

Real-Time Whole-Body Control of Legged Robots with Model-Predictive Path Integral Control

Juan Alvarez-Padilla¹, John Z. Zhang², Sofia Kwok², John M. Dolan², and Zachary Manchester²

Abstract— This paper presents a system for enabling real-time synthesis of whole-body locomotion and manipulation policies for real-world legged robots. Motivated by recent advancements in robot simulation, we leverage the efficient parallelization capabilities of the MuJoCo simulator to achieve fast sampling over the robot state and action trajectories. Our results show surprisingly effective real-world locomotion and manipulation capabilities with a very simple control strategy. We demonstrate our approach on several hardware and simulation experiments: robust locomotion over flat and uneven terrains, climbing over a box whose height is comparable to the robot, and pushing a box to a goal position. To our knowledge, this is the first successful deployment of whole-body sampling-based MPC on real-world legged robot hardware. Experiment videos and code can be found at: whole-body-mppi.github.io

I. INTRODUCTION

Building robots that can gracefully traverse difficult terrains and skillfully manipulate objects like humans and animals has been a long-standing goal in robotics. For a long time, the topics of robot locomotion [1] and manipulation [2] have been studied separately. Recent interest in general-purpose robot agents (i.e. humanoid robots) in both industry and academia has motivated designing control and planning algorithms capable of mastering both locomotion and manipulation skills in the same embodiment [3], [4]. Despite this rise in interest, general methods capable of producing whole-body behaviors in real time on real-world quadruped and humanoid robots have so far remained elusive.

In this paper, we take advantage of the increasing performance of modern-day robotics simulation technology, in particular, the MuJoCo physics engine [5], to compute real-time control policies on legged robots using model-predictive path integral control (MPPI) [6], [7]. Contrary to the common belief that sampling approaches in high-dimensional tasks are computationally intractable, especially in real-time scenarios, we find that MPPI can be surprisingly effective at solving legged robot locomotion and manipulation tasks with a few simple design choices. To the best knowledge of the authors, this work represents the first time sample-based whole-body control has been successfully deployed on real-world legged robots.

There are several key design choices in our implementation: First, we reduce the size of the search space by

¹Juan Alvarez-Padilla is with the Department of Electrical and Computer Engineering, Carnegie Mellon University. jralvare@andrew.cmu.edu

²John Z. Zhang, Sofia Kwok, John Dolan, and Zachary Manchester are with the Robotics Institute, Carnegie Mellon University. johnzhang@cmu.edu, sofiak@andrew.cmu.edu, jdolan@andrew.cmu.edu, zacm@cmu.edu

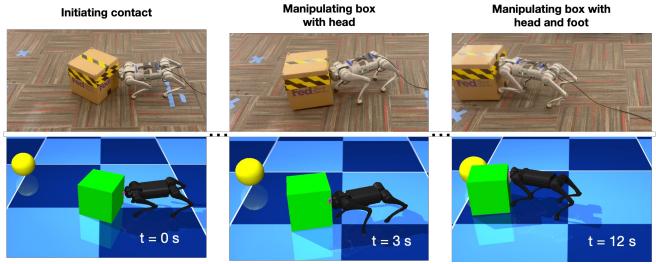


Fig. 1. A Unitree Go1 robot pushing a box to a desired location with MPPI on hardware (top row) and corresponding MuJoCo simulation states (bottom row) on a single sequence. Contact-rich behaviors like body pushes and leg kicks emerge in real-time without manual pre-specification or offline policy training.

sampling over the control points of smooth splines in the robot’s joint space and then tracking with low-level PD controllers to produce torque commands. Second, we leverage performant multi-threaded robot simulation to achieve fast real-time sampling and evaluation of simulation roll-outs. Finally, we identify key controller parameters through both empirical observations and a set of ablation studies in controlled simulation environments. Our real-world and simulation results show that a sampling-based controller enabled by a modern physics engine can effectively reason about whole-body contact-rich behaviors during locomotion and manipulation that are challenging for gradient-based MPC algorithms. Additionally, different from RL approaches that require expensive offline training, our controller reasons about such contact-rich behaviors online in real time.

Our specific contributions include:

- 1) A system for deploying sampling-based predictive controllers on legged robots in real time.
- 2) A set of ablation studies demonstrating the importance of algorithm hyperparameters that impact system performance.
- 3) A collection of hardware and simulation experiments demonstrating our system’s capabilities for solving high-dimensional, contact-rich, whole-body control problems in real time.

The remainder of this paper is organized as follows: We first review related works on legged robot control and MPPI in Section II. Next, we present our system design in Section III. Then, we describe real-world results and ablation studies in simulation in Section IV. Finally, we summarize our conclusions and point to avenues for future research in Section V.

