Sumo: Dynamic and Generalizable Whole-Body Loco-Manipulation

Author Names Omitted for Anonymous Review. Paper-ID [239]

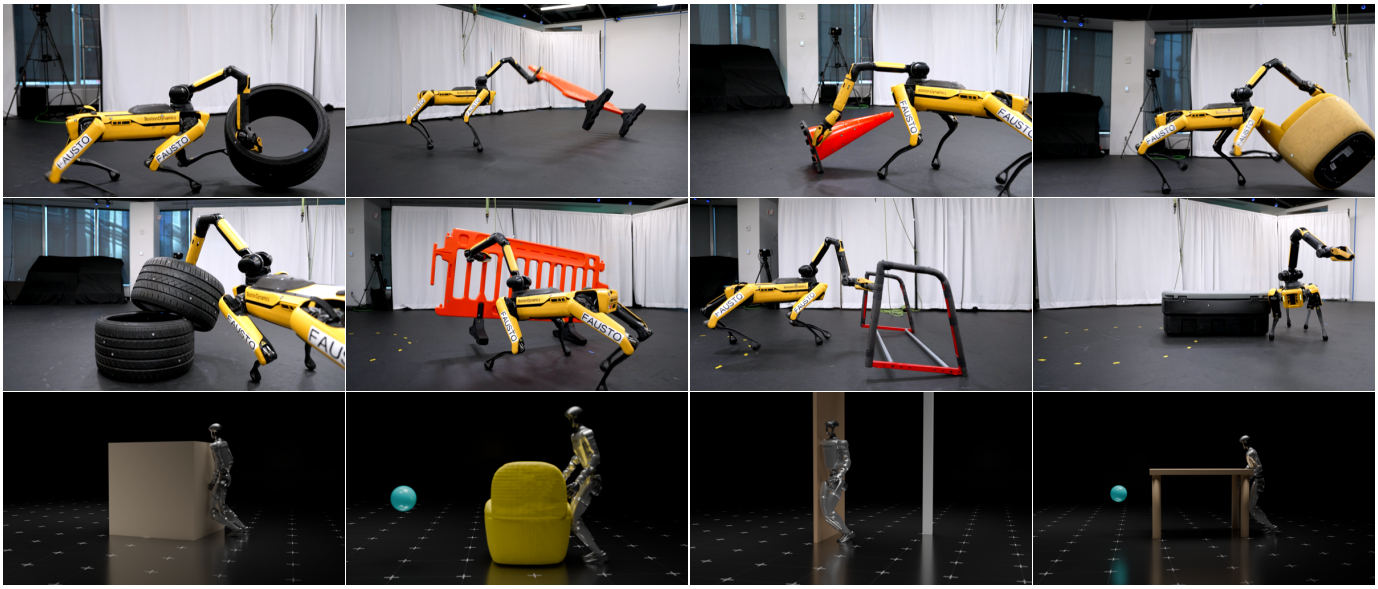

Fig. 1: First row from left to right: Spot robot uprights a tire, crowd-control barrier, traffic cone, and chair. Second row from left to right: Spot robot stacks a tire on top of another tire, drags a crowd-control barrier, a tire rack, and large box to a target position. Third row from left to right: simulated G1 humanoid robot pushes a box, chair, door, and a dining table.

***Abstract*—This paper presents a sim-to-real approach that enables legged robots to dynamically manipulate large and heavy objects with whole-body dexterity. Our key insight is that by performing test-time steering of a pre-trained whole-body control policy with a sample-based planner, we can enable these robots to solve a variety of dynamic loco-manipulation tasks. Interestingly, we find our method generalizes to a diverse set of objects and tasks with no additional tuning or training, and can be further enhanced by flexibly adjusting the cost function at test time. We demonstrate the capabilities of our approach through a variety of challenging loco-manipulation tasks on a Spot quadruped robot in the real world, including uprighting a tire heavier than the robot's nominal lifting capacity and dragging a crowd-control barrier larger and taller than the robot itself. Additionally, we show that the same approach can be generalized to humanoid loco-manipulation tasks, such as opening a door and pushing a table, in simulation.**

I. Introduction

Humans and animals physically interact with the world in creative ways: Their ability to gracefully make and break contact with their environment enables them to traverse treacherous terrain and move objects several times larger than themselves. Achieving similar levels of athletic intelligence for robotic systems has been a long-standing challenge. Over the past decade, developments in numerical optimization [9, 11, 4] and machine learning [14, 7] have made significant progress towards enabling robots to overcome challenging terrain with agility in the wild and manipulate objects with dexterity [32, 27]. The next research frontier lies in enabling robots to manipulate objects during locomotion, or so-called loco-manipulation, in the open world. So far, research on loco-manipulation has largely focused on learning from human demonstrations through teleoperation or video imitation, which are usually limited to quasi static table-top settings and fails to leverage the passive dynamics of the object or the robot. Algorithms and systems that enable a robot to autonomously coordinate its entire body to dynamically move large and heavy objects during loco-manipulation have, so far, been elusive.

This paper presents an approach for solving challenging loco-manipulation problems where dynamic whole-body coordination is required due to the size and weight of the objects. Recent works have shown extensive in-the-wild locomotion agility enabled by reinforcement learning. However, RL training takes significant compute time, requires many iterations of reward engineering, and can be difficult to adapt to out-of-distribution scenarios. On the other hand, sample-based model-predictive control (MPC) has proven to be a simple and training-free method for contact-rich manipulation [13, 1, 19, 18]. Yet, while being flexible and training free, sample-based MPC struggles to find good solutions for high-degree-of-freedom robots or dynamically unstable tasks, which frequently occur for quadruped and humanoid robots.

We find that RL and sampled-based control can perfectly

complement each other. In particular, we use RL to train a whole-body control policy offline and use real-time sample-based MPC to steer this policy for contact-rich manipulation of previously unseen objects. Our key insight is that, while sample-based control and RL solve a similar policy-optimization problem, this two-stage optimization framework addresses the shortcomings of both end-to-end RL and sample-based MPC while maintaining their key advantages: First, RL is incredibly powerful when real-world distributions can be well modeled at training time with randomization. As a result, we take a pre-trained whole-body control policy trained with sufficient randomization over physical parameters and disturbances, allowing us to deploy the policy robustly in the real-world [43]. Second, the whole-body control policy effectively reduces the action space for sample-based MPC during online planning, *and* acts as a stabilizing policy that mitigates divergence issues associated with single-shooting rollouts of unstable dynamics, significantly improving the online sample efficiency.

Because the RL policy is trained for general whole-body control with sufficient domain randomization and perturbation, the high-level sample-based controller can effectively steer this policy for real-world manipulation. The two-stage setup allows us to freeze the whole-body control policy and quickly adapt models, rewards, and optimization hyperparameters for sample-based MPC at test time when designing new tasks, enabling us to accomplish a wide variety of tasks in the real world. Additionally, we find that our approach generalizes to new tasks well in many cases by simply swapping the object model from the base policy or adding an additional term to the MPC cost function. Furthermore, leveraging a low-level policy for whole-body control allows us to design much simpler cost functions for the sample-based planner that only need to encode the manipulation objectives. The result is a robust yet flexible framework that enables quadruped and humanoid robots to use their entire body to manipulate objects with complex geometries or with sizes and weights comparable to the robots themselves, Fig. 1.

