

Whole-Body Inverse Dynamics MPC for Legged Loco-Manipulation

Lukas Molnar¹, Jin Cheng², Gabriele Fadini², Dongho Kang², Fatemeh Zargarbashi², Stelian Coros²

Abstract—Loco-manipulation demands coordinated whole-body motion to manipulate objects effectively while maintaining locomotion stability, presenting significant challenges for both planning and control. In this work, we propose a whole-body model predictive control (MPC) framework that directly optimizes joint torques through full-order inverse dynamics, enabling unified motion and force planning and execution within a single predictive layer. This approach allows emergent, physically consistent whole-body behaviors that account for the system’s dynamics and physical constraints. We implement our MPC formulation using open software frameworks (*Pinocchio* and *CasADi*), along with the state-of-the-art interior-point solver *Fatrop*. In real-world experiments on a *Unitree B2* quadruped equipped with a *Unitree Z1* manipulator arm, our MPC formulation achieves real-time performance at 80 Hz. We demonstrate loco-manipulation tasks that demand fine control over the end-effector’s position and force to perform real-world interactions like pulling heavy loads, pushing boxes, and wiping whiteboards.

I. INTRODUCTION

Legged robots equipped with robotic arms offer a powerful combination of mobility and manipulation capabilities, allowing them to interact with the environment while moving through it. To fully leverage this capability, such robots must execute precise end-effector tasks, as well as apply and resist substantial forces when pulling or pushing heavy objects, all while maintaining locomotion stability. These loco-manipulation tasks introduce strong coupling between the arm and the rest of the body, motivating control approaches that jointly optimize whole-body motion and force generation while enforcing physical constraints of the robotic system.

In this work, we address these challenges by developing a whole-body inverse dynamics model predictive control (MPC) framework that enables legged robots with arms to perform loco-manipulation tasks involving both precision and strong physical interaction. Unlike traditional methods for loco-manipulation that rely on reduced-order planning followed by whole-body tracking controllers, our approach directly optimizes whole-body dynamics at the torque level. This unified formulation enables simultaneous reasoning about motion and force generation while explicitly respecting actuation limits. As a result, we achieve emergent, dynamically consistent behaviors without relying on a hierarchical control stack, simplifying the overall pipeline.

¹Lukas Molnar is with the Department of Mechanical and Process Engineering, ETH Zurich, Switzerland. lukas.molnar@bluewin.ch.

²The remaining authors are with Computational Robotics Lab, Department of Computer Science, ETH Zurich, Switzerland. {jicheng, gfadini, kangd, fzargarbashi, scoros}@ethz.ch.

This work was supported by ETH Zurich RobotX Research Program. Project website: lukasmolnar.github.io/wb-mpc-locoman

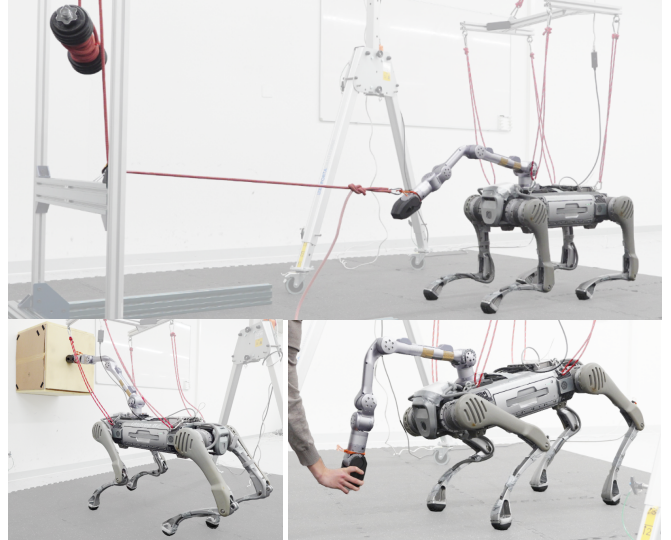


Fig. 1: *Unitree B2* quadruped with a *Z1* manipulator arm: Pulling a 10 kg load in stance (**top**), pushing a box against the wall while walking (**bottom left**), and demonstrating compliant human interaction (**bottom right**).

