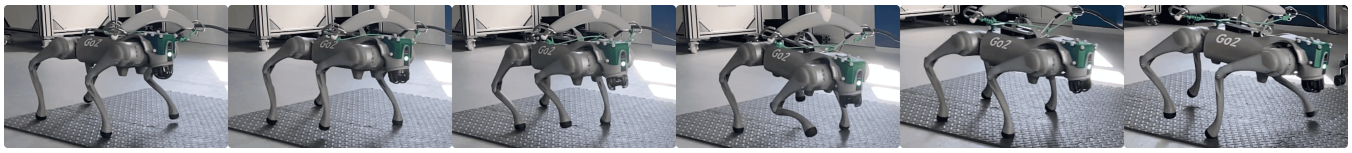


Fig. 2: Sequence illustrating the discovered walking gait on the Go2 quadruped.

variables are the node parameters

$$\theta_k = (\theta_k^q, \theta_k^v), \qquad k = 0, \ldots, K-1,$$

where $\theta_k^q = q(t_k)$ and $\theta_k^v = \dot{q}(t_k)$ denote the joint position and joint velocity at node k. Throughout the paper, the superscript ^{v} denotes joint-velocity quantities associated with q. The nodes are uniformly spaced in time with interval Δt, and we consider now one spline segment over $t \in [t_k, t_{k+1}]$. With the normalized local time

$$s = \frac{t - t_k}{\Delta t} \in [0,1], \tag{1}$$

the joint trajectory is reconstructed using the standard cubic Hermite interpolant [25]

$$\begin{aligned} q(t) = h_{00}(s)\,\theta_k^q \;&+ h_{10}(s)\,\Delta t\,\theta_k^v \\ &+ h_{01}(s)\,\theta_{k+1}^q + h_{11}(s)\,\Delta t\,\theta_{k+1}^v, \end{aligned} \tag{2}$$

$$\text{where} \quad h_{00}(s) = 2s^3 - 3s^2 + 1, \quad h_{10}(s) = s^3 - 2s^2 + s, \tag{3}$$

$$h_{01}(s) = -2s^3 + 3s^2, \qquad h_{11}(s) = s^3 - s^2 \tag{4}$$

are the cubic Hermite basis functions. This parameterization is applied only to the actuated joints. The floating-base motion is not parameterized directly. Instead, it emerges from the forward dynamics rollout under the sampled joint references and contact interactions.

Compared with position-only spline parameterizations, cubic Hermite splines control both node positions and node velocities. This provides additional local control over the shape of the generated reference trajectories, as illustrated in Fig. 3. To reduce overshoot near joint limits, we limit per-node derivatives based on the distance to the nearest bound.

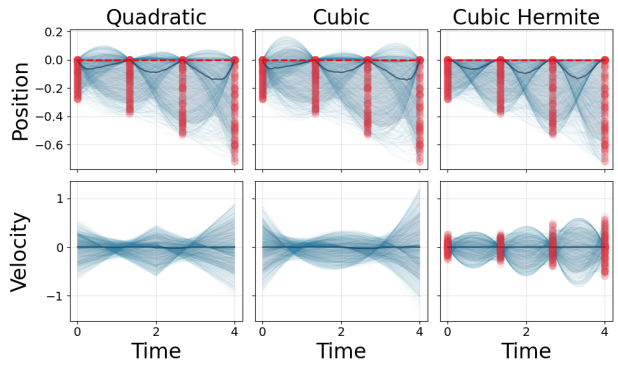


Fig. 3: Comparison of spline parameterizations for the same interpolation points (red). Quadratic, cubic, and cubic Hermite splines produce different position and velocity profiles. By parameterizing both node positions and velocities, cubic Hermite splines provide additional local shape control. Derivative clamping reduces overshoot near joint limits.

For node value $\theta_k^q \in [q_{min}, q_{max}]$ and node spacing Δt, we impose

$$|\theta_k^v| \le \frac{\min\{q_{max} - \theta_k^q,\, \theta_k^q - q_{min}\}}{\Delta t/2}. \tag{5}$$

This inexpensive clamp reduces mid-interval excursions and improves preservation of bounds in practice. In Fig. 3, the red dashed lines indicate the upper position bounds. While quadratic and cubic position splines often overshoot these bounds, our cubic Hermite formulation respects them throughout the trajectory. In the current formulation, the spline parameterization enforces position bounds via the derivative clamp above, but does not explicitly enforce joint-velocity limits at any intermediate time. In practice, feasibility is further shaped by the rollout dynamics, PD tracking, and the cost function.