Our contributions include:

- A hierarchical framework that combines a pre-trained whole-body control policy and test-time sampling for dynamic whole-body loco-manipulation.
- A set of comparisons against state-of-the-art baselines highlighting the importance of the hierarchical approach and strong generalization capabilities from test-time search.
- A set of eight real-world demonstrations on the Spot quadruped robot and four demonstrations on the G1 humanoid robot in simulation covering diverse and challenging loco-manipulation scenarios.
- An open-source benchmark and dataset focusing on loco-manipulation tasks with objects that require whole-body coordination due to their size and weight relative to the robots' physical limits.

The remainder of this paper is organized as follows: In Sec. II, we discuss the previous success of RL and sample-based MPC, as well as their pitfalls. Additionally, we review recent approaches to loco-manipulation. In Sec. III, we detail the Sumo framework and key design decisions. In Sec. IV, we analyze the performance, test-time flexibility, and sample efficiency of Sumo against baseline algorithms in controlled simulation. In Sec. V, we further demonstrate the capabilities of our framework on a diverse set of challenging loco-manipulation tasks on a real-world Spot quadruped robot and a simulation G1 humanoid robot. Finally, we conclude in Sec. VI by addressing the limitations of the current system and propose directions for future research.

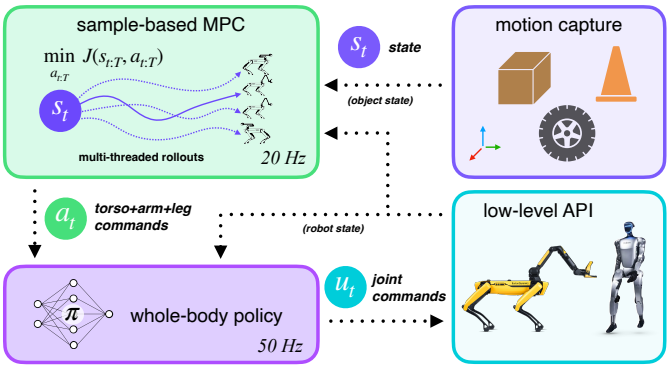

Fig. 2: System overview: Our method takes a hierarchical approach that combines a pre-trained whole-body control (WBC) policy (purple) with high-level sample-based MPC (green). The low-level whole-body control policy takes in the current state and desired torso, arm, and leg commands and outputs the joint-level commands for the quadruped or humanoid robot at 50Hz. The high-level sample-based MPC aims to minimize a task-specific cost function by taking in the current state estimate and solving for the desired torso, arm, and leg commands for the low-level policy at 20Hz.

II. Background and Related Works

This section reviews relevant literature in RL, sample-based MPC, and their recent applications to loco-manipulation.

A. Reinforcement Learning for Whole-Body Control

Deep reinforcement learning aims to learn neural-network policies via self-play trial and error. In robotics, the advancements in large-scale parallel simulation platforms [23, 10] on accelerated hardware (such as GPUs) have significantly reduced the cost of collecting data in simulation. As a result, on-policy RL algorithms, such as PPO [29], have become a staple in the sim-to-real policy optimization paradigm. Additionally, RL naturally incorporates robust policy optimization via domain randomization techniques [33], reducing the so-called sim-to-real gap.

Despite its promise, this RL paradigm faces several significant drawbacks in practice: First, RL typically requires dense reward design to find efficient policies with finite compute [25, 2]. Second, the final policy is often sensitive to reward design, and finding a suitable reward term itself is

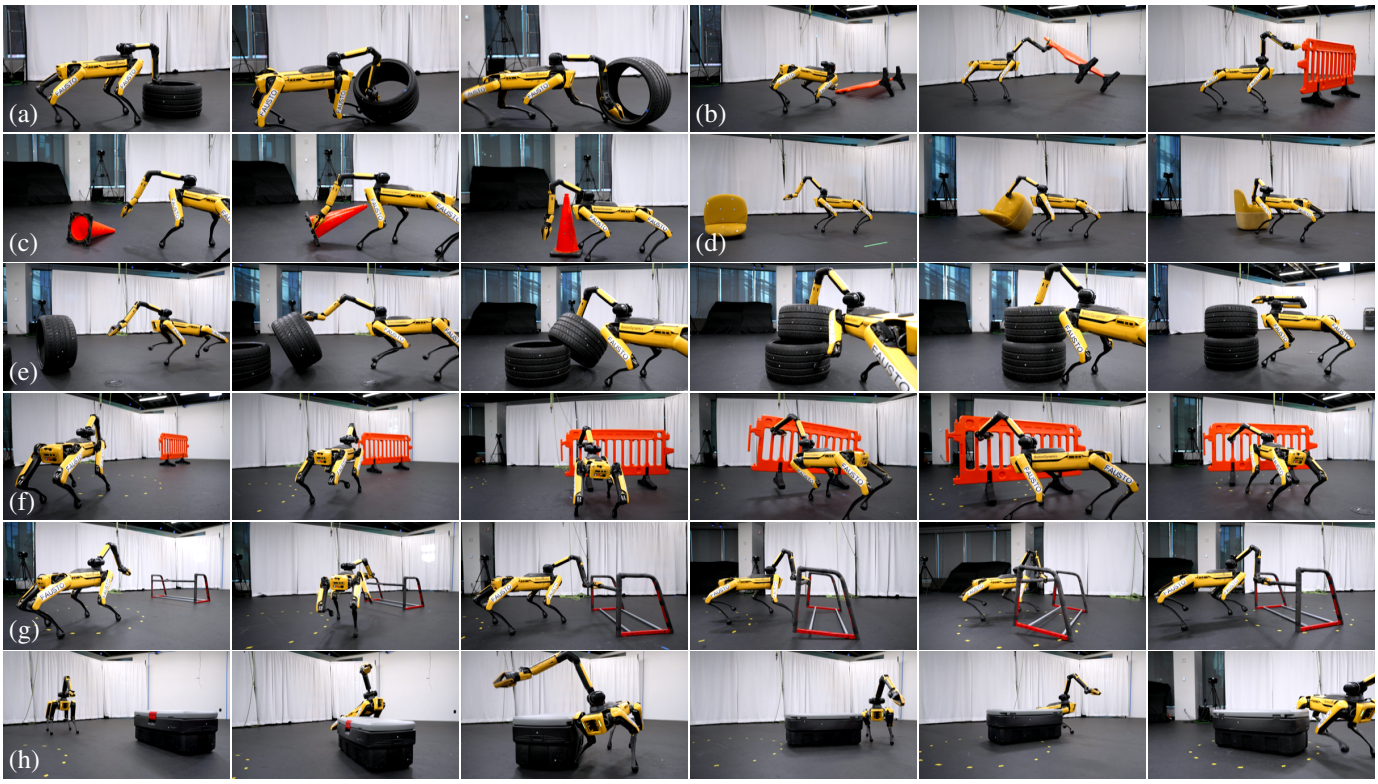

Fig. 3: Freeze-frame sequences showing Spot robot task progressions: (a) uprighting a tire, (b) uprighting a crowd control barrier, (c) uprighting a traffic cone, (d) uprighting a chair, (e) stacking a two tires, (f) dragging a crowd control barrier, (g) dragging a tire rack, and (h) pushing a heavy box.