We validate our approach on the *Unitree B2* quadruped equipped with a *Z1* manipulator arm, demonstrating real-time performance with the MPC solving at 80 Hz. In simulation, the controller enables accurate end-effector tracking across diverse whole-body motions and remains robust to external disturbances on the base. On hardware, the robot performs physically interactive tasks such as walking while pulling a 10 kg load, demonstrating the controller’s ability to coordinate locomotion and force-level manipulation under real-world conditions.

The main contributions of this work are as follows:

- 1) A whole-body torque-level MPC formulation based on inverse dynamics for legged loco-manipulation.
- 2) A custom optimization framework built on *Pinocchio* [1] and *CasADi* [2], supporting multiple dynamics formulations for benchmarking.
- 3) Real-time deployment using the interior-point solver *Fatrop* [3], with the MPC solution applied directly to hardware without a whole-body tracking controller.

II. RELATED WORK

Control of legged robots has historically relied on a hierarchical structure that separates high-level planning from low-level control. At the highest level, many locomotion frameworks use a single rigid body (SRB) dynamics model to plan contact locations, contact forces, and a center-of-mass or base trajectory [4–7]. These models approximate the robot as a floating rigid body with constant inertia, ignoring joint-level kinematics, limb mass distribution, and actuation limits.

To execute these plans on hardware, a low-level whole-body controller (WBC) with prioritized task execution is typically used [5, 8], often relying on heuristics such as Bezier curves for swing leg trajectories.

Due to the computational efficiency of SRB models, they have recently been extended to loco-manipulation tasks [9, 10]. However, these methods require an additional higher-level planner to generate kinematically feasible references for the SRB model to track. This layered approach introduces hierarchical complexity, which requires tuning for each layer and can complicate deployment on hardware. Moreover, because SRB models assume massless limbs, they become inaccurate for systems with heavy manipulators, where the arm significantly affects the overall dynamics.

To improve consistency between planning and control, centroidal dynamics models have been introduced for quadruped and humanoid locomotion [11–14]. These models retain computational efficiency by focusing on the center of mass evolution, while incorporating the robot’s full kinematic structure through the centroidal momentum matrix. This allows the planner to reason about joint motions and contact forces more accurately, enabling dynamic locomotion behaviors. For loco-manipulation, recent work has adopted centroidal dynamics for the ANYmal quadruped with a 4-DoF [15] and 6-DoF manipulator [16, 17], achieving dynamic tasks such as door opening and heavy object pushing. The same dynamics method is applied to the 37-DoF dual-arm quadruped CENTAURO [18], demonstrating the scalability of centroidal models for high-dimensional systems.

A key limitation of these methods is that they rely on additional steps to ensure dynamic feasibility at the joint level. For the ANYmal robot [15, 16], a low-level whole-body controller based on hierarchical quadratic programming (QP) tracks the motions planned by the centroidal MPC, adding complexity to the control pipeline. In contrast, for the CENTAURO robot [18], joint torques are computed directly from the MPC solution via inverse dynamics, eliminating the need for a separate tracking controller. While this simplifies the control architecture, it assumes that the centroidal model produces motions consistent with the whole-body dynamics, which may result in unsafe torques in practice.

Whole-body torque-level MPC offers a unified optimization framework for simultaneous planning and control, and has shown success for both quadruped [19, 20] and humanoid locomotion [21–23]. By directly modeling the full rigid-body dynamics and optimizing over joint torques, this approach enables physically consistent trajectory generation without requiring a separate low-level tracking controller. This is particularly appealing for loco-manipulation tasks involving intense environmental interactions—such as pushing or pulling heavy objects—where accurate modeling of force transmission and joint-level actuation is essential.