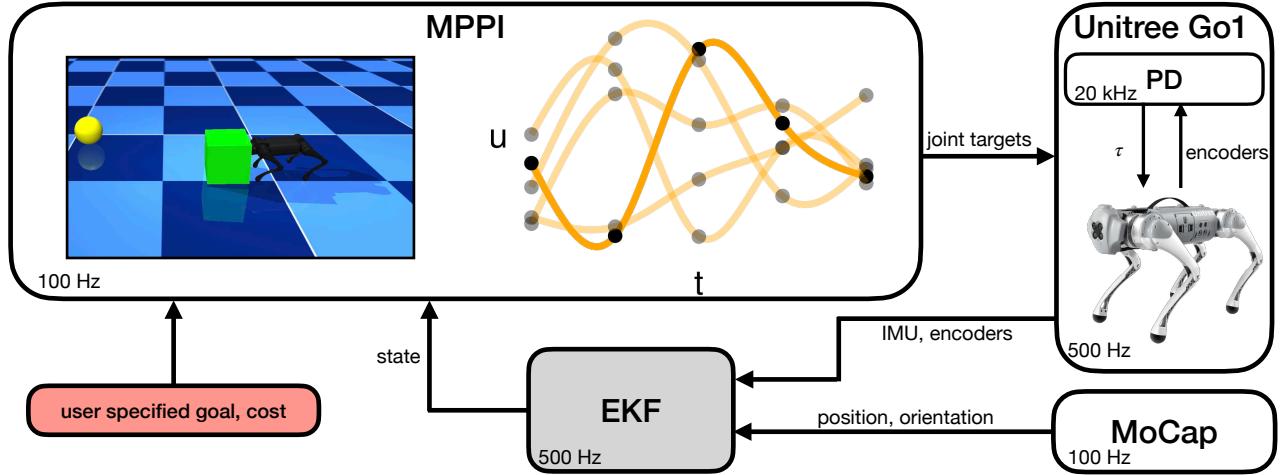


Fig. 2. System diagram for deploying the MPPI policy on a Unitree Go1 robot. Joint target controls (u) are sampled at the evenly distributed knot points (black dots) and represented as a cubic spline over the planning horizon. A cost from each sample is evaluated based on the user-specified goal (yellow ball) and cost function. The first control from the control sequence with the lowest total cost (opaque orange line) is applied to the robot and repeated in a receding-horizon fashion. The robot's state is estimated using an EKF from motion-captured position and orientation, robot onboard IMU, and joint encoder measurements.

II. BACKGROUND AND RELATED WORK

This section reviews related algorithms for legged robot locomotion and manipulation and relevant literature on MPPI.

A. Locomotion and Manipulation for Legged Robots

Current algorithms for legged robot locomotion and manipulation generally fall into two categories: gradient-based model-predictive control (MPC) [8]–[11] and gradient-free reinforcement learning (RL) [12]–[15]. MPC policies leverage first or second-order gradient information from the model to achieve real-time policy optimization without any offline computation. Despite the obvious pitfalls of relying on simplified models, MPC has been a staple for real-world deployment of legged robots and demonstrates impressive generalization to different robots and surprising robustness to model mismatch, even on challenging terrains. However, due to real-time and onboard computation requirements for these algorithms, simplified models are often employed and only contacts between the feet and the terrain are considered [11], [16], preventing these model-based policies from taking full advantage of the robot's dynamics and leveraging full-body contact to solve complicated locomotion or manipulation tasks. Existing approaches for model-based locomotion and manipulation require introducing manually designed task-specific contact pairs to the model [17], [18], significantly limiting generalization capabilities when facing new tasks.

On the other hand, simulation-based RL has shown impressive progress in recent years thanks to improved sim-to-real transfer via domain randomization techniques [12], [14] and efficient parallel simulation on modern hardware [19]–[21]. The fundamental difference between simulation-based RL and MPC is that RL attempts to learn a neural network policy *offline* through large-scale trial-and-error in simulation. In this offline policy optimization regime without real-time computation constraints, it is standard to train policies by simulating whole-body robot dynamics and collision geometries from all parts of the robot and its environment,

enabling discovery of non-trivial contact modes while solving complicated locomotion and manipulation tasks [22], [23]. In the online setting, the optimized policy network can be evaluated at real-time rates on onboard computers [14], [15]. Compared to online MPC, one downside of this offline policy optimization approach is the significant computation and time required to find performant reward functions and hyperparameters for each task.

In this paper, we aim to combine the benefits of both online MPC and simulation-based RL by solving whole-body motion-planning and control problems in real time by directly sampling over whole-body dynamics and collision models, *without* any offline policy optimization.