often a non-trivial and manual search problem. Because of these practical shortcomings, researchers typically have to go through a time-consuming loop of reward engineering, policy training, and hardware testing, before going back to reward engineering for each task. While the recent emergence of LLMs [39, 22] can automate this process to some degree, reliably finding suitable rewards that lead to successful sim-to-real transfer still requires (expert) human-in-the-loop intervention, especially when policy optimization is time-consuming.

The sim-to-real RL paradigm has been particularly successful for legged robot locomotion [14, 7], where rewards have become relatively standardized and one can carefully design curricula for difficult terrain variations. However, in open-world manipulation, robots are faced with reasoning over a diverse set of scenarios including objects that vary in size, weight, geometry, friction, etc.. This test-time diversity makes randomizing all possible distributions challenging at training time for sim-to-real RL. As a result, successful real-world manipulation, so far, has come from learning from demonstrations [8].

In this work, we take advantage of RL's strength—learning robust locomotion policies over complex terrains offline—and leave the complexity of generalizable contact-rich decision making to online search via sample-based MPC.

B. Sample-Based MPC

Sample-based MPC is a family of zeroth-order trajectory-optimization methods that aims to solve for an optimal control sequence over a (typically short) prediction horizon. Sample-based MPC has been derived from many perspectives, including path integrals [36], information theory [37], importance sampling [42], and diffusion [26]. Additionally, sample-based MPC methods are closely related other derivative-free optimization algorithms such as CMA-ES [12]. Notably, similar to zeroth-order RL, which typically requires lots of compute *offline*, sample-based MPC can be viewed as an *online* policy-gradient method [28] that is entirely training-free, making it extremely flexible at test time.

Because of its parallel nature and ease of implementation, this family of trajectory optimization algorithms has grown in popularity in recent years, along with the rise of modern parallel computing hardware. The derivative-free nature of sample-based MPC is amenable to contact-rich dynamics and learned black-box deep neural network dynamics models, where computing derivatives is expensive or unreliable [35, 41], which often occurs in contact-rich control. In contrast, classical gradient-based methods such as the iterative linear-quadratic regulator (iLQR) [15, 20] and direct-collocation [17] typically struggle in these scenarios without additional care to modify the derivatives [9].

Online sample-based MPC has two main pitfalls: the curse of dimensionality and unstable dynamical systems. As an online search algorithm, sample-based MPC suffers from the curse of dimensionality that stems from high-dimensional input spaces associated with highly articulated robots and longer

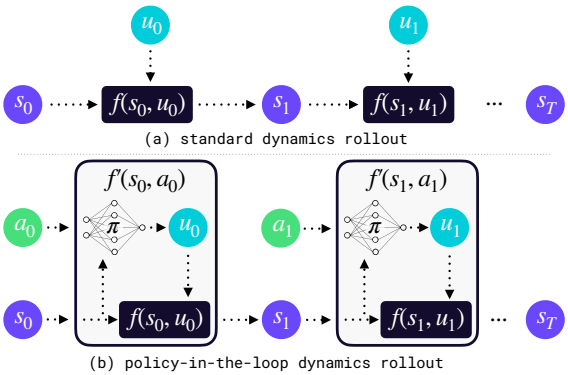

Fig. 4: Illustrations comparing (a) standard dynamics rollouts where the actions u are the joint-level controls for the multi-body dynamics model and (b) our network-policy-augmented dynamics rollouts where the actions a are inputs to the low-level locomotion policy.

prediction horizons. Recent works have proposed to reduce the temporal dimensionality by sampling over low-order spline control points [13, 19, 1], making sample-based MPC tractable in real-time for robots like dexterous hands, quadrupeds, and humanoids. Furthermore, sample-based MPC generally relies on single-shooting dynamics rollouts, leading to divergence on open-loop unstable systems, which commonly occur in legged robotics.

In our framework, by leveraging a low-level WBC policy, the sample-based MPC enjoys the benefit of sampling in a reduced action space on a stabilized dynamical system, drastically improving sample efficiency during online search.

C. Loco-Manipulation

Building robots that manipulate objects during locomotion, or so-called "loco-manipulation", has become an increasingly important yet challenging research direction as it inherits the fundamental challenges of both manipulation and locomotion for legged robots. Prior works have deployed both model-based methods [1] and RL methods [30] to achieve joint locomotion and manipulation of objects. Notably, several works have investigated using legs as manipulators [43, 6, 3, 5].

For humanoid loco-manipulation, retargeting human motions has become a popular method to bootstrap downstream learning and optimization [38, 40]. However, these teleoperation and retargeting approaches are much less applicable to non-anthropomorphic robots, like Spot. [31] explores building a planning and control framework for legged robot loco-manipulation, but the control module requires fixed contact modes during planning and is limited quasi-static behaviors.

Different from existing literature, we focus on manipulating objects that are larger than the robot itself or heavier than the maximum lifting capacity of the robot, thus requiring the robot to efficiently coordinate its entire body to solve these tasks. Our method naturally incorporates RL and online MPC, without simplifying assumptions to the robot dynamics and collision geometry [5] or computing fixed contact modes [31].

TABLE I: Timing comparison of MuJoCo simulation of 32 parallel rollouts over 1.5 seconds with and without the low-level policy on an Intel Core i7-12700K CPU.

Method	Rollout Time (ms)
MuJoCo with low-level policy	43.45 ± 1.88 ms
MuJoCo only	21.73 ± 1.86 ms

III. Sumo

In this section, we provide a detailed description of our hierarchical control framework, which we call Sumo, that uses a high-level sample-based controller to steer a pre-trained whole-body control policy to perform whole-body loco-manipulation with quadruped and humanoid robots.

A. Overview

Our system, detailed in Figure 2, uses a pre-pretrained steerable whole-body control policy that takes in the current state and desired torso, arm, and optionally leg commands of the robot and outputs the low-level joint commands for the quadruped or humanoid robot at 50Hz. At the high-level, we use a sample-based MPC policy (e.g., MPPI, CEM, etc.) that takes in the current state estimate and updates the desired torso and arm commands for the low-level locomotion policy at 20Hz. Together, our framework enables complex loco-manipulation skills that can only be achieved through whole-body contact-rich coordination, such as uprighting and stacking a fallen tire heavier than the robot's lifting capacity (Fig. 3 (a) (e)), and uprighting and dragging a crowd control barrier larger than the robot itself (Fig. 3 (b) (f)).