Despite the advantages, applying torque-level whole-body MPC to loco-manipulation remains largely unexplored, primarily due to the computational challenges of solving high-dimensional optimization problems in real-time. For pure locomotion, most real-time methods avoid optimizing contact

forces explicitly, instead enforcing contact constraints within the dynamics [14, 20–22]. While this reduces the number of decision variables, it is not well-suited for manipulation tasks that require simultaneous control of end-effector force and motion—for example, when pulling a heavy object.

Full whole-body formulations that explicitly optimize both contact forces and joint torques [23, 24, 19] are better suited for loco-manipulation, but have received limited attention in this context. To date, such methods have only been applied offline for legged loco-manipulation [25], without demonstrating real-time control. In this work, we address this gap by achieving real-time, torque-level whole-body MPC for loco-manipulation.

III. PRELIMINARIES

A. Model Predictive Control

Model Predictive Control (MPC) is an optimization-based control strategy that computes actions by solving a finite-horizon optimal control problem (OCP) at each time step. The general form of the OCP for a discrete-time system can be expressed as:

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{u}} \quad & \sum_{k=0}^{N-1} \ell(\mathbf{x}_k, \mathbf{u}_k) + \ell_N(\mathbf{x}_N) \\ \text{s.t.} \quad & \mathbf{x}_0 = \hat{\mathbf{x}}, \\ & \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \quad \forall k = 0, \dots, N-1 \\ & \mathbf{g}(\mathbf{x}_k, \mathbf{u}_k) \leq 0, \quad \forall k = 0, \dots, N-1 \\ & \mathbf{g}_N(\mathbf{x}_N) \leq 0 \end{aligned} \quad (1)$$

Here, \mathbf{x}_k and \mathbf{u}_k denote the system state and control input at time step k , respectively. $\hat{\mathbf{x}}$ is the measured initial state, $\ell(\cdot)$ is the stage cost and $\ell_N(\cdot)$ the terminal cost. The function $\mathbf{f}(\cdot)$ represents the system dynamics, $\mathbf{g}(\cdot)$ and $\mathbf{g}_N(\cdot)$ represent path constraints on each node, which may include both equality and inequality conditions.

B. Whole-Body Dynamics

We model the quadruped and manipulator system using a floating-base representation. The generalized coordinate vector is defined as $\mathbf{q} = [\mathbf{p}_b, \mathbf{q}_b, \mathbf{q}_j]$, consisting of the base position $\mathbf{p}_b \in \mathbb{R}^3$, base orientation $\mathbf{q}_b \in SO(3)$ represented as a unit quaternion, and joint angles $\mathbf{q}_j \in \mathbb{R}^{n_j}$. The corresponding generalized velocity and acceleration vectors are given by $\mathbf{v} \in \mathbb{R}^{6+n_j}$, $\mathbf{a} \in \mathbb{R}^{6+n_j}$, respectively, including linear and angular components for the base.

The full rigid-body (whole-body) dynamics are given by the following equations of motion:

$$\mathbf{M}(\mathbf{q})\mathbf{a} + \mathbf{b}(\mathbf{q}, \mathbf{v}) = \begin{bmatrix} \mathbf{0}_{6 \times 1} \\ \boldsymbol{\tau}_j \end{bmatrix} + \mathbf{J}_c^\top(\mathbf{q})\mathbf{F}_c, \quad (2)$$

where $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{(6+n_j) \times (6+n_j)}$ denotes the mass matrix, and $\mathbf{b}(\mathbf{q}, \mathbf{v}) \in \mathbb{R}^{6+n_j}$ represents the Coriolis and gravitational forces. The vector $\mathbf{F}_c = [\mathbf{F}_{c_1}^\top, \dots, \mathbf{F}_{c_4}^\top, \mathbf{F}_{c_{\text{arm}}}^\top]^\top \in \mathbb{R}^{18}$ concatenates the linear contact forces at the four feet and, for the arm end-effector, the linear force and optionally the angular moment. The term $\boldsymbol{\tau}_j \in \mathbb{R}^{n_j}$ denotes the joint torques, and $\mathbf{J}_c(\mathbf{q}) \in \mathbb{R}^{18 \times (6+n_j)}$ is the contact Jacobian.