B. Model-Predictive Path Integral Control

Model-predictive path integral (MPPI) control is a gradient-free sampling-based algorithm often applied to real-time motion planning and control. MPPI samples N control trajectories from a multivariate Gaussian distribution $u_t \sim \mathcal{N}(\mu_t, \Sigma_t)$, where μ_t is the mean at time t and Σ_t is the covariance matrix. Each control trajectory is then simulated to compute a corresponding state trajectory, and a cost function is evaluated for each state-input trajectory sample. The control input to the system is then calculated through the exponentially weighted average of the samples based on their cost:

$$\omega_n = \frac{\exp\left(-\frac{\mathcal{L}_n - \mathcal{L}_{\min}}{\lambda}\right)}{\sum_{n=1}^N \exp\left(-\frac{\mathcal{L}_n - \mathcal{L}_{\min}}{\lambda}\right)}, \quad (1)$$

$$\mu_t = \sum_{n=1}^N \omega_n u_t, \quad (2)$$

where \mathcal{L}_n represents the cost of the n -th trajectory, ω_n denotes the corresponding trajectory weight, and λ is the temperature parameter that determines the controller's sensitivity to differences in trajectory cost. A lower value of

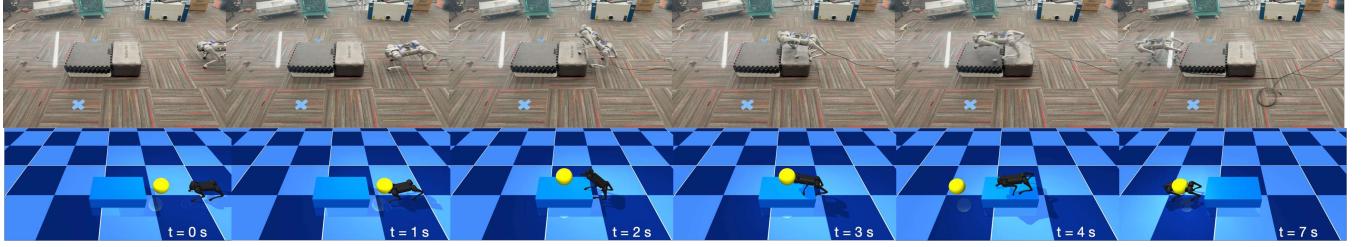


Fig. 3. Keyframes from the Unitree Go1 robot climbing up and down a box of its own height with the MPPI policy on hardware (top row) and corresponding MuJoCo states (bottom row). The robot is tasked to reach the consecutive goals (yellow spheres) specified in the task.

Algorithm 1 MPPI Control Algorithm

```

Require: Initial state  $x_0$ , control parameters  $(\mu_0, \Sigma_0)$ , number of samples  $N$ , time horizon  $T$ , temperature parameter  $\lambda$ 
Ensure: Optimal control input  $u$ 
1: Initialize control trajectory mean  $\mu_t$  and covariance  $\Sigma_t$ 
2: Initialize sample trajectory cost  $\mathcal{L}_n = 0$ 
3: for each sample  $n = 1, \dots, N$  do
4:   for each timestep  $t = 0, 1, \dots$  do
5:     Sample action sequence:  $u_t \sim \mathcal{N}(\mu_t, \Sigma_t)$ 
6:     Simulate forward for  $T$  timesteps using  $u_n$ 
7:     Compute the resulting cost and add it to  $\mathcal{L}_n$ 
8:   end for
9: end for
10: Compute weights  $\omega_n$  for each sample using Equation 1
11: Update control trajectory mean  $\mu_t$  using Equation 2
12: Select the control input for the system:  $u = \mu_t[0]$ 
13: Apply control input  $u$ 
14: Shift control mean:  $\mu_t \leftarrow \text{shifted}(\mu_t)$ 
```

λ increases the influence of the best-performing trajectory, while a higher value distributes the weight more uniformly across all samples [6]. Finally, the process is repeated in a receding horizon approach. MPPI is summarized in Algorithm 1.

Driven by significant advancements in parallel computation on modern GPUs and its downstream effects on massively parallel simulation, MPPI has seen growing popularity and success in recent years [7], [24]. In off-road autonomous driving [7], [24] domains where environment dynamics are difficult to model analytically, MPPI naturally incorporates learned black-box models from real-world data thanks to its derivative-free nature, which is difficult for gradient-based MPC. However, so far, MPPI is mainly successfully deployed in low-dimensional control tasks (e.g., driving and drone flight) due to the curse of dimensionality for sampling-based algorithms.

Recent work [25] proposes a simple yet effective modification better scale MPPI to higher-dimensional tasks (e.g., quadruped locomotion or dexterous manipulation): sample over the control points of a polynomial spline and calculate the remaining controls via spline interpolation. While this approach yielded impressive behaviors in simulation [25], closing the sim-to-real gap in real-world systems remains an open problem. Alternatively, [26] deploys MPPI on a

quadruped robot by learning a sampling distribution offline but only considers a kinodynamic robot model with foot contacts. Our work considers the whole-body dynamics and collision model, enabling automatic planning over contact strategies while solving locomotion and manipulation tasks without offline precomputation.