B. Pre-Trained Whole-Body Control Policy

For the low-level policy, we use an existing whole-body control policy trained via RL that takes high-level commands and outputs the joint-level commands for the entire robot. Methods for training humanoid and quadruped policies have been studied extensively in the literature and are widely available [14, 7]. We do not aim to improve whole-body control capabilities in this paper, and instead focus on their downstream applications for loco-manipulation. Specifically, we use the Relic policy [43] which is designed for multi-limb loco-manipulation on the Spot quadruped robot. In particular, the Relic policy enables stable gaits with only three legs and allows the robot to use the fourth leg as a manipulator in addition to its arm and torso. This setup allows the high-level sampling-based controller to automatically reason about using multiple limbs or a combination of limbs and torso to manipulate an otherwise challenging object.

For the G1 humanoid robot, we use the standard velocity-tracking policy from MJLab [24] that takes in the desired torso velocity commands. While this policy is not explicitly trained for loco-manipulation and does not take in the desired arm commands, we find that simply overriding the arm commands with target commands from the high-level sampling-based MPC is also effective.

Note that, while we believe these are natural choices for low-level policies for the Spot and G1 robots, one can similarly

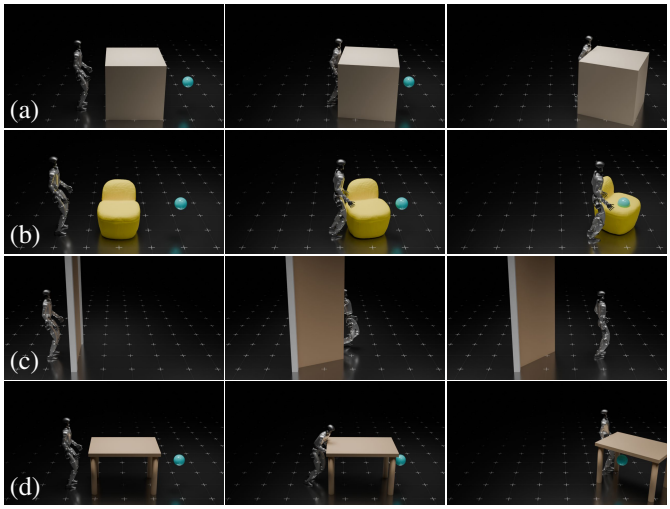

Fig. 5: Freeze-frame sequences showing G1 humanoid task progression: (a) pushing a box, (b) pushing a chair, (c) opening a door, and (d) pushing a table to a goal position (blue sphere).

choose other policies over arbitrary input spaces from which the sample-based controller can sample. We focus on the benefits of leveraging an existing whole-body control policy in this paper and leave testing different whole-body control policies to future work, as researchers actively expand the capabilities of whole-body control policies.

C. Policy-in-the-Loop Parallel Rollouts

A key component of our system is a policy-in-the-loop MuJoCo physics simulation that allows us to sample actions in the input space of the low-level locomotion policy instead of the whole-body joint-level action space of the entire robot as was done in previous works [1, 19]. This design choice is critical for online sample efficiency for several reasons: First, this allows the sample-based MPC to sample in a lower-dimensional action space. It is well documented that search algorithms like sampling-based MPC suffer from the curse of dimensionality as the action space grows in size. Second, with the low-level locomotion policy as a stabilizing controller, we avoid the well-known divergence issues associated with single-shooting methods applied to unstable dynamics that commonly occur with legged robots. Finally, because the whole-body control policy handles the locomotion part of the problem, this setup allows us to design much simpler cost function that only needs to encode manipulation objectives.

We implement the policy-in-the-loop parallel simulation by augmenting the CPU-based MuJoCo physics engine [34] in C++ using thread pool. In Table I we report the rollout times of 32 parallel rollouts over 1.5 seconds with and without the low-level policy on an Intel Core i7-12700K CPU. Indeed, the low-level policy adds overhead to the physics simulation, but the total times are faster than the 20Hz or 50ms update rate of the sample-based MPC.

D. High-Level Sample-Based MPC

Naturally, the high-level sample-based MPC can sample over any action space supported as inputs to the low-level policy. In practice, we choose to sample actions over a compact, task-relevant control vector, then rely on the low-level policy to map this to a full whole-body command. Here, we describe the details for the Spot robot with the Relic policy, but similar principles apply to other robots and low-level policies. We use $\mathbf{a}$ to denote the action space for the sample-based MPC, $\mathbf{c}$ to denote the commands to the WBC policy, and $\mathbf{u}$ to denote the joint-level control to the robot.

For the Spot robot, the full $\mathbf{c}_{\text{policy}} \in \mathbb{R}^{25}$ includes the $\text{SE}(2)$ velocities for the base $\mathbf{c}_{\text{base}} \in \mathbb{R}^3$, the arm joint-angle targets $\mathbf{c}_{\text{arm}} \in \mathbb{R}^6$, gripper position $\mathbf{c}_{\text{gripper}} \in \mathbb{R}^1$, leg joint angle targets $\mathbf{c}_{\text{leg}} \in \mathbb{R}^3$ for all four legs, and torso pitch, roll, and height targets $\mathbf{c}_{\text{torso}} \in \mathbb{R}^3$. By default, we only sample over the base velocities $\mathbf{a}_{\text{base}} \in \mathbb{R}^3$ and arm joint angle targets $\mathbf{a}_{\text{arm}} \in \mathbb{R}^6$ for the arm, excluding the gripper. To pad remaining policy commands, we set leg targets to zero (the low-level policy automatically outputs leg joint commands to track desired base velocities) and use default values for the torso pitch, roll, and height targets and set the gripper to a closed position. This base setup is sufficient for many loco-manipulation tasks but can be further enhanced by reasoning over legs, torso, and gripper commands at test time for tasks that require multi-limb coordination or gripper dexterity.