B. Noise annealing

The annealing schedule is motivated by the iterative nature of receding-horizon MPC and follows the diffusion-inspired design introduced in DIAL-MPC [16]. A nominal control sequence is refined over I internal iterations at each time step before executing the first action, after which the horizon advances. Consequently, control decisions farther in the horizon receive more updates (and thus can be explored more aggressively) while controls near execution should be more stable, with less variance.

Trajectory-level annealing. We therefore reduce the variance of exploration across iterations, transitioning from exploration to exploitation as the nominal trajectory is refined. For spline control points, this corresponds to shrinking the sampling covariance over iterations $i = I, \ldots, 1$:

$$\det\big(\Sigma_\theta^i\big) \propto \exp\Big(-\frac{I-i}{\beta_1 I} K d_u\Big), \tag{6}$$

where β_1 is a temperature parameter, K is the number of spline points, and d_u is the per-node control dimension.

Action-level annealing. We also increase exploration of control points corresponding to later actions in the horizon. For spline node index $k \in \{0, \ldots, K-1\}$, this yields

$$\det\big(\Sigma_{\theta_k}^i\big) \propto \exp\Big(-\frac{K-k}{\beta_2 K} d_u\Big), \tag{7}$$

with β_2 controlling the horizon-wise decay. In our implementation, we use an isotropic, multiplicative combination of the two effects and parameterize the per-node covariance:

$$\Sigma_{\theta_k}^i = \exp\Big(-\frac{I-i}{\beta_1 I} - \frac{K-k}{\beta_2 K}\Big)\mathbf{I}, \tag{8}$$

where $\mathbf{I} \in \mathbb{R}^{d_u \times d_u}$. The schedule in Eq. (8) can be precomputed and cached once. It provides larger variance for later

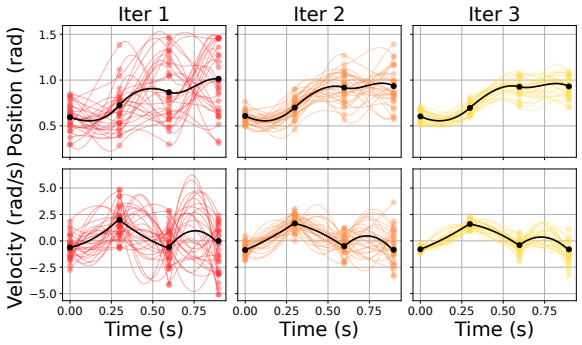


Fig. 4: The nominal trajectory (black) evolves through spline control points that are updated iteratively. New perturbed spline points are sampled around the nominal points with annealing noise according to Eq. (8).

nodes and earlier iterations, and smaller variance for near-term nodes and later iterations. Fig. 4 visualizes this behavior for a representative horizon and iteration count.

C. Algorithm description

The algorithm maintains a nominal sequence of spline control points in joint position θ^q and velocity θ^v, defining a smooth trajectory over the prediction horizon via cubic Hermite interpolation. The position control points are initialized from a stable standing configuration and the velocity control points are initialized to zero. At each control step, new perturbed control points (θ_n^q, θ_n^v) are sampled around the nominal points with structured noise from the annealing scheme. Interpolating these perturbed points yields a batch of candidate trajectories $\tau_n = (\tau_n^q, \tau_n^v)$, which are rolled out in `Simple` [26], evaluated under the task cost, and assigned importance weights. The nominal sequence is then updated via importance-weighted averaging, and new control points are extracted by resampling the updated trajectory at spline node times, uniformly distributed across the horizon.

The exponential weighting scheme considers the relative quality of all candidates (see Eq. (9)) and enables effective trajectory synthesis. The softmax smoothing concentrates probability mass around high-quality solutions and can be interpreted as performing approximate natural gradient descent on a smoothed surrogate function [27]. The normalized weight ω_n for each trajectory n is computed as

$$\omega_n = \frac{\exp\big(-S_n/\lambda\big)}{\sum_j \exp\big(-S_j/\lambda\big)}, \tag{9}$$

where S_n is the cost of trajectory n and λ is a temperature parameter that controls the selectiveness of the weighting. In practice, we implement the calculation of ω_n using min-max-normalization of costs [28]:

$$\widehat{S}_n = \frac{S_n - S_{\min}}{S_{\max} - S_{\min}}, \quad S_{\max} = \max_j S_j, \quad S_{\min} = \min_j S_j, \tag{10}$$

and the weights are computed as

$$\omega_n = \frac{\exp\big(-\widehat{S}_n/\lambda\big)}{\sum_j \exp\big(-\widehat{S}_j/\lambda\big)}. \tag{11}$$