IV. MPC FORMULATION

In this section, we present the MPC formulation for locomanipulation using whole-body dynamics. The state vector $\mathbf{x} = [\mathbf{q}, \mathbf{v}]^\top$ consists of the generalized coordinates and velocities. The control input vector $\mathbf{u} = [\boldsymbol{\tau}_j, \mathbf{a}, \mathbf{F}_c]^\top$ includes joint torques, generalized accelerations, and contact forces.

A. Cost Function

The stage cost is defined as a quadratic function that penalizes deviations from a desired state and control input:

$$\ell(\mathbf{x}_k, \mathbf{u}_k) = \|\mathbf{x}_k - \mathbf{x}^{\text{des}}\|_{\mathbf{Q}}^2 + \|\mathbf{u}_k - \mathbf{u}^{\text{des}}\|_{\mathbf{R}}^2, \quad (3)$$

where $\mathbf{Q} \succeq 0$ and $\mathbf{R} \succ 0$ are weight matrices for states and inputs, respectively. The terminal cost is chosen to be equivalent to the stage cost. The desired state and inputs are:

- \mathbf{q}^{des} : the nominal robot configuration
- \mathbf{v}^{des} : commanded linear and angular velocity of the base, along with zero joint velocities for regularization
- $\mathbf{a}^{\text{des}}, \boldsymbol{\tau}_j^{\text{des}}$: zero for regularization
- $\mathbf{F}_c^{\text{des}}$: evenly distribute the weight over the stance feet

B. Constraints

1) *Dynamics Constraints*: To avoid defining quaternions as decision variables and enforcing their unit-norm constraint, we represent the generalized coordinates using a local increment $\delta \mathbf{q}_k \in \mathbb{R}^{6+n_j}$ in the tangent space of $SO(3)$. This increment is defined relative to the initial measured configuration $\hat{\mathbf{q}}$, and the absolute configuration is recovered via manifold integration, $\mathbf{q}_k = \hat{\mathbf{q}} \oplus \delta \mathbf{q}_k$. The state dynamics are discretized using explicit Euler integration with a time step δt_k :

$$\begin{aligned} \delta \mathbf{q}_{k+1} &= \delta \mathbf{q}_k + \mathbf{v}_k \delta t_k, \\ \mathbf{v}_{k+1} &= \mathbf{v}_k + \mathbf{a}_k \delta t_k. \end{aligned} \quad (4)$$

The state dynamics in Eq. 4 represent a purely kinematic formulation. To ensure consistency with the whole-body dynamics, we impose the equations of motion (Eq. 2) as path constraints at each time step k :

$$\begin{bmatrix} \mathbf{0}_{6 \times 1} \\ \boldsymbol{\tau}_{j,k} \end{bmatrix} = \mathbf{f}_{\text{RNEA}}(\mathbf{q}_k, \mathbf{v}_k, \mathbf{a}_k, \mathbf{F}_{c,k}). \quad (5)$$

Here, \mathbf{f}_{RNEA} computes the inverse dynamics using the Recursive Newton-Euler Algorithm (RNEA) [26]. Prior work has shown that direct shooting trajectory optimization problems, such as ours, benefit from using inverse dynamics rather than forward dynamics, resulting in faster solve times and greater robustness to coarse discretization [27]. In Section VI-A, we validate this finding by benchmarking our inverse dynamics formulation against both forward dynamics and reduced-order centroidal dynamics.

