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 dual-space 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 allows us to achieve real-time control on standard CPU hardware, eliminating the need for GPU acceleration typically required by other state-of-the-art MPPI methods. We validate our approach on the Go2 quadrupedal robot, demonstrating various 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.**

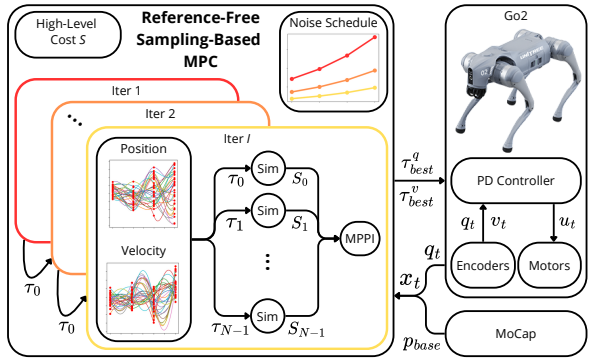
I. Introduction

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

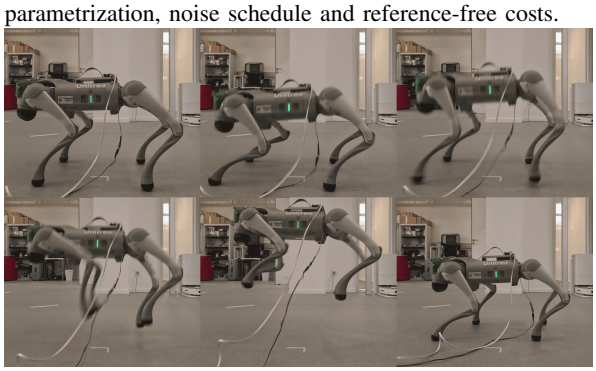
On the other side, trajectory optimization (TO) methods using gradient-based techniques can find high-quality solutions but are computationally intensive and require accurate gradient information. This may be unavailable or unreliable in contact-rich scenarios, leading most TO methods to use predefined contact sequences [7], [8]. Contact-implicit TO removes the need for predefined contact schedules [9] and has enabled advanced demonstrations of complex whole-body behaviors [10], [11]. 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.

[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



(a) Overview of the framework showing the dual-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: **Overview of the reference-free sampling-based MPC framework (top):** our approach enables emergent jumping motion experimentally achieved on the Go2 robot without any guiding reference (bottom).

These methods also necessitate hand-crafted cost functions to obtain good contact sequences [11].

Sampling-based methods offer an attractive alternative by providing derivative-free optimization that is inherently 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 a slow convergence rate, and may be unable to converge to accurate solutions. Advances in sampling-based control over the past few years, particularly Model Predictive Path Integral (MPPI) control [12], 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 con-

	DIAL-MPC [15]	RT-Whole-Body MPPI [13]	**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 fully eliminating the need for reference gait requirements.

trol 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 [13], foot swing trajectories [14], or predefined contact sequences [15], 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 dual-space spline parameterization that jointly samples position and velocity control points, improving exploration and dynamic consistency. Constraining endpoint derivatives further 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. For instance, Howell et al. [16] evaluate predictive sampling (PS) as a zero-order baseline for comparison against methods like vanilla gradient descent, the iterative Linear Quadratic Regulator (iLQR) [17] and MPPI [12]. 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 [18] simulator and required workstation-class hardware.

More informative updates arise from leveraging statistics across all sampled trajectories, as done in MPPI. This approach has shown strong performance in real-time control for racing, where parallel rollouts can be combined efficiently [19]. Turrisi et al. [14] 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 [15], which introduces a two-level annealing schedule inspired by diffusion models [20]. 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. [21] add a constraint-enforcing mechanisms and a terminal value function approximation for longer-horizon reasoning.