To reason over the torso roll, pitch, and height commands, we can simply add those dimensions $\mathbf{a}_{\text{torso}} \in \mathbb{R}^3$ to the action space. To enable the robot to use front legs for manipulation, we can additionally sample over $\mathbf{a}_{\text{leg}} \in \mathbb{R}^7 = [\mathbf{s}_{\text{leg}}, \mathbf{c}_{\text{leg}}]$ where $\mathbf{s}_{\text{leg}} \in [-1, 1]$ is a selection variable and $\mathbf{c}_{\text{leg}} \in \mathbb{R}^6$ is the leg joint angle targets, then use the selection variable to mask the leg joint angle targets:

$$\mathbf{c}_{\text{leg}}^{\text{masked}} = \begin{cases} [\mathbf{c}_{\text{leg}}[0{:}3], 0, 0, 0] & \text{if } s_{\text{leg}} < -0.5 \text{ (FL)} \\ [0, 0, 0, \mathbf{c}_{\text{leg}}[3{:}6]] & \text{if } s_{\text{leg}} > 0.5 \text{ (FR)} \\ [0, 0, 0, 0, 0, 0] & \text{else (no legs)} \end{cases} \tag{1}$$

Note that it is also possible to use rear legs for manipulation, but our implementation chooses to only focus on front legs as they cover most practical loco-manipulation tasks. Similarly, we can control the gripper via a binary action variable $\mathbf{a}_{\text{gripper}} \in [-1, 1]$ that maps to open and close positions:

$$\mathbf{c}_{\text{gripper}} = \begin{cases} \mathbf{c}_{\text{gripper close}} & \text{if } \mathbf{a}_{\text{gripper}} > 0 \\ \mathbf{c}_{\text{gripper open}} & \text{if } \mathbf{a}_{\text{gripper}} \leq 0 \end{cases} \tag{2}$$

This design allows us to flexibly choose task-specific action spaces to be any combination of $\mathbf{a} = [\mathbf{a}_{\text{base}}, \mathbf{a}_{\text{arm}}, \mathbf{a}_{\text{torso}}, \mathbf{a}_{\text{leg}}, \mathbf{a}_{\text{gripper}}]$ for the sample-based MPC. Additionally, the selection variables allow the controller to reason over different manipulation modalities at test time. Note that our sample-based MPC naturally reasons over both continuous and discrete decision variables which would be challenging for traditional gradient-based optimal control algorithms.

For the G1 humanoid robot, we similarly sample over the $\text{SE}(2)$ velocities for the torso and the arm joint angle targets for the arm. While it is straight forward to enable more inter-limb coordination by using a more steerable underlying

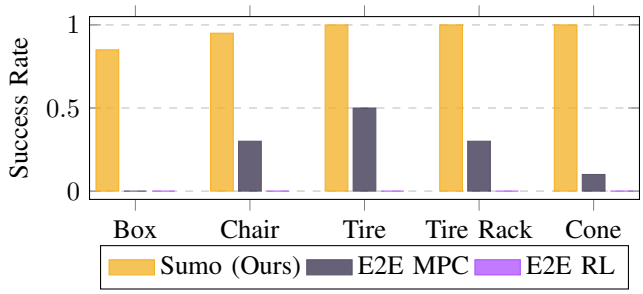

Fig. 6: Comparing Sumo (ours, yellow) to end-to-end MPC (green) and RL (purple) on 5 loco-manipulation tasks asking the robot to move an object to a goal. Our method significantly outperforms end-to-end baselines that struggle to maintain balance while manipulating objects without a stabilizing policy.

low-level policy [21], we leave this investigation as a future direction.

IV. Experimental Analysis

In this section, we perform simulation experiments in MuJoCo [34] to demonstrate the advantages of Sumo's two-stage optimization framework.

A. Task Definition and Experiment Setup

In this section, we consider two types of loco-manipulation tasks on the Spot quadruped robot: moving an object to a goal position (Move) and reorienting an object to the upright orientation (Upright). In both cases, we consider five objects covering a range of size, weight, and geometry representative of real-world scenarios: a 1.5 kg box, a chair, a car tire, a tire rack, and a traffic cone.

For Move tasks, we consider success if the object is within 0.1 meters of the goal and velocity is less than 0.05 m/s within 30 seconds. For Upright tasks, we consider task success if the object is within 0.1 radians of the upright orientation and angular velocity is less than 0.05 rad/s within 30 seconds.

To simplify the analysis, we restrict the Sumo high-level planner to only control the torso $\mathrm{SE}(2)$ velocities and joint angle targets for the arm in this section and leave exploring more expressive policy behaviors to Sec. V. Additionally, we design a minimal cost function for Move tasks:

$$J_{\text{Move}} = w_{\text{goal}} \cdot \|\mathbf{p}_{\text{obj}} - \mathbf{p}_{\text{goal}}\|_2 + w_{\text{gripper}} \cdot \|\mathbf{p}_{\text{gripper}} - \mathbf{p}_{\text{obj}}\|_2 + w_{\text{vel}} \cdot \|\mathbf{v}_{\text{obj}}\|_2 \tag{3}$$

where $\mathbf{p}_{\text{obj}}$, $\mathbf{p}_{\text{goal}}$, and $\mathbf{p}_{\text{gripper}}$ are the positions of the object, goal, and gripper in world frame, respectively, $\mathbf{v}_{\text{obj}}$ is the linear velocity of the object, and $w_{\text{goal}}, w_{\text{gripper}}, w_{\text{vel}}$ are weighting coefficients. Similarly, for Upright tasks:

$$J_{\text{Upright}} = w_{\text{Upright}} \cdot \|\mathbf{q}_{\text{obj}} - \mathbf{q}_{\text{upright}}\|_2 + w_{\text{gripper}} \cdot \|\mathbf{p}_{\text{gripper}} - \mathbf{p}_{\text{obj}}\|_2 \tag{4}$$

where $\mathbf{q}_{\text{obj}}$ and $\mathbf{q}_{\text{upright}}$ are the quaternions of the object and upright orientation, respectively, and $w_{\text{Upright}}, w_{\text{gripper}}$ are weighting coefficients.

For all Sumo experiments, we use a Cross-Entropy Method (CEM) optimizer with a prediction horizon of 1.5 seconds, 32 rollouts, and update the action distribution with three elite samples at a frequency of 20Hz. We sample over four spline control points along the prediction horizon and interpolate the remaining actions. Finally, we linearly schedule the noise variance ramping from 0.02 to 0.6 over the prediction horizon. All experiments are conducted synchronously on a desktop computer equipped with an Intel Core i7-12700K CPU and 64GB of RAM. Current robot and object states are provided by the ground-truth simulator.

B. Hierarchical Control is Key for Loco-Manipulation

First, we show that our two-stage framework (Sumo) involving pre-training in RL and real-time search strictly outperforms both real-time MPC (E2E MPC) and RL (E2E RL) in Move tasks.

For the E2E MPC baseline, we use the same CEM optimizer, which is used in Sumo and common for locomotion and manipulation [19, 13, 1], whose action space is the joint-level commands of the robot. We design rewards to include terms for both locomotion and manipulation objectives:

$$J_{\text{E2E MPC}} = J_{\text{Locomotion}} + J_{\text{Move}} \tag{5}$$

where $J_{\text{Locomotion}}$ is the locomotion-related cost from [13] and J_{Move} is defined in (3). For the RL baseline, we train a PPO policy using the popular mjlab [24] framework and make our best-faith attempt to augment existing velocity-tracking rewards with manipulation objectives. In this experiment, we train a separate policy for each object-moving task. In both baselines, we make a best-faith effort to tune hyperparameters to achieve the best performance.