Rather than innovating over the MPPI algorithm itself, our work instead focuses on the system-level considerations and integration necessary to deploy this sampling-based control strategy on real-world legged robots that are high-dimensional, highly agile, and must reason about making and breaking contact with the environment.

III. SAMPLING-BASED MPC FOR LEGGED ROBOTS

This section discusses key implementation details and design considerations for deploying sampling-based MPC on real-world legged robot hardware. A diagram of the overall system is illustrated in Fig. 2.

A. MuJoCo Physics Engine

Robotics simulators are increasingly accessible and performant. In particular, Drake [27] and Dojo [28] focus on physical accuracy for algorithm verification and model-based control but pay the price of high computation costs and cannot generate simulation data at scale. Isaac [20] and MuJoCo [5], [21] provide mature parallel implementations and, as a result, have become popular choices for data-hungry algorithms like RL. For sampling-based MPC, the simulator of choice must generate large numbers of simulation samples in parallel while providing fast enough individual rollouts to close the real-time feedback control loop. With this consideration in mind, we use the MuJoCo physics engine and parallelize the rollouts on a multi-core CPU. While Isaac and MJX [21] (MuJoCo on GPU or TPU) are capable of simulating hundreds to thousands of parallel rollouts compared to dozens for the CPU-based MuJoCo, the individual rollout speeds are insufficient to close the real-time feedback loop. In practice, we find that 30-50 parallel rollouts in MuJoCo on a high-end CPU with strong per-core performance (e.g. recent Intel Core-i9) are sufficient to solve the locomotion and manipulation tasks considered in this paper.

B. Representing Controls as a Cubic Spline

Directly sampling in the robot joint space can be challenging for MPPI, especially when planning over medium to long horizons. Following [25], we reduce the size of

the search space by sampling over spline control points and interpolating using a cubic spline (Fig. 2). This spline representation also provides the added benefit of smoothing the controls. While it is possible to use zeroth-order or linear interpolation, we find that cubic splines perform the best in practice and focus on this representation in this paper. Similarly, in many applications, one can further reduce the search space by sampling over a reduced action space. In Sec. IV-E, we investigate the impact of sampling representation on control performance.

C. State Estimation

We estimate the full state of the robot (global position, attitude, joint angles, body angular velocity, and joint velocities) by fusing position and attitude measurements from a motion-capture system (100 Hz), onboard IMU (500 Hz), and onboard joint encoders (500 Hz) with an EKF running at 500 Hz, Fig. 2. Specifically, we take body velocity estimation from the single-rigid body EKF [11] and fuse it with the motion-capture measurements. Different from standard locomotion policies [11], [15] focusing on body-velocity estimation and tracking, precise global states are key to correctly resetting the MuJoCo simulation. Admittedly, a mismatched model between planning (whole-body, soft contact) and estimation (single-rigid body, hard contact) is not ideal and can degrade real-world performance. We find that this mismatch can be minimized with estimator tuning (especially the height of the robot, as it directly relates to planned contact forces during locomotion) and plan to further investigate estimation algorithms using MuJoCo dynamics in the future.

IV. EXPERIMENTS AND RESULTS

In this section, we present results demonstrating the capabilities of our sampling-based MPC implementation on a variety of tasks and describe the implementation details of the hardware system. In particular, we highlight the ability of our system to generate emergent whole-body contacts in real time while solving challenging locomotion and manipulation tasks without any contact pre-specification or offline policy optimization. Experiment videos and code can be found on our project website:

whole-body-mppi.github.io

A. Hardware Implementation

We run MPPI on a workstation computer equipped with an Intel i9-12900KS CPU and 64 GB of memory. This workstation is connected to the robot via Ethernet and communicates using ROS. All hardware experiments are run on a Unitree Go1 quadruped robot. Across all hard experiments, we update the MPPI policy at 100 Hz, discretize the dynamics at 0.01 s, plan over a 0.4 s horizon ($T = 40$ timesteps), and roll out 30 samples. For contact simulation in MuJoCo, we use the default parameters for the contact model and rely on the linearized friction cone for speed. We refer the interested reader to our website for a full list of hyperparameters for each task. We publish joint targets from

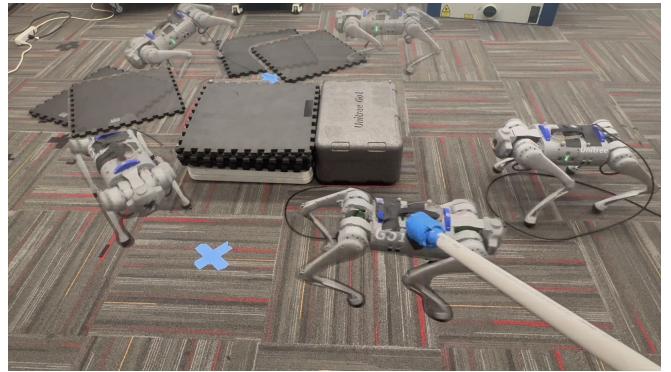


Fig. 4. Go1 robot walking in a clockwise hexagon trajectory under small to moderate model mismatch and external disturbance. More transparent robots represent earlier keyframes.