Algorithm 1 Reference-Free MPPI

Require: parameters H, K, I, N, λ, (β_1, β_2), (K_p, K_d)

init. nominal spline control points $\theta^q \leftarrow q_0, \theta^v \leftarrow 0$
$\tau_{best} := \tau_0 = (\tau_0^q, \tau_0^v) = \text{CubicHermite}(\{\theta^q\}, \{\theta^v\})$
define annealed noise factors σ_k^i using Eq. (8)
for every control step t **do**
  state prediction and warm-start (Sec. III-E)
  $\tau_0 \leftarrow \tau_{best}$ ▷ init from best known trajectory
  for each planning iteration $i \in \{I, \dots, 1\}$ **do**
    extract nominal control points θ^q, θ^v from τ_0
    for each sample $n \in \{1, \dots, N-1\}$ **do**
      for each spline point $k \in \{0, \dots, K-1\}$ **do**
        sample $\theta_{n,k}^q \sim \mathcal{N}(\theta_k^q, \sigma_k^i \cdot \text{scale}_q)$
        sample $\theta_{n,k}^v \sim \mathcal{N}(\theta_k^v, \sigma_k^i \cdot \text{scale}_v)$
      end for
      $\tau_n = (\tau_n^q, \tau_n^v) = \text{CubicHermite}(\{\theta_n^q\}, \{\theta_n^v\})$
    end for
    simulate $\{\tau_n\}_{n=0}^{N-1}$ and compute costs $\{S_n\}_{n=0}^{N-1}$
    compute weights ω_n using Eq. (11)
    update nominal τ_0 using Eq. (12)
    update τ_{best} (Sec. III-D)
  end for
  compute torque u_t using Eq. (13)
end for

The nominal control sequence τ_0 is then updated as a weighted average of the candidates:

$$\tau_0^q = \sum_n \omega_n \tau_n^q, \qquad \tau_0^v = \sum_n \omega_n \tau_n^v. \tag{12}$$

This update gradually shifts the nominal trajectory towards higher-quality solutions while retaining exploration diversity, and new nominal control points can be extracted. The executed torque command is computed by a low-level PD controller from the computed best joint position and velocity targets to actuator torques

$$u_t = K_p\big(\tau_{best}^q[0] - q_t\big) + K_d\big(\tau_{best}^v[0] - v_t\big), \tag{13}$$

where q_t, v_t are the measured joint positions and velocities, and K_p, K_d are diagonal gain matrices. Torques are clipped to actuator limits before application.

Overall, we build on MPPI with noise annealing and extend it with two key enhancements that enable reference-free locomotion discovery with a low number of samples: (1) cubic Hermite spline sampling, which jointly perturbs position and velocity control points (θ^q, θ^v) with physics-aware scaling, and (2) best trajectory tracking, which separates the evolving nominal sequence τ_0 from the executed actions τ_{best} for robustness and as a safeguard for consistent performance across iterations. Alg. 1 summarizes our complete approach, integrating these aspects within the MPPI framework.

D. Best trajectory tracking

One distinction is the separation between trajectory evolution and action execution. While the nominal trajectory τ_0 evolves via standard MPPI importance-weighted averaging

across iterations, the robot always executes actions from τ_{best}, the best-performing trajectory tested in simulation rollout. This serves two purposes. First, it ensures that executed actions always come from a verified, fully-simulated trajectory rather than from an untested weighted mixture, providing safety guarantees. Second, it prevents performance degradation during iterative refinement by maintaining monotonic improvement in the quality of the executed solution.

E. Real-time state prediction and warm-starting strategy

A practical challenge in real-time MPC is that both the sampling and optimization stages require time. In our setup, a full MPPI update with three iterations and 30 samples requires typically $\Delta t \approx 20-30\,\text{ms}$ for a quadruped, during which the robot continues to execute the previously optimized trajectory. By the time the new solution is available, the robot has already moved on, so directly applying its first control would be inconsistent.