We find that our method successfully solves all five Move tasks with at least 80% success across 20 evaluation trials, with some tasks at 100% success. In contrast, E2E MPC struggles to consistently solve these tasks, achieving at most 50% success. We hypothesize that this degradation in performance is due to the increased search space and instability of the robot's dynamics when manipulating objects in the absence of a stabilizing policy. E2E RL fails to solve any of the tasks despite our best efforts to shape rewards and design curricula. We summarize these results in Fig. 6.

These experiments demonstrate that using a two-stage hierarchical control framework leveraging a pre-trained RL policy simplifies solving complex loco-manipulation tasks.

C. Test-Time Search Enables Generalization

We have shown that leveraging a pre-trained low-level policy is important for solving complex loco-manipulation tasks. Yet, there still remain multiple options for the high-level planner, namely RL and MPC. While single-task RL is effective on many tasks, training a multi-task RL policy conditioned on the task-specific object model and sufficient domain randomization is still challenging. More importantly, when training an RL policy, one typically assumes a *known* distribution of environments and objects at test time. Despite

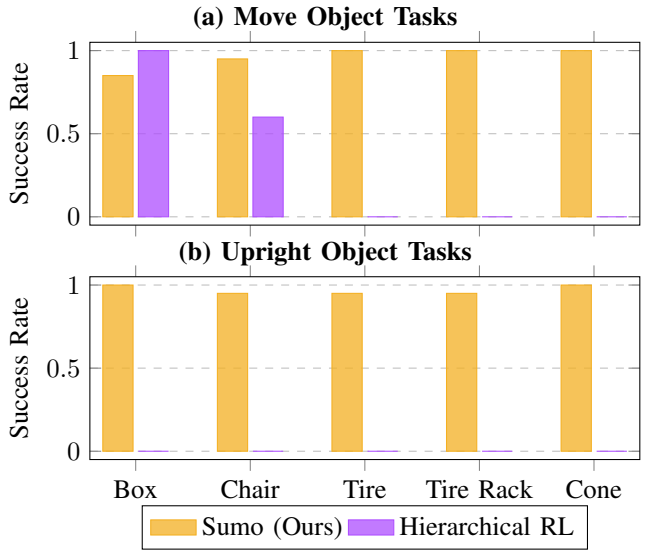

Fig. 7: Top: comparing Sumo (yellow, ours) and hierarchical RL (purple) on pushing 5 different objects to a goal. Our method, relying on test-time planning, generalizes by simply replacing the object model at test time, where Hierarchical RL trained on push box fails to generalize. Bottom: Sumo generalizes to uprighting objectives at test time, where the same hierarchical RL policy fails.

RL's success in robot locomotion, where robot and terrain model distributions can be randomized at training time, we believe open-world manipulation tasks are simply too diverse and complex to be modeled at training time, making test-time generalization to *unknown* distributions a necessity.

In Fig. 7 (a), we train a hierarchical RL policy to move a 1.5 kg box to a goal and randomize the box size, weight, and friction properties. Indeed, while the hierarchical RL policy achieves a 100% success rate on this task, its performance degrades quickly when the object geometry and weight deviate from the training distribution (e.g., moving a tire or a traffic cone). In contrast, our method with online planning can generalize to different objects *without additional training* by simply replacing the object model. All tasks here use the same cost functions and hyperparameter described in Eq. (3).

Furthermore, test-time search allows us to flexibly adjust the cost function at test time to generalize to new task objectives, similar to guidance for diffusion models [16]. We demonstrate this idea in Fig. 7 (b), where we replace the goal-redirecting objective with an orientation objective in Eq. (4) to enable the robot to reliably upright different objects without additional training. In contrast, the same RL policy cannot be steered to upright different objects without additional training.

D. Sample-Based MPC Provides High Practical Efficiency

Finally, we demonstrate the superior practical wall-clock efficiency of sample-based control over RL. In practice, both sample-based MPC and sim-to-real RL algorithms require hyperparameter tuning to achieve good performance. In Fig. 8, we mimic this process by performing a Bayesian optimization over the cost weights in (3) for the Move Box task for both Hierarchical RL and Sumo to find the weights that maximize

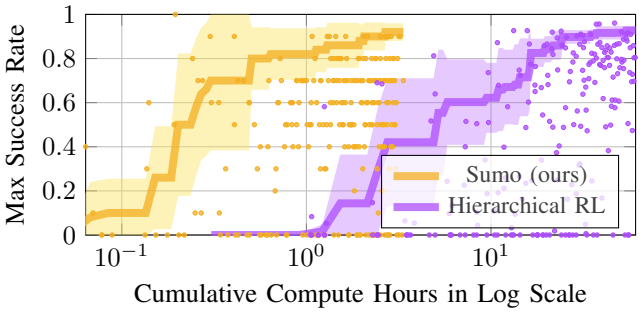

Fig. 8: Maximum success rate vs compute time comparison between Sumo (blue) and hierarchical RL (red) baseline during hyperparameter Bayesian optimization. Solid lines represent the mean across 5 optimization runs, and the shaded region shows ± 1 standard deviation. Individual success rates are shown as scatter points. Our method achieves similar asymptotic performance compared to hierarchical RL in an order of magnitude less wall-clock time.

the success rate. We perform five independent optimization runs and track the maximum success rate over time across 50 trials. We find that both methods achieve similar asymptotic performance, but Sumo requires an order-of-magnitude less compute time to achieve the same performance. For Sumo, we measure the compute time as CPU hours on a desktop computer equipped with an Intel Core i7-12700K CPU and 64GB of RAM, and RL training time is measured as GPU hours on an NVIDIA RTX A6000 GPU.

V. Robot Demonstrations with Sumo

In this section, we demonstrate the capabilities of our loco-manipulation framework on eight Spot tasks in the real world and four G1 tasks in simulation. We plan to open-source all tasks as a whole-body loco-manipulation benchmark.

A. Spot Quadruped Loco-Manipulation

In the first case study, we show that our hierarchical control framework can enable the Boston Dynamics Spot quadruped robot to perform a variety of challenging loco-manipulation tasks. Many of these tasks involve the manipulation of objects that are larger than the robot and heavier than the robot's payload capacity, thus requiring its entire body. The tasks and corresponding performance are summarized in Table II.

For all hardware deployments, we use the same set of optimizer parameters as detailed in Sec. IV. To estimate the robot and object states, we fuse the Spot robot joint encoder reading at 333Hz with motion-capture (MoCap) measurements for torso and object positions and orientations at 120Hz using a low-pass filter. We compute the dynamics rollouts and CEM updates asynchronously on a desktop computer equipped with an AMD Threadripper Pro 5995WX CPU with 64 cores, which sends joint-level commands to the Spot robot via WiFi. Cost functions are provided in the appendix. **Tire Upright:** In Fig. 3 (a), the robot is tasked to upright a tire of 15kg, which is heavier than the maximum lifting capacity of 11kg of the Spot arm. Additionally, the rubber tire presents hard-to-model geometries and friction properties, causing a large sim-to-real

TABLE II: Real-world completion time and success rate performance of Spot loco-manipulation tasks across 10 trials.