Recent work has also investigated CPU-based implementations. Alvarez-Padilla et al. [13] 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 [18], avoiding fixed contact sequences but 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 CPU with a limited number of rollouts, does not rely on gait priors, and employs a dual-space spline parameterization that samples both position and velocity targets for the PD controller. Unlike prior joint-space sampling methods, which provide only position references (and therefore set velocity targets to zero), our formulation yields dynamically consistent PD targets and richer exploration.

III. Methodology

We formulate our controller within the MPPI framework [12], 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 two practical enhancements: (i) a dual-space spline parameterization that jointly optimizes position and velocity control points to improve exploration and enforce dynamically consistent trajectories, (ii) reference-free cost formulations that support 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 crucial for effective random-search optimization. Unlike methods that sample in torque space, which typically require extensive parallel simulations on GPUs [22], we adopt joint-space sampling with a subsequent PD controller for impedance control, transforming joint-space references into torque references. By delegating low-level torque control to the PD controller, our method achieves stable configurations efficiently. This choice enables us to leverage efficient CPU-based simulation at high frame rates, requiring only 20-30 parallel trajectories to control a quadruped robot, and up to 60-70 parallel trajectories for a humanoid robot, compared to GPU-based

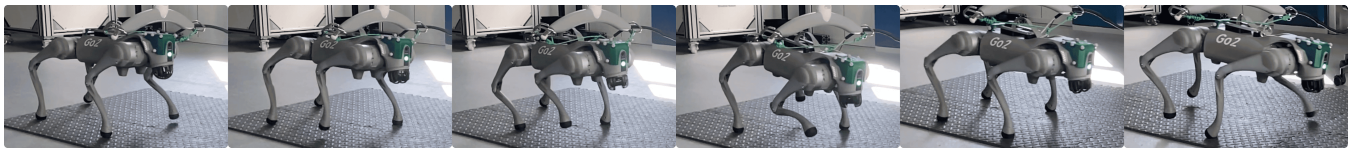


Fig. 2: Sequence illustrating the discovered walking gait on the Go2 quadruped.

simulators that offer high throughput but slower per-frame execution.

Dual-space spline parameterization. Control sequences are parameterized using cubic Hermite splines defined by K node control points per controlled degree of freedom. Each spline node consist of both a position and a velocity component,

$$\theta_k = (\theta_k^q, \theta_k^v),$$

so that the continuous joint trajectory $q(t)$ and its time derivative are reconstructed via Hermite interpolation. Denoting the set of spline basis functions by $\mathscr{H}_k^{(3)}(t)$ and their time derivatives by $\dot{\mathscr{H}}_k^{(3)}(t)$, the trajectory reads

$$q(t) = \sum_{k=0}^{K-1} \theta_k^q \, \mathscr{H}_k^{(3)}(t) + \theta_k^v \, \dot{\mathscr{H}}_k^{(3)}(t). \tag{1}$$

By explicitly controlling position (joint position q) and its first-order time derivatives (joint velocity v) at the spline control points, Hermite splines provide more flexible representations of motion and enable better coverage of the trajectory space, as illustrated in Fig. 3. To prevent spline interpolants from crossing joint limits, we limit per-node derivatives based on the distance to the nearest bound. For node value $\theta_k^q \in [q_{min}, q_{max}]$ and node spacing Δt, we impose

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

This inexpensive clamp effectively suppresses mid-interval excursions. In Fig. 3, the red dashed lines indicate the upper bounds. While quadratic and cubic position splines

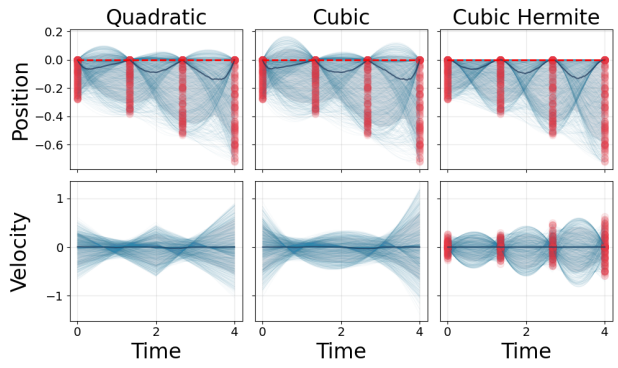


Fig. 3: **Plot comparison of different spline types** with the same interpolation points (red), resulting in different normalized position and velocity trajectories. Cubic Hermite splines exhibit a lower variance than quadratic and cubic splines, resulting in finer sampling granularity.