To address this issue, we predict the state the robot will reach when the optimization is complete. Starting from the last known state $x_{t-\Delta t}$, we simulate forward using the prefix of the previously best trajectory τ_{best}:

$$x_t = \text{simulate}\big(x_{t-\Delta t}, \tau_{\text{best}}[0:\lfloor \Delta t/dt \rfloor], \Delta t\big). \tag{14}$$

This predicted state x_t is then used as the starting point of the subsequent optimization instance. We then shift τ_{best} forward by the number of actions already executed during the computation delay,

$$\tau_{\text{best}}[h] \leftarrow \tau_{\text{best}}[h+\lfloor \Delta t/dt \rfloor], \quad h \in [0, H-\lfloor \Delta t/dt \rfloor), \tag{15}$$

and pad the tail by repeating the final action to preserve the horizon length. This maintains the solution continuity across receding-horizon steps and avoids re-optimizing from scratch. To improve convergence across control steps, we initialize the nominal trajectory τ_0 with the shifted best trajectory from the previous optimization τ_{best}.

Our approach differs from prior work that handles computation delays by constraining the first $\lfloor \Delta t/dt \rfloor$ actions across all parallel rollouts to match the actions executed during computation. While conceptually simpler, this wastes computational budget by forcing identical initial actions across all sampled trajectories, reducing exploration diversity. Instead, our state prediction strategy optimizes from the predicted future state, allowing all samples to explore freely from the start of their planning horizons, thereby maximizing the effective use of the limited sample budget.

IV. Experiments

We evaluate our framework through real-world hardware experiments on the Go2 quadruped and complementary studies in simulation, including validation on the G1 humanoid. The experiments are designed to highlight three key aspects: (1) the emergence of diverse locomotion strategies without reliance on gait references, (2) real-time execution with minimal computational resources, and (3) adaptability across platforms, tasks, and challenging behaviors. Demonstrations from both hardware and simulation are included in the accompanying video.

TABLE II: Cost weights for different experiments.

Task	w_h	w_{orient}	w_q	$w_{c,vel}$	$w_{c,force}$	w_H
Walking	1e2	10	0.0	0.5	5e-2	2.5e3
Jumping	1.0	0.5	0.3	1.0	5e-4	2e3
Handstand	50	10	0.3	1.0	5e-4	0.0
Backflip	1.0	0.5	0.3	1.0	5e-4	2e3
Bipedal	10	1.0	0.3	1.0	5e-4	2e2

A. Experimental setup

Real-world experimental hardware (Go2 quadruped only). Experiments are conducted on the Unitree Go2, a 16 kg quadruped with 12 actuated joints. Joint states are measured from onboard encoders, while accurate global tracking is provided by a 300 Hz motion capture system. The high-level MPC controller runs at 50 Hz on a Mac Studio M3 equipped with 16 efficient cores and outputs desired joint positions and velocities, which are then tracked by a low-level PD control on the robot running at 12 kHz.

Experiments in simulation (Go2 quadruped and G1 humanoid). To complement hardware experiments, we use the `Simple` physics simulator [26], which provides accurate and efficient frictional contact dynamics simulation by proper handling of nonlinear complementarity contact models [29]. Simulation experiments serve two purposes: (i) showing behaviors that are unsafe or impractical to test on hardware, such as backflips, aggressive jumps, and high-speed locomotion up to $2.0\,\text{m}\,\text{s}^{-1}$, and (ii) demonstrating cross-platform generalization by validating our method on the Unitree G1 humanoid.

Cost function design. All behaviors are driven by a modular cost formulation composed of running costs $c_t(x_t,u_t)$ and a terminal cost $c_T(x_H)$:

$$S = \sum_{t=0}^{H-1} c_t(x_t,u_t) + c_T(x_H). \tag{16}$$

The running cost combines residual terms on base motion, joint states, and contacts:

$$\begin{aligned} c_t(x_t,u_t) = {} & w_h\,|p_{base,z}-p_{des,z}| + w_{orient}\,\|\log_3(R_{base}^T R_{des})\|_2^2 \\ & + w_q\,\|q-q_0\|_2^2 + w_{c,vel}\,\|v_c\|_1 + w_{c,force}\,\|f_c-f_0\|_1, \end{aligned} \tag{17}$$

where p_{base} and R_{base} denote the base position and orientation, q the joint angles, v_c contact velocities, f_c contact forces, and q_0, f_0 denote constant initial joint configurations and forces. Quantities with the subscript "des" indicate task-specific desired targets. The terminal cost encourages velocity tracking through base displacement:

$$c_T(x_H) = w_H\,\|p_{base}(x_H) - p_{target}\|_1, \tag{18}$$

with $p_{target} = p_{base}(x_0) + \dot{p}_{des}\cdot H\cdot dt$. Task-specific behaviors are obtained by adjusting the various weights while leaving the algorithmic framework unchanged. Tab. II summarizes the weights used across experiments.