Task Name	Completion Time	Successes	Time Limit
Tire Upright	9.2 ± 4.7s	10/10	30s
Barrier Upright	10.5 ± 7.1s	9/10	30s
Cone Upright	10.2 ± 7.9s	9/10	30s
Chair Upright	27.3 ± 19.1s	8/10	60s
Tire Stack	16.5 ± 8.4s	8/10	30s
Barrier Drag	20.2 ± 6.7s	9/10	30s
Tire Rack Drag	19.1 ± 6.2s	9/10	30s
Rugged Box Push	38.3 ± 16.9s	10/10	90s

gap. Using a combination of its arm, torso, and legs, Sumo enables the robot to complete the task 10 out of 10 trials with an average completion time of 9.2 ± 4.7s, Tab. II.

Crowd Barrier Upright: In Fig. 3 (b), the robot is tasked to upright a crowd barrier of 16kg from a lying position. Using its arm and gripper, our method enables the robot to complete the task 9 out of 10 trials with an average completion time of 10.5 ± 7.1s, Tab. II.

Traffic Cone Upright: In Fig. 3 (c), the robot is tasked to upright a traffic cone of 3.5kg. Using its arm, our method enables the robot to complete the task 9 out of 10 trials with an average completion time of 10.2 ± 7.9s, Tab. II.

Chair Upright: In Fig. 3 (d), the robot is tasked to upright a chair of 16.5kg. Using its arm and body, our method enables the robot to complete the task 8 out of 10 trials with an average completion time of 27.3 ± 19.1s, Tab. II.

Tire Stack: In Fig. 3 (e), the robot is tasked to stack a 15kg tire on top of another tire. In addition to large weight and deformation, the friction coefficient between the rubber tires is high, which current simulators struggle to simulate accurately, causing an even larger sim-to-real gap. Using a combination of its arm, torso, and legs, our method enables the robot to complete the task 8 out of 10 trials with an average completion time of 16.5 ± 8.4s, Tab. II.

Crowd Barrier Drag: In Fig. 3 (f), the robot is tasked to drag a crowd barrier of 15kg in upright orientation. The robot uses the gripper to grasp the barrier and drags it towards a goal position with its arm and body. Our method enables the robot to complete the task 8 out of 10 trials with an average completion time of 20.2 ± 6.7s, Tab. II.

Tire Rack Drag: In Fig. 3 (g), the robot is tasked to drag a tire rack of 10kg. The robot uses the gripper to grasp the tire rack and drags it towards a goal position. Our method enables the robot to complete the task 9 out of 10 trials with an average completion time of 19.1 ± 6.2s, Tab. II.

Rugged Box Push: In Fig. 3 (h), the robot is tasked to push a 20kg box to a goal. Using its arm and body, our method enables the robot to complete the task 10 out of 10 trials with an average completion time of 38.3 ± 16.9s, Tab. II.

B. G1 Humanoid Loco-Manipulation

In the second case study, we show that Sumo can also be applied to humanoid robots. Through simulated experiments using a Unitree G1, our method uses a pre-trained locomotion policy that is widely available [24], and solves loco-manipulation tasks such as pushing a box and opening a door.

TABLE III: Performance of G1 humanoid loco-manipulation tasks in simulation.

Task Name	Completion Time (s)	Successes	Time Limit
G1 Box Push	11.83 ± 2.97 s	9/10	30s
G1 Door Open	4.73 ± 0.98 s	10/10	30s
G1 Chair Push	6.86 ± 0.288 s	10/10	30s
G1 Table Push	4.86 ± 1.65 s	8/10	30s

Simulation performances are summarized in Table III.

Box Pushing In Fig. 5 (a), the robot is tasked to push a 10kg box to a target position. Using its arms and body, our method enables the robot to complete the task 9 out of 10 trials with an average completion time of 11.83 ± 2.97s, Tab. III.

Chair Pushing In Fig. 5 (b), the robot is tasked with pushing a 16.5kg chair to a goal. Using its arms and body, our method enables the robot to complete the task 10 out of 10 trials with an average completion time of 6.86 ± 0.288s, Tab. III.

Door Opening In Fig. 5 (c), the robot is asked to open a door. Using its arm and body, our method enables the robot to complete the task 10 out of 10 trials with an average completion time of 4.73 ± 0.98s, Tab. III.

Table Pushing In Fig. 5 (d), the robot is tasked with pushing a 10kg table to a goal. Using its arms and body, our method enables the robot to complete the task 8 out of 10 trials with an average completion time of 4.86 ± 1.65s, Tab. III.

VI. Conclusions and Future Work

This paper explores the combination of RL pre-training and test-time sample-based MPC for whole-body loco-manipulation. We find this approach, which we call Sumo, excels at challenging tasks involving large and heavy objects that cannot be solved via pick-and-place-style manipulation. We also find that the decomposition of difficult loco-manipulation tasks into a two-stage optimization problem makes the sub-problems much more tractable. Additionally, we show that using sample-based MPC as a high-level policy enables test-time task flexibility and generalization that is difficult for RL.

Sumo still has several shortcomings and avenues for future work: First, we rely on a MoCap system for robot and object state estimation and are restricted to laboratory settings. Future work should explore incorporating fully onboard perception systems that allow the robot to navigate challenging terrains during loco-manipulation. Second, as an algorithm relying entirely on sim-to-real transfer, the "sim-to-real gap" remains a significant challenge. While modeling the robot and object dynamics perfectly is impossible and often not necessary, building a "good enough" model for control remains an art that requires some trial and error. In fact, we believe real-time MPC as the high-level policy significantly accelerates this tuning loop compared to RL training. Nevertheless, Sumo can undoubtedly benefit from system-identification and model-learning methods that update object parameters on the fly from real-world interactions. Finally, Sumo, in its current form, does not leverage any human priors. Future work should explore incorporating pre-trained foundation models that can automatically guide robot behaviors without manually engineered rewards.

References

[1] Juan Alvarez-Padilla, John Z Zhang, Sofia Kwok, John M Dolan, and Zachary 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)*, pages 14721–14727. IEEE, 2025. doi: 10.1109/ICRA55743.2025.11128271.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, volume 30, pages 5048–5058, 2017.

[3] Philip Arm, Mayank Mittal, Hendrik Kolvenbach, and Marco Hutter. Pedipulate: Enabling manipulation skills using a quadruped robot's leg. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5717–5723, 2024. doi: 10.1109/ICRA57147.2024.10611307.

[4] Gerardo Bledt, Matthew J. Powell, Benjamin Katz, Jared Di Carlo, Patrick M. Wensing, and Sangbae Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252, 2018. doi: 10.1109/IROS.2018.8593885.

[5] Jin Cheng, Dongho Kang, Gabriele Fadini, Guanya Shi, and Stelian Coros. Rambo: Rl-augmented model-based whole-body control for loco-manipulation. *IEEE Robotics and Automation Letters*, 10(9):9462–9469, 2025. doi: 10.1109/LRA.2025.3594984.