MPPI to the robot that are tracked using the Go1’s onboard low-level PD controller at 20 kHz.

B. Walking on Flat Terrain

We first verify our implementation on a flat terrain locomotion task. The walking task cost function tracks a state and control reference:

$$\mathcal{L}_{\text{walk}} = \sum_{t=0}^T \left[(x_{\text{ref}} - x_t)^T Q (x_{\text{ref}} - x_t) + (u_{\text{ref}}(t) - u_t)^T R (u_{\text{ref}}(t) - u_t) \right], \quad (3)$$

where x_t and u_t are the state and control of the robot at index t , respectively. x_{ref} includes the desired robot position and attitude. $u_{\text{ref}}(t)$ represents the joint targets of a walking gait based on the Raibert heuristic [29]. Although we observe various locomotion gaits emerge by setting a time-invariant u_{ref} to the standing pose and a goal position for the robot, we find the walking policy is significantly more robust when tracking a gait reference. Finally, Q and R are diagonal weight matrices on states and controls.

We successfully deploy the MPPI walking policy by trotting in place and walking to various user-specified waypoints (Fig. 4). Our policy is robust to moderate external disturbances and unmodeled terrain mismatch.

C. Locomotion Over Challenging Terrains

In more challenging locomotion scenarios, we model the terrain geometry in MuJoCo and set waypoints for the robot to track. We successfully deploy the MPPI policy on hardware to climb up a box of up to 0.24 m tall — roughly the height of the robot when standing (Fig. 3). Although we use the same trotting gait reference as when walking on flat ground, we observe the robot leverage unplanned motions such as jumps and contacts with the body to traverse the tall obstacle.

D. Box Pushing

Next, we showcase the whole-body contact planning capabilities of our MPPI policy to push a 3.5 kg box of size $0.36m \times 0.36m \times 0.36m$, Fig. 1. In addition to the locomotion cost on the robot, we add an ℓ_1 -norm cost for the position

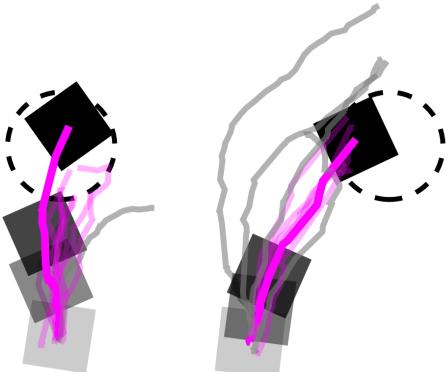


Fig. 5. Top-down view of real-world box trajectories (magenta and grey lines) from 10 trials of the Go1 robot pushing the box (black square) into the goal area (dashed circle) that is placed in front (left) and front-right (right) of the original box position. More transparent boxes represent earlier in the trajectory. Magenta lines represent runs where the box successfully reaches the target area while the grey lines indicate otherwise.

of the box: $\mathcal{L}_{\text{box}} = Q_{\text{box}} \|\bar{x}_{\text{box}} - x_{\text{box}}\|_1$, where \bar{x}_{box} and x_{box} are the target and current positions of the box. The total cost function for this task is:

$$\mathcal{L}_{\text{box push}} = \mathcal{L}_{\text{walk}} + \mathcal{L}_{\text{box}} \quad (4)$$

Note that while the box orientation feedback is important for planning, we do not penalize the orientation for this task. As a simple heuristic to encourage robot-box interaction, we place the goal for the robot at the center of the box but do not specify when or how the robot should interact with the object in any way.

We set the box 1m in front of the quadruped and ask the robot to manipulate the box to two different goal locations: 1) directly 1m forward and 2) 1m forward and 0.75m to the right from the original box position, Fig. 1. We define a trial as a success and stop the run if the position of the box is within 0.3m from the goal at any time. In the first scenario, we complete the task 9 out of 10 times. In the second, more challenging, scenario, we complete the task 6 out of 10 times. In this scenario, the robot must horizontally manipulate the box, requiring more sophisticated interactions between the robot body, leading to a larger sim-to-real gap. Box trajectories can be found in Fig. 5. Experiment videos can be found on our website and in the supplementary video.

Interestingly, from online sampling in the robot action space, we observe emergent contact interactions between the robot and the box, such as body or shoulder pushes and leg kicks to move the box to the goal position.