B. Motion discovery

A central contribution of our work is the demonstration of reference-free motion discovery in real time across diverse

tasks and robot morphologies. We evaluate the quadruped's ability to discover motion strategies purely from high-level goals and show that our framework generalizes to a humanoid robot, enabling walking without tuning gait parameters or other motion priors. For the more aggressive behaviors shown in simulation, we do not claim fully optimized or hardware-ready motions, but rather proof-of-concept behaviors obtained from the same generic framework.

1) Velocity-adaptive gaits: In simulation and on the real Go2 robot, we specify the desired forward velocity in the cost function without providing any explicit gait references or contact sequence. We use an extended planning horizon of 0.9 s that is important for locomotion discovery. In contrast to reference-tracking methods that can use shorter horizons of 0.4 s to follow predefined patterns, emergent gait discovery requires sufficient horizon time to evaluate locomotion stability and quality. For slow walking in particular, a 0.9 s window allows the optimization to assess whether a motion pattern leads to stable periodic locomotion or to a fall. We evaluate three representative velocity commands:

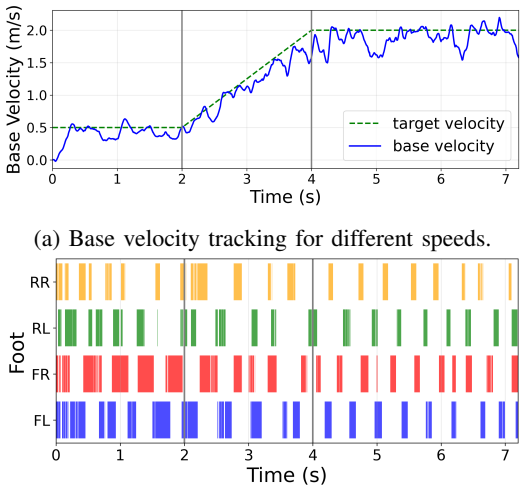
- Standing still ($0\,\mathrm{m\,s^{-1}}$): The robot consistently adopts a stable posture with the 4 feet in contact, maintaining balance and rejecting disturbances through leg adjustments and stepping.
- Trotting ($0.5\,\mathrm{m\,s^{-1}}$): A trotting gait emerges, characterized by alternating pairs of legs in contact.
- Galloping ($2.0\,\mathrm{m\,s^{-1}}$): The robot transitions to a more dynamic gait with extended flight phases, resembling a galloping or bounding pattern. This transition occurs smoothly as the velocity command increases.

To analyze the different gaits, Fig. 5a shows base-velocity tracking as the commanded speed is ramped from $0.5\,\mathrm{m\,s^{-1}}$ to $2.0\,\mathrm{m\,s^{-1}}$. Fig. 5b shows the contact patterns, highlighting the transition from trotting to more dynamic gaits as the velocity command increases. The resulting behavior on the real Go2 platform is illustrated in Fig. 2.

2) Robustness: To evaluate robustness, we conduct real-world disturbance tests over 10 min of continuous operation. We command the robot to maintain a target position (x, y, z) and yaw angle while systematically applying various disturbances. We begin with direct physical perturbations that push the robot from multiple directions (sides, top, diagonally) and manually pull individual legs. The controller resists these external perturbations and generates corrective stepping motions without falling.

When we rotate the mattress on which the robot is standing, it responds by generating turning walking motions to restore its original yaw angle. If the mattress is pushed or pulled, the robot walks to recover its original position. In the most extreme test, we lift the robot into the air, rotate it by more than 90 deg, and set it back down; the robot then walks back to its target pose. These tests are illustrated in the companion video.

3) Jumping: We demonstrate diverse jumping behaviors by specifying only sparse task-level objectives where the goal is to jump vertically, turn the base, and land safely.



(a) Base velocity tracking for different speeds.

(b) Contact pattern of the emergent gaits.

Fig. 5: Smooth transitioning from trotting to galloping as velocity commands change, showing adaptive gait discovery.