2) *Contact and Swing Constraints*: The MPC receives a fixed gait schedule as an input, based on which the following contact and swing constraints are formulated:

$$\begin{aligned} \text{Contact: } & \begin{cases} \mathbf{F}_{c_i,z} \geq 0, & \mathbf{v}_{c_i} = \mathbf{0} \\ \mu^2 \mathbf{F}_{c_i,z}^2 \geq \mathbf{F}_{c_i,x}^2 + \mathbf{F}_{c_i,y}^2. \end{cases} \\ \text{Swing: } & \mathbf{F}_{c_i} = \mathbf{0}, \quad \mathbf{v}_{c_i,z} = \mathbf{v}_z^{\text{ref}}. \end{aligned} \quad (6)$$

The contact constraints ensure that the normal force $\mathbf{F}_{c_i,z}$ for foot i is positive and inside the friction cone, with friction coefficient μ . Furthermore, the linear velocity \mathbf{v}_{c_i} of foot i must be zero, ensuring the stance leg does not separate or slip with respect to the ground.

The swing constraints ensure that no external force acts on the swing feet and that the foot velocity tracks a reference trajectory in the vertical direction. This reference $\mathbf{v}_z^{\text{ref}}$ is parametrized by a cubic spline. In horizontal directions, the foot velocities are left unconstrained, allowing the MPC to optimize foot trajectories and step locations.

3) *Arm Constraints*: For the arm task, the desired end-effector force and velocity are constrained to track a user-defined command for all time steps k :

$$\mathbf{F}_{c_{\text{arm}}} = \mathbf{F}_{c_{\text{arm}}}^{\text{des}}, \quad \mathbf{v}_{c_{\text{arm}}} = \mathbf{v}_{c_{\text{arm}}}^{\text{des}}. \quad (7)$$

This formulation is suitable for tasks such as pulling a heavy object with a specified force and velocity. However, it does not explicitly model the object or its dynamics, and therefore does not enable reasoning about how the object will respond to the applied forces.

4) *State and Input Bounds*: Joint positions and velocities are bound to their operational limits for all time steps k , while torque limits are only enforced for the first 2 steps:

$$\begin{aligned} \mathbf{q}_{j,\min} &\leq \mathbf{q}_{j,k} \leq \mathbf{q}_{j,\max} & \forall k, \\ \mathbf{v}_{j,\min} &\leq \mathbf{v}_{j,k} \leq \mathbf{v}_{j,\max} & \forall k, \\ \boldsymbol{\tau}_{j,\min} &\leq \boldsymbol{\tau}_{j,k} \leq \boldsymbol{\tau}_{j,\max} & \forall k \leq 2. \end{aligned} \quad (8)$$

Since the torques applied to the robot are interpolated only up to $k = 2$ (see Section V-B), we remove torque limits beyond this node while ensuring safe hardware execution. This allows us to completely remove the torques from the decision variables for the rest of the horizon, while still guaranteeing dynamic feasibility through the first six rows of the RNEA constraint (Eq. 5).

C. Adaptive Time Steps

Accurate whole-body control requires high temporal resolution to capture fast dynamics, while the MPC must also look far enough ahead to plan footstep and joint trajectories effectively. Using uniformly small time steps across the entire horizon does not scale well computationally, as it requires a large number of nodes to span longer durations. To address this trade-off, previous work proposes a cascaded-fidelity MPC, using coarser time steps later in the horizon together with a simplified SRB dynamics model [28]. Other work on general MPC introduces an adaptive time step strategy that becomes exponentially more sparse throughout the horizon [29].

Inspired by these approaches, we adopt a geometric time step scheme of the form: $\delta t_k = \gamma^k \delta t_0$, with $\gamma > 1$. This provides fine resolution early in the horizon for accurate control, while reducing the number of nodes for long-term planning. The impact of this strategy compared to a uniform grid is evaluated in simulation (see Section VI-B.2).