E. Ablation Studies

We investigate the effects of key MPPI hyperparameters on overall task performance in a controlled simulation setting. Note that it is nearly impossible to discuss a single hyperparameter’s impact on overall system performance without the task or other parameters. Our goal is to provide an intuitive understanding of the tradeoffs when making design decisions. For all experiments, we fix the remaining hyperparameter to the default values presented above. In controlled MuJoCo and Gazebo simulation environments, we

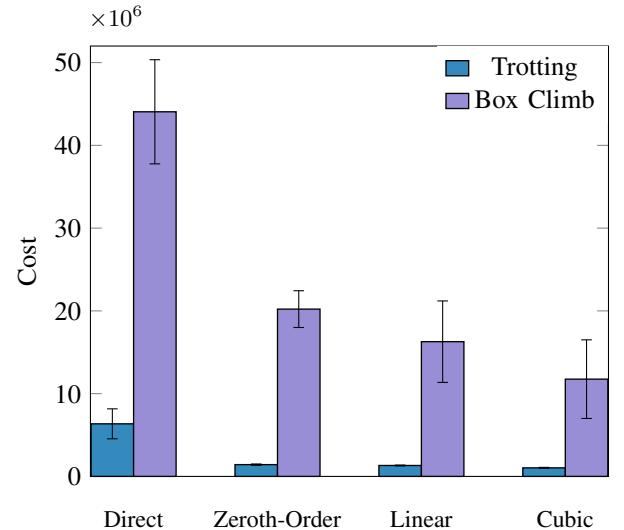


Fig. 6. Policy rollout costs (lower is better) with different sampling representation for MPPI on trotting forward (blue) and box-climb (purple) tasks in MuJoCo. Directly sampling controls over the prediction horizon (Direct) performs significantly worse than sampling spline control points and interpolating. Cubic interpolation (ours) outperforms zeroth-order and linear interpolation in both tasks.

slow down simulation to match the policy update frequency to avoid real-time computation-related issues.

1) *Sampling representation*: We first look into the choice of sampling representation in MPPI on two tasks: walking forward and climbing up a box in MuJoCo (Fig. 6). Unsurprisingly, directly sampling robot joint targets over the entire prediction horizon is unable to sufficiently explore the optimization landscape given fixed samples compared to compressed sampling representations. Among interpolation schemes, higher-order polynomials generally produce better results, especially on a challenging task like climbing up a box. Given that we choose to sample four points along each trajectory, a cubic polynomial is the highest order possible. We leave higher-order interpolation in the presence of added knot points for future work.

The remaining ablations are performed on the walking task alone in both MuJoCo and sim-to-sim transfer to Gazebo:

2) *Control frequency*: Similar to other MPC algorithms, the controller update frequency is critical for deploying MPPI on legged robots. While the conventional wisdom of “faster is better” still applies in this context, we find that performance plateaus after ~ 100 Hz in Gazebo and ~ 150 Hz in MuJoCo (Fig. 7). We believe this is due to our choice of sampling over joint targets and relying on low-level PD controllers at much higher frequencies to compute torques. In practice, we find ~ 100 Hz to be sufficient on hardware.

3) *Temperature*: Temperature λ is a key parameter in the MPPI algorithm. Lower λ means added weight on the best-performing predicted rollout during the exploration, and higher λ means otherwise. Similar to our experience on hardware, we find in our ablation that $\lambda = \sim 0.1$ produces the best behaviors, Fig. 7.

4) *Prediction horizon*: Planning or prediction horizon plays a key role in MPC policies, especially in the absence

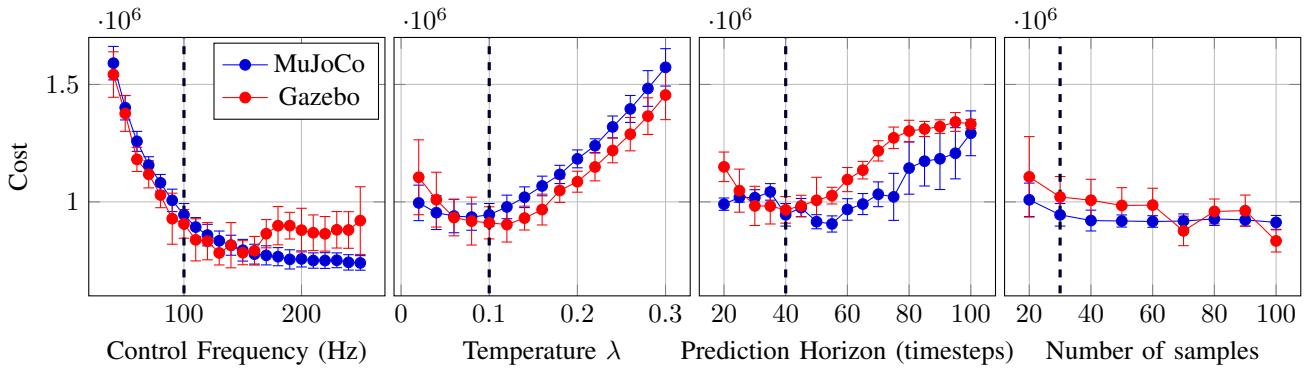


Fig. 7. Sim-to-sim policy rollout costs while varying key MPPI hyperparameters in both MuJoCo (blue) and Gazebo (red) simulation environments. The means and standard deviations are computed from 10 different random seeds for each setting. Black dotted lines denote the default parameters we deploy on hardware.