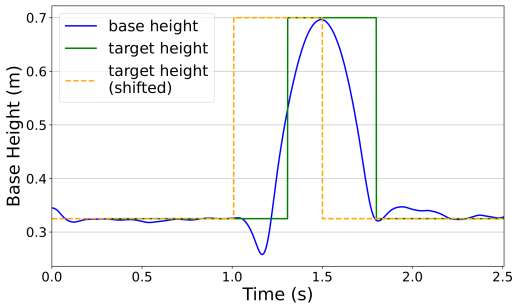


Fig. 6: Robot base height during vertical jumping. When commanded to reach 0.7 m (green), the robot discovers a jumping strategy. The orange dashed line shows when the terminal height objective enters the MPC horizon, triggering the robot to initiate its jump sequence (blue).

Vertical jumping: Commanding a target base height increase ($0.325\,\mathrm{m} \rightarrow 0.7\,\mathrm{m}$) as terminal cost via a step function, the robot discovers a complete jumping strategy. As soon as the terminal height objective becomes visible in the current MPC horizon (see orange dashed line in Fig. 6), the robot initiates a multi-phase jumping maneuver. First, it crouches by flexing its legs, then rapidly extends them to launch into the air, reaching a peak height of 0.7 m above the ground. During flight, the robot controls its body orientation to maintain stability and prepare for landing. We have successfully validated this jumping behavior on the real Go2 robot platform, as shown in Fig. 1b.

Backflip: By specifying a target pitch orientation (180°) at the highest point, the robot discovers a complete backflip maneuver. It coordinates a pre-jump crouch, an explosive takeoff that generates both vertical impulse and angular momentum, mid-air rotation control, and precise landing absorption to achieve the desired orientation and height. Starting from the elevated platform of 0.5 m, the robot completes a full 360° rotation with landing as shown in Fig. 8b.

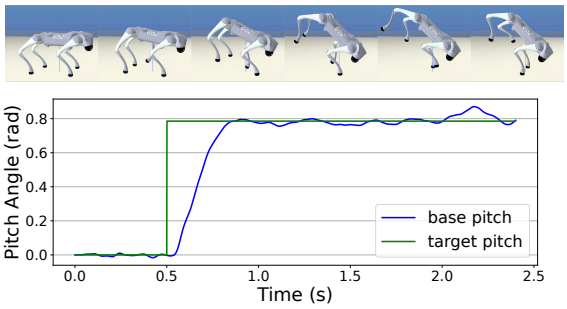
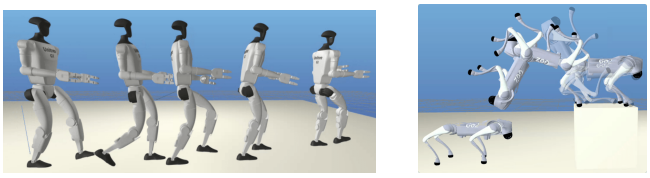


Fig. 7: Base pitch trajectory during handstand pose.



(a) Humanoid locomotion. (b) Backflip maneuver.

Fig. 8: Emergent dynamic behaviors in simulation.

4) Dynamic handstand balancing: Starting from a standard quadrupedal stance with a level base, when commanded to achieve an inverted pose (45° pitch angle), the robot executes a dynamic swing maneuver to transition into a handstand configuration. Then, the controller identifies stabilizing strategies by continuously repositioning the legs and adjusting contact patterns. Fig. 7 illustrates simulation snapshots of emergent contact-making and breaking patterns, where the legs dynamically reposition to maintain balance. The accompanying plot shows that the handstand pose is stably maintained for about 1 s with low orientation error.

5) Humanoid locomotion: To highlight the generality of our reference-free framework beyond quadrupedal locomotion, we also evaluate it on the G1 humanoid in simulation. Despite the shift in morphology and increase in DoFs (from 12 to 37), the same algorithm successfully discovers a walking gait from the exact high-level cost without modification, as displayed with overlayed snapshots in Fig. 8a. While all previously demonstrated behaviors were achieved in real-time, the humanoid experiment requires an increase from 30 to 70 parallel sample trajectories due to the increased complexity of bipedal balance control, which prevents real-time execution. This cross-platform study underlines a key advantage of our approach. The identical framework, cost functions, and spline parameterization transfer across robot morphologies, enabling rapid deployment on new platforms. However, these humanoid results are currently limited to simulation and should be interpreted as a proof of concept for motion discovery rather than as hardware-ready whole-body behaviors. The discovered motions are not yet optimized for smoothness, efficiency, or robustness to real-world sensing and actuation constraints.

C. Ablation studies

To better understand the contribution of our algorithmic components, we perform ablation studies that isolate and evaluate individual design choices. Specifically, we investigate two central enhancements of our framework:

TABLE III: Ablation results. Values are success rates (over 10 seeds) and mean ± std for continuous metrics.