V. IMPLEMENTATION DETAILS

We implement the MPC method using symbolic expressions of the system dynamics provided by the *CasADi* [2] interface available in *Pinocchio* [1]. *CasADi* is additionally used to perform automatic differentiation and interface with numerical solvers. In this section, we provide implementation details of this integration and additional information related to the full deployment pipeline.

A. Solution Method

1) *Interior-Point Fatrop Solver*: We demonstrate that real-time whole-body MPC can be achieved using the interior-point (IP) solver *Fatrop* [3], which solves the full nonlinear program (NLP) by leveraging the block-sparse structure of stage-wise constraints. This is enabled by a structure-exploiting linear solver based on Riccati recursion [30]. In our experiments, *Fatrop* achieves over a $10\times$ speedup compared to the more standard IP solver *IPOPT* [31], which does not explicitly exploit the problem structure. Its integration with *CasADi* makes it easily applicable in our MPC pipeline. The main implementation requirement is that states and inputs must be interleaved in the decision variable vector to expose the stage-wise structure.

Most prior work on MPC for locomotion and loco-manipulation relies on sequential quadratic programming (SQP) [23, 13, 16], which solves a sequence of quadratic subproblems obtained from successive linearizations of the nonlinear constraints. In contrast, *Fatrop* allows us to solve the full nonlinear problem directly using second-order information from the exact Hessian. This approach achieves high accuracy, strict constraint satisfaction, and robust convergence with minimal parameter tuning, requiring as low as three iterations per solve.

2) *Warm-Starting*: Warm-starting solvers is generally critical for achieving real-time performance during hardware deployment. *Fatrop* offers the option to be warm-started, which resulted in a 15% to 20% reduction in solve time in our experiments. Each MPC iteration is warm-started using the solution from the previous step. However, the use of adaptive time steps introduces a slight time misalignment between successive horizons, which particularly affects the contact force variables due to mode switches between stance and swing phases. To mitigate this, we reinitialize the contact force guesses by distributing the robot's weight evenly across stance feet and setting them to zero for swing feet.

3) *Code Generation*: Using *Pinocchio* [1] and *CasADi* [2], the MPC can be implemented in Python and compiled into a shared library, which improves portability and facilitates fast execution. In our experiments, code generation resulted in a speed-up factor of 2 to 2.5. The compiled MPC can be directly evaluated for the *Fatrop* solver, given the respective parameters and warm-started decision variables. This enables straightforward hardware deployment, with the flexibility to tune parameters such as the state and input weights \mathbf{Q} and \mathbf{R} in real-time without recompiling the MPC.

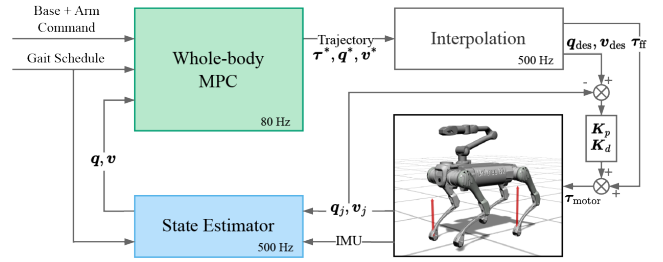


Fig. 2: The pipeline of whole-body inverse dynamics MPC.

B. Pipeline Description

Fig. 2 displays the complete pipeline for deployment on the *Unitree B2* quadruped, equipped with a *Z1* manipulator. The *B2* robot weighs approximately 70 kg, including the battery, and has three motors per leg, totaling 12 actuated joints. The *Z1* robot weighs 4.5 kg and has 6 actuated joints. Since our experiments focus on linear arm tracking tasks, we lock the last 2 joints of the arm, resulting in a whole-body model with 22 degrees of freedom (6 base, 12 legs, 4 arm).