of a good estimate of the cost-to-go. For standard MPC algorithms, longer prediction horizons are generally preferred as they often replace the need for an accurate cost-to-go estimate. Surprisingly, we find that the MPPI policy performs best with a 40 – 50 timestep horizon (Fig 7). We hypothesize that the fixed number of samples prevents our MPPI policy from sufficiently exploring the longer-horizon planning problem, leading to poor performance.

5) Number samples: Generally, more samples are preferred in sampling-based optimization algorithms, usually at the expense of more computation. To our surprise, the cost plateaus at ~ 40 samples for the quadruped walking task, even if more time is allowed to compute the policy in simulation. Our practical experience with the real robot also indicates that limited computing is much better spent on achieving a ~ 100 Hz policy than additional sample evaluations during agile locomotion.

	Offline Training	Whole-Body Contact	Gradient-Free
MPC	No	No	No
RL	Yes	Yes	Yes
MPPI (ours)	No	Yes	Yes

TABLE I

COMPARISON OF FEATURES ACROSS MPC [8]–[11], [17], [18], RL [12]–[15], [19], [22], [23], AND MPPI

F. Comparisons to RL and MPC

We qualitatively compare our MPPI policy against current MPC and RL algorithms for legged robots in Table I. MPC policies generally do not require offline policy training as they leverage model gradients for real-time policy optimization. However, it is computationally intractable to include legged robot whole-body dynamics and collision models during real-time MPC policy evaluation. Simulation-based RL offers a gradient-free alternative that optimizes a neural network policy offline by collecting data using full-body dynamics and contact information in simulation. The MPPI policy (ours) is, so far, the only class of policy that can achieve policy optimization with whole-body dynamics and contact models without any offline training and can be fast enough to run directly in real time on legged robot hardware.

V. CONCLUSIONS AND FUTURE WORK

We present the first deployment of whole-body sampling-based MPC on real-world legged robots. By leveraging a performant and easily parallelizable modern robotics simulation engine, we can generate real-time contact-rich, whole-body motion plans that were previously only possible through manual pre-specification or offline policy training.

A. Limitations

Sampling-based MPC, and MPC in general, are fundamentally myopic, which means they can not see and therefore reason about interactions beyond the prediction horizon. Complementing sampling-based MPC with a global planner can enable solution of complicated long-horizon tasks. Additionally, these methods can only plan for what the simulator can simulate. Many important physical properties such as fluids [30], soft materials [31], or complicated contact and friction interactions [32] between objects cannot be accurately or efficiently simulated by common robotics physics engines.

B. Future Work

Several important future research directions remain: First, we believe closing the sim-to-real and real-to-sim loop by estimating contact and friction parameters online can significantly improve real-world policy performance. Second, improving upon our baseline MPPI controller by sampling from more sophisticated distributions or over different model parameters can enable better performance in real-world settings. Finally, we plan to integrate the robot hardware system with the MJPC interactive software [25]. This can empower users to change task parameters and cost terms and observe changes to robot behavior in real time, enabling much more efficient and intuitive task design and cost tuning for model-based controllers.

ACKNOWLEDGMENTS

The authors thank Taylor Howell for assistance with MuJoCo and Mitchell Fogelson, Swaminathan Gurumurthy, and Jon Arrizabalaga for valuable feedback and discussions on experiment design.