frequently overshoot these limits, our cubic Hermite formulation respects the bounds throughout the trajectory. It is worth noting that quadratic or cubic splines inherently limit expressiveness: if the initial velocity is low, both do not generate significant mid-trajectory velocity changes. In contrast, Hermite splines naturally incorporate velocity information at control points, exhibit lower variance than quadratic and cubic splines, and result in finer sampling granularity. This richer parameterization enables a greater diversity of behaviors to be explored during sampling-based optimization, as shown in our experiments in Section IV.

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 [15]. A nominal control sequence is refined over I internal iterations at each time step before executing the first action, and then the horizon rolls forward. 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 exploration variance 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 $i = I, \ldots, 1$:

$$\det\big(\Sigma_\theta^i\big) \propto \exp\Big(-\frac{I-i}{\beta_1 I} K d_u\Big), \tag{3}$$

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 additionally increase exploration for control points that correspond to later actions in the horizon. For spline node index $k \in \{0, \ldots, K-1\}$, this yields

$$\det\big(\Sigma_{\theta_k}^i\big) \propto \exp\Big(-\frac{K-k}{\beta_2 K} d_u\Big), \tag{4}$$

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:

$$\Sigma_{\theta_k}^i = \exp\Big(-\frac{I-i}{\beta_1 I} - \frac{K-k}{\beta_2 K}\Big)\mathbf{I}, \tag{5}$$

where $\mathbf{I} \in \mathbb{R}^{d_u \times d_u}$. The schedule in Eq. (5) can be precomputed and cached once. It provides larger variance for later 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.

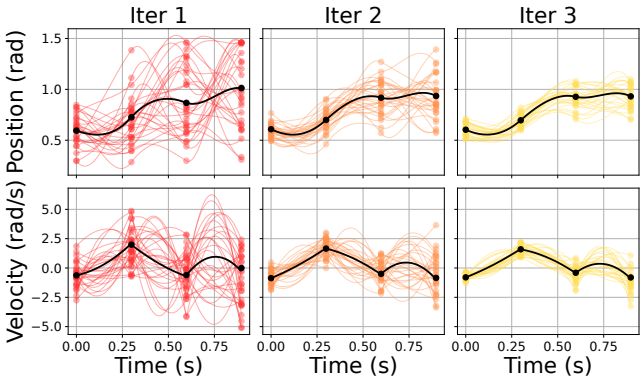


Fig. 4: The nominal trajectory (black) evolves through spline control points that are updated iteratively. New perturbed spline points are sampled around the nominal points with annealing noise according to Eq. (5).

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. (5)
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. (8)
    update nominal τ_0 using Eq. (9)
    update τ_{best} (Sec. III-D)
  end for
  compute torque u_t using Eq. (10)
end for

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, while velocity points start at 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 [23], evaluated under the task cost, and assigned importance weights. The nominal sequence is then updated by importance-weighted averaging, and new control points are extracted by resampling the updated trajectory at spline node times, distributed uniformly across the horizon.

The exponential weighting scheme considers the relative quality of all candidates, see Eq. (6), and enables effective trajectory synthesis. The softmax smoothing naturally concentrates probability mass around high-quality solutions and can be interpreted as performing approximate natural gradient descent on a smoothed surrogate function [24]. The normalized weight ω_n for each trajectory n is computed as

$$\omega_n = \frac{\exp\big(-S_n/\lambda\big)}{\sum_j \exp\big(-S_j/\lambda\big)}, \tag{6}$$