The MPC runs at 80 Hz on a PC equipped with an AMD Ryzen 9 9950X (16 cores, 4.3 GHz). As inputs, it receives:

- 1) **Tracking commands**: Targets for the base velocity (linear and angular), and arm end-effector velocity and force (linear).
- 2) **Gait schedule**: Fixed contact schedule over the predicted horizon. The actual footstep locations are optimized by the MPC.
- 3) **Initial state**: Generalized coordinate positions and velocities from the state estimator.

The MPC outputs desired joint position, velocity, and torque trajectories. These are interpolated linearly at 500 Hz:

$$\begin{aligned} \tau_{\text{ff}}(t) &= \tau_0^* + \frac{\tau_1^* - \tau_0^*}{\delta t_0}(t - t_0), \\ \mathbf{q}_{\text{des}}(t) &= \mathbf{q}_0^* + \mathbf{v}_0^*(t - t_0), \\ \mathbf{v}_{\text{des}}(t) &= \mathbf{v}_0^* + \mathbf{a}_0^*(t - t_0). \end{aligned} \quad (9)$$

If t_1 has passed and we have not yet received a new solution from the MPC module, we interpolate the last received solution analogously between t_1 and t_2 . The interpolated commands are then fed into a low-level PD controller to compute the desired motor torque:

$$\tau_{\text{motor}} = \tau_{\text{ff}} + \mathbf{K}_p(\mathbf{q}_{\text{des}} - \mathbf{q}) + \mathbf{K}_d(\mathbf{v}_{\text{des}} - \mathbf{v}). \quad (10)$$

This joint-level feedback is only required for hardware deployment and is standard practice to compensate for actuator limitations and unmodeled dynamics, particularly in systems with quasi-direct drive actuators.

Finally, IMU and joint encoder readings are processed by a state estimator to update the initial state of the MPC. As the *B2* robot does not provide contact sensing, the estimator infers foot contact states from the planned gait schedule, following a two-stage Kalman filter approach [34]. Furthermore, since the joint encoder velocities seemed to be noisy on hardware, an additional low-pass filter is added at the input to the MPC, with a cutoff frequency of 50 Hz.

| Model | Related Work | States | Inputs | Dynamics Constraint | Path Constr. | Instr. per Constr. | Dec. Vars. | Solve Time [ms] | |
|-----------------------------|--------------|-----------------------|--------------------|---|--------------|--------------------|------------|-----------------|------------|
| | | | | | | | | Pre-comp. | Post-comp. |
| Whole-Body Inv. Dyn. (Ours) | [23–25] | $[q, v]$ | $[\tau_j, a, F_c]$ | $\begin{bmatrix} 0_{6 \times 1} \\ \tau_j \end{bmatrix} = M\alpha + \mathbf{b} - \mathbf{J}_c^\top F_c$ | ✓ | 3818 | 1226 | 29.0 (0.9) | 12.5 (0.5) |
| Whole-Body Forw. Dyn. | [19, 32] | $[q, v]$ | $[\tau_j, F_c]$ | $\alpha = \mathbf{M}^{-1} \left(\begin{bmatrix} 0_{6 \times 1} \\ \tau_j \end{bmatrix} + \mathbf{J}_c^\top F_c - \mathbf{b} \right)$ | ✗ | 7764 | 1094 | 81.6 (9.2) | 34.2 (3.8) |
| Centroidal Velocity | [11] | $[h_{\text{com}}, q]$ | $[v, F_c]$ | $h_{\text{com}} = \mathbf{A}v = [\mathbf{A}_b \ \mathbf{A}_j] \begin{bmatrix} v_b \\ v_j \end{bmatrix}$ | ✓ | 4070 | 938 | 33.0 (1.9) | 16.0 (0.9) |
| | [15–18] | $[h_{\text{com}}, q]$ | $[v_j, F_c]$ | $v_b = \mathbf{A}_b^{-1} (h_{\text{com}} - \mathbf{A}_j v_j)$ | ✗ | 4727 | 854 | 115.5 (14.5) | — * |
| Centroidal Acceleration | — | $[q, v]$ | $[a, F_c]$ | $\dot{h}_{\text{com}} = \dot{\mathbf{A}}v + [\mathbf{A}_b \ \mathbf{A}_j] \begin{bmatrix} a_b \\ a_j \end{bmatrix}$ | ✓ | 15628 | 1178 | 103.7 (3.9) | — * |
| | [14, 33] | $[q, v]$ | $[a_j, F_c]$ | $a_b = \mathbf{A}_b^{-1} (\dot{h}_{\text{com}} - \dot{\mathbf{A}}v - \mathbf{A}_j a_j)$ | ✗ | 16339 | 1094 | 189.8 (18.0) | — * |