Variant	Walk		Handstand		Jump
	Succ.	Cost	Succ.	Cost	Max h. (cm)
Hermite (ours)	10/10	913.9 ± 25.8	10/10	811.7 ± 253.4	68.4 ± 0.2
Cubic	6/10	1124.1 ± 135.7	10/10	1540.2 ± 655.8	63.8 ± 0.4
Quadratic	0/10	–	9/10	1882.2 ± 493.9	47.7 ± 0.4
Best traj. (ours)	10/10	913.9 ± 25.8	10/10	811.7 ± 253.4	68.4 ± 0.2
Nominal only	10/10	924.9 ± 24.3	10/10	844.1 ± 203.2	67.4 ± 0.3

(1) Cubic Hermite splines. We compare our cubic Hermite spline parameterization with velocity targets against cubic splines without velocity targets and quadratic splines. The goal is to quantify the improvement in trajectory quality and stability achieved by explicitly controlling both position and velocity at spline nodes. Note that we are using the best trajectory tracking mechanism III-D in all cases.

(2) Best trajectory tracking. We study the effect of executing actions from the best sampled trajectory τ_{best} rather than from the evolving nominal trajectory τ_0. This ablation tests whether explicitly safeguarding execution against untested mixtures yields more reliable performance. In both cases, trajectories are parameterized using Hermite splines. We then track how often the optimizer failed to improve upon the shifted previous solution: walking (27.9%), jumping (29.7%), and handstand (27.1%).

For both ablations, we evaluate performance across multiple tasks, walking, handstand, and jumping, using 10 randomized seeds per setting. The primary metric is the average trajectory cost, which directly reflects the optimization objective. In Tab. III, we report mean ± standard deviation to highlight consistency across runs. Failures are recorded when the robot base hits the ground. For the handstand task, Fig. 9 additionally shows the base pitch trajectories. For the jumping task, we report the mean ± standard deviation of the maximum reached height to provide a more interpretable performance metric. Across all tasks, our Hermite spline parameterization, combined with best-trajectory tracking, yields lower mean trajectory costs and lower variance. This advantage is particularly evident in jumping, where our method achieves the highest mean height (h), and in handstands, where cubic and quadratic splines exhibit larger variance and occasional failures.

V. Discussion and Conclusion

We presented a sampling-based MPC framework that enables reference-free locomotion by combining cubic Hermite spline parameterization with diffusion-inspired noise annealing. In contrast to prior MPPI approaches that rely on thousands of GPU rollouts and gait priors, our method achieves real-time performance on a CPU with as few as 30 samples. Experiments on the Go2 quadruped and G1 humanoid show that the same framework can produce diverse

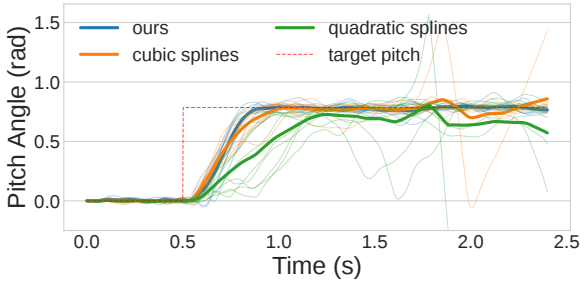


Fig. 9: Handstand ablation: base pitch trajectories for different spline parameterizations during the handstand task.

locomotion behaviors across tasks and robot morphologies without reference tracking or predefined contact sequences.

A limitation of the current formulation is that the resulting motions are not always fully optimized with respect to smoothness or efficiency. This follows from our design choice to sample spline control points in joint space and delegate torque-level execution to a PD controller, which reduces the search space but also limits fine-grained optimization compared with direct torque optimization.

This is a drawback of the present formulation, but also a trade-off that makes the approach useful as a motion-discovery mechanism. The framework provides a flexible source of candidate behaviors, contact sequences, and strategies that can serve as warm starts for higher-level solvers or be integrated into larger control pipelines. Future directions include improving the noise sampling scheme and integrating our method with trajectory optimization or learning-based refinement to transform diverse exploratory motions into polished, task-specific controllers.

Acknowledgments

Supported by the French government via ANR under France 2030 (PEPR O2R, PR[AI]RIE-PSAI ANR-23-IACL-0008, RODEO ANR-24-CE23-5886); the EU (ARTIFACT 101165695, AGIMUS 101070165); and Région Île-de-France (DIM AI4IDF). Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the funding agencies.