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 [25]:

$$\widehat{S}_n = \frac{S_n - S_{\min}}{S_{\max} - S_{\min}}, \quad S_{\max} = \max_j S_j, \quad S_{\min} = \min_j S_j, \tag{7}$$

and the weights are computed as

$$\omega_n = \frac{\exp\big(-\widehat{S}_n/\lambda\big)}{\sum_j \exp\big(-\widehat{S}_j/\lambda\big)}. \tag{8}$$

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

$$\tau_0^q = \sum_n \omega_n \tau_n^q, \qquad \tau_0^v = \sum_n \omega_n \tau_n^v. \tag{9}$$

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

$$u_t = K_p\big(\tau_{best}^q[0] - q_t\big) + K_d\big(\tau_{best}^v[0] - v_t\big), \tag{10}$$

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) *dual-space 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 through standard MPPI importance-weighted averaging across iterations, the robot always executes actions from

τ_{best}, which is the best-performing trajectory that has been explicitly tested through simulation rollout. This serves two purposes: (1) it ensures that executed actions always come from a verified, fully-simulated trajectory rather than from an untested weighted mixture, providing safety guarantees; and (2) it prevents performance degradation during iterative refinement by maintaining monotonic improvement in executed solution quality.

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}:

$$x_t = \text{simulate}\big(x_{t-\Delta t}, \tau_{\text{best}}[0:\lfloor \Delta t/dt \rfloor], \Delta t\big). \tag{11}$$

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,

$$\tau_{\text{best}}[h] \leftarrow \tau_{\text{best}}[h+\lfloor \Delta t/dt \rfloor], \quad h \in [0, H-\lfloor \Delta t/dt \rfloor), \tag{12}$$

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 works that handle computation delays by constraining the first $\lfloor \Delta t/dt \rfloor$ actions across all parallel rollouts to match the actions being 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 [23], which provides high-fidelity, accurate, and efficient frictional contact dynamics simulation by proper handling of nonlinear complementarity contact models [26]. 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 $c_T(x_H)$:

$$S = \sum_{t=0}^{H-1} c_t(x_t,u_t) + c_T(x_H). \tag{13}$$

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

$$\begin{aligned} c_t(x_t,u_t) = {} & w_h\,|p_{base,z}-p_{des,z}| + w_{orient}\,\|\log_3(R_{base}^T R_{des})\|_2^2 \\ & + w_q\,\|q-q_0\|_2^2 + w_{c,vel}\,\|v_c\|_1 + w_{c,force}\,\|f_c-f_0\|_1, \end{aligned} \tag{14}$$

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:

$$c_T(x_H) = w_H\,\|p_{base}(x_H) - p_{target}\|_1, \tag{15}$$

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 demonstrating reference-free motion discovery in real-time across different tasks and even robot morphologies. We systematically evaluate the quadruped's ability to discover motion strategies purely from high-level goals and demonstrate that our framework transfers to a humanoid robot, enabling the achievement of walking gaits without requiring the tuning of gait parameters or other motion priors.

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: contrary to reference-tracking methods that can use shorter horizons of 0.4 s following 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 results in falling. 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.

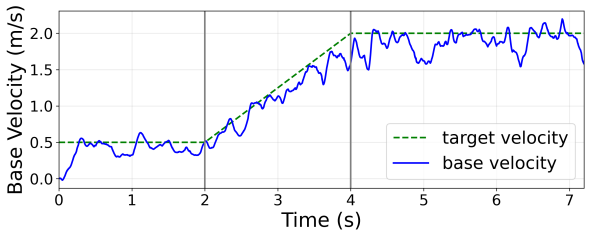
To analyze the different gaits, we plot the obtained contact patterns in Fig. 5b where the initial target velocity of $0.5\,\mathrm{m\,s^{-1}}$ was increased linearly until reaching $2.0\,\mathrm{m\,s^{-1}}$.

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.