[6] Xuxin Cheng, Ashish Kumar, and Deepak Pathak. Legs as manipulator: Pushing quadrupedal agility beyond locomotion. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[7] Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme parkour with legged robots. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 11443–11450, 2024. doi: 10.1109/ICRA57147.2024.10610200.

[8] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

[9] Simon Le Cleac'h, Taylor Howell, Shuo Yang, Chi-Yen Lee, John Zhang, Arun Bishop, Mac Schwager, and Zachary Manchester. Fast Contact-Implicit Model-Predictive Control. *IEEE Transactions on Robotics and Automation*, January 2024. doi: 10.48550/arXiv.2107.05616.

[10] Google DeepMind and NVIDIA. Mujoco warp: Gpu-optimized version of the mujoco physics simulator. https://github.com/google-deepmind/mujoco_warp, 2025. Accessed: 2025-11-19.

[11] Ruben Grandia, Fabian Jenelten, Shaohui Yang, Farbod Farshidian, and Marco Hutter. Perceptive locomotion through nonlinear model-predictive control. *IEEE Transactions on Robotics*, 39(5):3402–3421, 2023. doi: 10.1109/TRO.2023.3275384.

[12] Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[13] Taylor Howell, Nimrod Gileadi, Saran Tunyasuvunakool, Kevin Zakka, Tom Erez, and Yuval Tassa. Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo. *arXiv preprint arXiv:2212.00541*, 2022.

[14] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario C. Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019. doi: 10.1126/scirobotics.aau5872.

[15] D. H. Jacobson and D. Q. Mayne. *Differential Dynamic Programming*. Elsevier, 1970.

[16] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

[17] Matthew Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017. doi: 10.1137/16M1062569.

[18] Albert H. Li, Preston Culbertson, Vince Kurtz, and Aaron D. Ames. Drop: Dexterous reorientation via online planning. *arXiv preprint arXiv:2409.14562*, 2024. URL https://arxiv.org/abs/2409.14562.

[19] Albert H. Li, Brandon Hung, Aaron D. Ames, Jiuguang Wang, Simon Le Cleac'h, and Preston Culbertson. Judo: A user-friendly open-source package for sampling-based model predictive control. In *Proceedings of the Workshop on Fast Motion Planning and Control in the Era of Parallelism at Robotics: Science and Systems (RSS)*, 2025. URL https://github.com/bdaiinstitute/judo.

[20] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 1, pages 222–229. INSTICC, SciTePress, 2004. doi: 10.5220/0001143902220229.

[21] Qiayuan Liao, Takara E. Truong, Xiaoyu Huang, Yuman Gao, Guy Tevet, Koushil Sreenath, and C. Karen Liu. Beyondmimic: From motion tracking to versatile humanoid control via guided diffusion. *arXiv preprint arXiv:2508.08241*, 2025.

[22] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv: Arxiv-2310.12931*, 2023.

[23] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg.

Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi: 10.1109/LRA.2023.3270034.

[24] MuJoCo Lab Contributors. mjlab: Isaac lab api, powered by mujoco-warp, for rl and robotics research. https://github.com/mujocolab/mjlab, 2025. Accessed: 2025-11-20.

[25] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*, pages 278–287, 1999.

[26] Chaoyi Pan, Zeji Yi, Guanya Shi, and Guannan Qu. Model-based diffusion for trajectory optimization. 2024. URL https://arxiv.org/abs/2407.01573.

[27] Haozhi Qi, Brent Yi, Mike Lambeta, Yi Ma, Roberto Calandra, and Jitendra Malik. From simple to complex skills: The case of in-hand object reorientation. *arXiv preprint arXiv:2501.05439*, 2025.

[28] Junbin Qiu, Zhengpeng Xie, Xiangda Yan, Yongjie Yang, and Yao Shu. Zeroth-order optimization is secretly single-step policy optimization. *arXiv preprint arXiv:2506.14460*, 2025.

[29] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[30] Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation. 2024.

[31] Jean-Pierre Sleiman, Farbod Farshidian, and Marco Hutter. Versatile multicontact planning and control for legged loco-manipulation. *Science Robotics*, 8(81):eadg5014, 2023. doi: 10.1126/scirobotics.adg5014.

[32] H.J. Terry Suh, Tao Pang, Tong Zhao, and Russ Tedrake. Dexterous contact-rich manipulation via the contact trust region. *arXiv preprint arXiv:2505.02291*, 2025.

[33] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.

[34] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

[35] Kevin Tracy, John Z. Zhang, Jon Arrizabalaga, Stefan Schaal, Yuval Tassa, Tom Erez, and Zachary Manchester. The trajectory bundle method: Unifying sequential-convex programming and sampling-based trajectory optimization. *arXiv preprint arXiv:2509.26575*, 2025.

[36] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440, May 2016. doi: 10.1109/ICRA.2016.7487277.

[37] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Information-Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, December 2018. ISSN 1941-0468. doi: 10.1109/TRO.2018.2865891.

[38] Lujie Yang, Xiaoyu Huang, Zhen Wu, Angjoo Kanazawa, Pieter Abbeel, Carmelo Sferrazza, C. Karen Liu, Rocky Duan, and Guanya Shi. Omniretarget: Interaction-preserving data generation for humanoid whole-body loco-manipulation and scene interaction. 2025. URL https://arxiv.org/abs/2509.26633.

[39] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, Brian Ichter, Ted Xiao, Peng Xu, Andy Zeng, Tingnan Zhang, Nicolas Heess, Dorsa Sadigh, Jie Tan, Yuval Tassa, and Fei Xia. Language to rewards for robotic skill synthesis. *Arxiv preprint arXiv:2306.08647*, 2023.

[40] John Z. Zhang, Shuo Yang, Gengshan Yang, Arun L. Bishop, Swaminathan Gurumurthy, Deva Ramanan, and Zachary Manchester. Slomo: A general system for legged robot motion imitation from casual videos. 2023. URL https://ieeexplore.ieee.org/abstract/document/10246373.

[41] John Z. Zhang, Taylor A. Howell, Zeji Yi, Chaoyi Pan, Guanya Shi, Guannan Qu, Tom Erez, Yuval Tassa, and Zachary Manchester. Whole-body model-predictive control of legged robots with mujoco. 2025. URL https://arxiv.org/abs/2503.04613.

[42] Wei Zhang, Han Wang, Carsten Hartmann, Marcus Weber, and Christof Schütte. Applications of the cross-entropy method to importance sampling and optimal control of diffusions. *SIAM Journal on Scientific Computing*, 36(6):A2654–A2672, 2014. doi: 10.1137/14096493X. URL https://doi.org/10.1137/14096493X.

[43] Xinghao Zhu, Yuxin Chen, Lingfeng Sun, Farzad Niroui, Simon Le Cleac'h, Jiuguang Wang, and Kuan Fang. Relic: Versatile loco-manipulation through flexible interlimb coordination. *arXiv preprint arXiv:2506.07876*, 2025.