References

[1] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[2] Q. Le Lidec, L. Montaut, Y. de Mont-Marin, F. Schramm, and J. Carpentier, "End-to-end and highly-efficient differentiable simulation for robotics," *arXiv preprint arXiv:2409.07107*, 2024.

[3] M. P. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011.

[4] Y. Zhao, L. Mou, and B. Chazelle, "Sim-to-real transfer in robotics: A review," *IEEE Transactions on Robotics*, vol. 36, no. 5, 2020.

[5] J. Hwangbo, J. Lee, and et al., "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019.

[6] M. Aractingi, P.-A. Léziart, T. Flayols, J. Perez, T. Silander, and P. Souères, "Controlling the solo12 quadruped robot with deep reinforcement learning," *Scientific Reports*, vol. 13, no. 1, July 2023.

[7] S. Ha, J. Lee, M. van de Panne, Z. Xie, W. Yu, and M. Khadiv, "Learning-based legged locomotion: State of the art and future perspectives," *The International Journal of Robotics Research*, vol. 44, no. 8, pp. 1396–1427, 2025.

[8] R. Budhiraja, J. Carpentier, C. Mastalli, and N. Mansard, "Differential dynamic programming for multi-phase rigid contact dynamics," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 1–9.

[9] W. Jallet, A. Bambade, E. Arlaud, S. El-Kazdadi, N. Mansard, and J. Carpentier, "PROXDDP: Proximal Constrained Trajectory Optimization," *IEEE Transactions on Robotics*, Mar. 2025.

[10] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[11] 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, July 2018.

[12] G. Kim, D. Kang, J.-H. Kim, S. Hong, and H.-W. Park, "Contact-implicit Model Predictive Control: Controlling diverse quadruped motions without pre-planned contact modes or trajectories," *The International Journal of Robotics Research*, vol. 44, no. 3, Mar. 2025.

[13] G. Williams, A. Aldrich, and E. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, pp. 1–14, 01 2017.

[14] J. Alvarez-Padilla, J. Z. Zhang, S. Kwok, J. M. Dolan, and Z. Manchester, "Real-time whole-body control of legged robots with model-predictive path integral control," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025.

[15] G. Turrisi, V. Modugno, L. Amatucci, D. Kanoulas, and C. Semini, "On the benefits of gpu sample-based stochastic predictive controllers for legged locomotion," *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13 757–13 764, 2024.

[16] 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," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[17] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, "Predictive sampling: Real-time behaviour synthesis with mujoco," 2022.

[18] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *International Conference on Informatics in Control, Automation and Robotics*, 2004.

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

[20] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1433–1440.

[21] H. Homburger, S. Wirtensohn, M. Diehl, and J. Reuter, "Feature-based mppi control with applications to maritime systems," *Machines*, vol. 10, no. 10, 2022. [Online]. Available: https://www.mdpi.com/2075-1702/10/10/900

[22] C. Pan, Z. Yi, G. Shi, and G. Qu, "Model-based diffusion for trajectory optimization," in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37. Curran Associates, Inc., 2024.

[23] P. N. Crestaz, L. D. Matteïs, E. Chane-Sane, N. Mansard, and A. Del Prete, "TD-CD-MPPI: temporal-difference constraint-discounted model predictive path integral control," *IEEE Robotics Autom. Lett.*, vol. 11, no. 1, pp. 498–505, 2026.

[24] B. Vlahov, J. Gibson, M. Gandhi, and E. A. Theodorou, "Mppi-generic: A cuda library for stochastic trajectory optimization," 2024.

[25] C. de Boor, *A Practical Guide to Splines*, revised ed. Springer, 2001.

[26] J. Carpentier, Q. Le Lidec, and L. Montaut, "From Compliant to Rigid Contact Simulation: a Unified and Efficient Approach," in *20th edition of the "Robotics: Science and Systems" (RSS) Conference*, Delft, Netherlands, July 2024.

[27] A. Jordana, J. Zhang, J. Amigo, and L. Righetti, "An introduction to zero-order optimization techniques for robotics," 2025.

[28] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *Journal of Machine Learning Research*, vol. 11, no. 104, pp. 3137–3181, 2010.

[29] B. Brogliato, T. ten Dam, L. Paoli, F. Génot, and M. Abadie, "Numerical simulation of finite dimensional multibody nonsmooth mechanical systems," *Applied Mechanics Reviews*, vol. 55, no. 2, 2002.