REFERENCES

- [1] A. Ruina, S. Collins, R. Tedrake, and M. Wisse, “Efficient bipedal robots based on passive-dynamic walkers,” *Science*, vol. 307, no. 5712, pp. 1082–1085, 2005.
- [2] M. T. Mason and J. K. Salisbury, *Robot Hands and the Mechanics of Manipulation*. Cambridge, MA: MIT Press, May 1985.
- [3] T. He, Z. Luo, X. He, W. Xiao, C. Zhang, W. Zhang, K. Kitani, C. Liu, and G. Shi, “Omnih2o: Universal and dexterous human-to-humanoid whole-body teleoperation and learning,” *arXiv preprint arXiv:2406.08858*, 2024.
- [4] NVIDIA, “Foundation model set to supercharge nvidia isaac robotics platform,” <https://nvidianews.nvidia.com/news/foundation-model-isaac-robotics-platform>, 2024, accessed: 2024-09-12.
- [5] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.
- [6] G. Williams, A. Aldrich, and E. A. Theodorou, “Model predictive path integral control: From theory to parallel computation,” *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.
- [7] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, “Information theoretic mpc for model-based reinforcement learning,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1714–1721.
- [8] A. L. Bishop, J. Z. Zhang, S. Gurumurthy, K. Tracy, and Z. Manchester, “Relu-qp: A gpu-accelerated quadratic programming solver for model-predictive control,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 13 285–13 292.
- [9] S. Le Cleac'h, T. A. Howell, S. Yang, C.-Y. Lee, J. Zhang, A. Bishop, M. Schwager, and Z. Manchester, “Fast contact-implicit model predictive control,” *IEEE Transactions on Robotics*, vol. 40, pp. 1617–1629, 2024.
- [10] S. Kuindersma, F. Permenter, M. Fallon, A. Valenzuela, H. Dai, R. Deits, T. Koolen, P. Marion, and R. Tedrake, “Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot,” in *Autonomous Robots*, vol. 40, no. 3. Springer, 2016, pp. 429–455.
- [11] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, “Mit cheetah 3: Design and control of a robust, dynamic quadruped robot,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2245–2252.
- [12] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, “Learning quadrupedal locomotion over challenging terrain,” *Science Robotics*, vol. 5, no. 47, 2020.
- [13] J. Hwangbo, J. Lee, A. Dosovitskiy, D. C. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, “Learning agile and dynamic motor skills for legged robots,” in *Science Robotics*, vol. 4, no. 26, 2019.
- [14] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, “Extreme parkour with legged robots,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 11 443–11 450.
- [15] G. B. Margolis and P. Agrawal, “Walk these ways: Tuning robot control for generalization with multiplicity of behavior,” in *Proceedings of The 6th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, K. Liu, D. Kulic, and J. Ichnowski, Eds., vol. 205. PMLR, 14–18 Dec 2023, pp. 22–31. [Online]. Available: <https://proceedings.mlr.press/v205/margolis23a.html>
- [16] J. Z. Zhang, S. Yang, G. Yang, A. L. Bishop, S. Gurumurthy, D. Ramanan, and Z. Manchester, “Slomo: A general system for legged robot motion imitation from casual videos,” *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7154–7161, 2023.
- [17] A. Rigo, Y. Chen, S. K. Gupta, and Q. Nguyen, “Contact optimization for non-prehensile loco-manipulation via hierarchical model predictive control,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 9945–9951.
- [18] F. De Vincenti and S. Coros, “Centralized model predictive control for collaborative loco-manipulation,” in *Proceedings of Robotics: Science and Systems (RSS)*, Daegu, Republic of Korea, July 10–14 2023.
- [19] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, “Learning to walk in minutes using massively parallel deep reinforcement learning,” in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 08–11 Nov 2022, pp. 91–100. [Online]. Available: <https://proceedings.mlr.press/v164/rudin22a.html>
- [20] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, “Isaac gym: High performance gpu-based physics simulation for robot learning,” 2021.
- [21] DeepMind, “Mujoco mjx documentation,” <https://mujoco.readthedocs.io/en/stable/mjx.html>, 2023, accessed: 2024-09-12.
- [22] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, “Anymal parkour: Learning agile navigation for quadrupedal robots,” *Science Robotics*, vol. 9, no. 88, p. eadi7566, 2024.
- [23] Y. Ji, G. B. Margolis, and P. Agrawal, “Dribblebot: Dynamic legged manipulation in the wild,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 5155–5162.
- [24] 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.
- [25] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, “Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo,” *arXiv preprint arXiv:2212.00541*, 2022.
- [26] J. Carius, R. Ranftl, F. Farshidian, and M. Hutter, “Constrained stochastic optimal control with learned importance sampling: A path integral approach,” *The International Journal of Robotics Research*, vol. 41, no. 2, pp. 189–209, 2022.
- [27] R. Tedrake and the Drake Development Team, “Drake: Model-based design and verification for robotics,” 2019. [Online]. Available: <https://drake.mit.edu>
- [28] T. Howell, S. Le Cleac'h, J. Bruedigam, Z. Kolter, M. Schwager, and Z. Manchester, “Dojo: A differentiable physics engine for robotics,” *arXiv preprint arXiv:2203.00806*, 2022. [Online]. Available: <https://arxiv.org/abs/2203.00806>
- [29] M. H. Raibert and H. B. Brown, “Dynamically stable legged locomotion,” 1983.
- [30] J. H. Lee, M. Y. Michelis, R. Katzschmann, and Z. Manchester, “Aquarium: A fully differentiable fluid-structure interaction solver for robotics applications,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 272–11 279.
- [31] M. Li, Z. Ferguson, T. Schneider, T. Langlois, D. Zorin, D. Panozzo, C. Jiang, and D. M. Kaufman, “Incremental potential contact: Intersection- and inversion-free, large-deformation dynamics,” *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, 2020.
- [32] R. Elandt, E. Drumwright, M. Sherman, and A. Ruina, “A pressure field model for fast, robust approximation of net contact force and moment between nominally rigid objects,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 8238–8245.