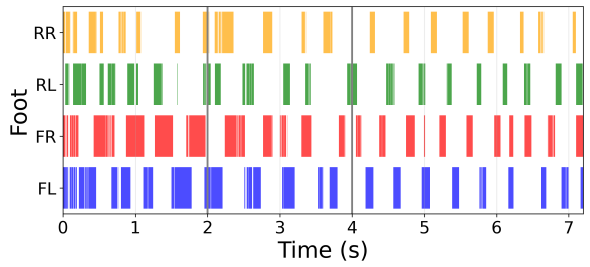
When we rotate the mattress on which the robot is standing, it responds by generating turning walking motions to restore its original yaw angle. If the mattress is pushed or pulled, the robot walks to recover its original position. In the most extreme test, we lift the robot into the air, rotate it by 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 different jumping behaviors by specifying only sparse task-level objectives where the goal is to jump vertically, turn the base, and land safely.

Vertical jumping: Commanding a target base height increase ($0.325\,\mathrm{m} \rightarrow 0.7\,\mathrm{m}$) as terminal cost via a step



(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.

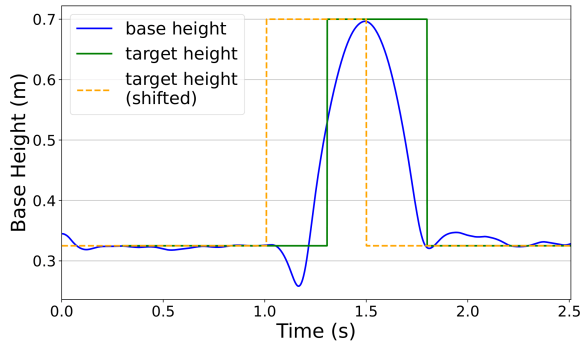


Fig. 6: Robot base height during vertical jumping. When commanded to reach 0.7 m (green), the robot discovers a jumping strategy. The orange dashed line shows when the terminal height objective enters the MPC horizon, triggering the robot to initiate its jump sequence (blue).

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 down by flexing its legs to then rapidly extends its legs to launch into the air, achieving 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, explosive take-off generating 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.

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 discovers stabilizing strategies through continuous leg repositioning and contact pattern adjustments. 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.

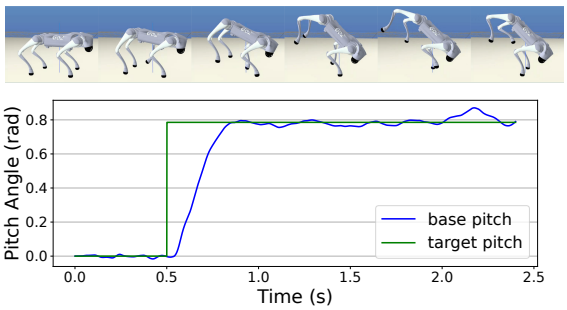
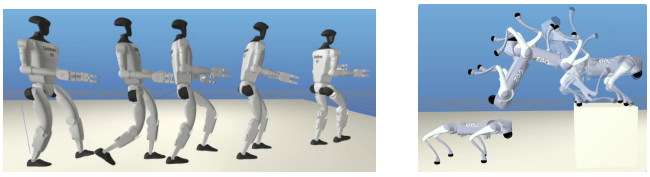


Fig. 7: Base pitch trajectory during handstand pose.



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

Fig. 8: **Emergent dynamic behaviors in simulation.**

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.

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:

(1) Dual-space Hermite spline parameterization. We compare our cubic Hermite parameterization with velocity targets against simpler alternatives: cubic splines without velocity targets and quadratic splines. The goal is to quantify the improvement in trajectory quality and stability that results from 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 find an improvement over the shifted previous solution: walking (27.9%), jumping (29.7%), and handstand (27.1%).

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

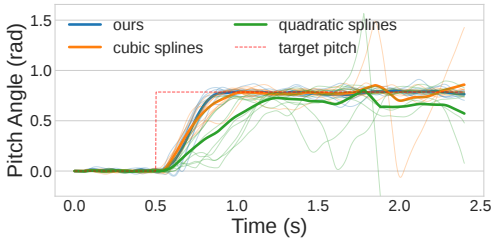
V. Discussion and Conclusion

We presented a sampling-based MPC framework that enables reference-free locomotion by combining dual-space spline parameterization with diffusion-inspired noise annealing. In contrast to prior MPPI-inspired approaches that rely on thousands of GPU rollouts and/or gait priors, our method achieves real-time performance on standard CPU hardware with as few as 30 samples. Experiments on the Go2 quadruped demonstrate real-world behaviors ranging from adaptive gaits to jumping, while simulation studies showcase more complex acrobatics such as 180-degree turns, backflips, and handstand balancing. Additionally, successful transfer to the G1 humanoid in simulation demonstrates the discovery of bipedal walking without modification. These results highlight that diverse and dynamic locomotion strategies can emerge purely from high-level objectives, without reference tracking or pre-computed contact sequences.

By reducing computational demands and removing the need for handcrafted priors, our work demonstrates that a wide variety of locomotion strategies can emerge in real time. At the same time, the motions we obtain are not always

TABLE III: **Ablation study table with figure.** The success rate (out of 10 seeds) and mean ± standard deviation cost are reported for Walking and Handstand tasks. For the Jumping task, the mean ± standard deviation of maximum reached height is reported. **Right.** Comparison of base rotation trajectories during the Handstand task.

Variant	Walking		Handstand		Jumping
	Succ.	Cost	Succ.	Cost	Max Height (cm)
Hermite (ours)	10/10	913.9 ± 25.8	10/10	811.7 ± 253.4	68.4 ± 0.2
Cubic (no vel)	6/10	1124.1 ± 135.7	10/10	1540.2 ± 655.8	63.8 ± 0.4
Quadratic (no vel)	0/10		9/10	1882.2 ± 493.9	47.7 ± 0.4
Best-trajectory (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



fully optimized in terms of smoothness or efficiency. This limitation is an expected consequence of our design choice. Sampling spline control points in joint space and delegating torque-level execution to a PD controller reduces the search space, but also limits fine-grained optimization, since we do not optimize direct torque commands at every control step.

Rather than a drawback, we view this as a tradeoff that makes the approach particularly 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 combining our method with trajectory optimization or learning-based refinement to transform diverse exploratory motions into polished, task-specific controllers.

Acknowledgments

This work has received support from the French government, managed by the National Research Agency, under the France 2030 program with the references Organic Robotics Program (PEPR O2R) and "PR[AI]RIE-PSAI" (ANR-23-IACL-0008). This research was funded, in part, by l'Agence Nationale de la Recherche (ANR), projects RODEO (ANR-24-CE23-5886) and PEPR O2R – PI3 ASSISTMOV (ANR-22-EXOD-0004). The European Union also supported this work through the ARTIFACT project (GA no.101165695) and the AGIMUS project (GA no.101070165). The Paris Île-de-France Région also supported this work in the frame of the 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] 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, pp. 465–472.

[3] Y. Zhao, L. Mou, and B. Chazelle, "Sim-to-real transfer in robotics: A review," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1481–1493, 2020.

[4] J. Hwangbo, J. Lee, and et al., "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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, pp. 486–510, Mar. 2025.

[12] 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.

[13] 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, pp. 14 721–14 727.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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, pp. 57 914–57 943.
[21] P. N. Crestaz, L. de Matteis, E. Chane-Sane, N. Mansard, and A. D. Prete, "TD-CD-MPPI: Temporal-Difference Constraint-Discounted Model Predictive Path Integral Control," Aug. 2025, working paper or preprint.
[22] B. Vlahov, J. Gibson, M. Gandhi, and E. A. Theodorou, "Mppi-generic: A cuda library for stochastic trajectory optimization," 2024.
[23] 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.
[24] A. Jordana, J. Zhang, J. Amigo, and L. Righetti, "An introduction to zero-order optimization techniques for robotics," 2025.
[25] 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.
[26] 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, pp. 107–150, 2002.