TABLE I: A comparison of our whole-body inverse dynamics MPC formulation with forward dynamics and reduced-order centroidal models, all of which are available in our framework. We evaluate solve times before and after generating C code for the OCP and compiling it to a shared library for hardware transfer. (*) Indicates that compiling the OCP failed due to memory exhaustion, on a system with 32 GB RAM.

VI. RESULTS

We first benchmark our whole-body inverse dynamics formulation against other state-of-the-art modeling approaches, highlighting the computational efficiency of our method. Following this, we evaluate the precise and robust loco-manipulation behaviors of our MPC in simulation. Finally, we demonstrate its capabilities on hardware through a series of interactive loco-manipulation tasks.

A. Benchmarking Dynamics Models

In Table I, we compare our whole-body inverse dynamics MPC against forward dynamics, as well as reduced-order centroidal models, the latter representing the current state-of-the-art for loco-manipulation [15, 16]. All dynamics formulations are implemented in our proposed framework, allowing for systematic comparisons.

The centroidal methods model the evolution of the center of mass (CoM) linear and angular momentum $h_{\text{com}} \in \mathbb{R}^6$. They describe the relationship of the CoM with the kinematic configuration of the robot through the centroidal momentum matrix (CMM) $\mathbf{A}(q)$. For both velocity- and acceleration-level centroidal dynamics, we can add the resulting CMM constraint as a path constraint, or by propagating it through the state transition path function $f(\cdot)$ (see Table I). The latter case is employed if we remove the base velocity/acceleration from the input, which is the more common choice in practice since the base is unactuated.

We compare the solve times for all dynamics formulations using the *B2+Zl* system. The MPC is evaluated in closed loop using Pinocchio’s forward dynamics without a high-fidelity physics simulator. It contains 15 nodes with adaptive time steps, covering a 0.56 s horizon (70 % of the gait period). We found that solve times for *Fatrop* scale approximately linearly with the number of nodes, and this choice provided a good balance between accuracy and computational speed. To match the constraints of the whole-body MPC, torque limits are applied to the centroidal formulations for $k \leq 2$, by estimating them through the contact Jacobian and gravity compensation.

The results in Table I show that our whole-body inverse dynamics MPC achieves the fastest solve times among all formulations, despite having the highest number of decision variables. It solves the nonlinear optimization problem to convergence in 12.5 ms post-compilation, enabling real-time control at 80 Hz. To help interpret this result, we evaluate the number of mathematical instructions in the *CasADi* expression for each constraint. We see that the inverse dynamics RNEA constraint has the lowest number of instructions, likely explaining the computational speed.

Furthermore, *Fatrop* consistently performs far better when dynamics are imposed as path constraints rather than embedded in the state transition function. Although both optimization formulations use direct multiple shooting, embedding the dynamics in $f(\cdot)$ effectively forward-propagates the input initial guess, which can drive the state trajectory farther from the optimum. In contrast, enforcing the dynamics as path constraints allows the solver to jointly optimize over states and inputs, resulting in faster convergence with lower variance (see Table I).

B. Simulation Experiments

We run our whole-body MPC for the *B2+Zl* model in a physics simulator based on the Open Dynamics Engine (ODE) [35]. We evaluate the controller’s performance with and without accounting for the computation time delay.

1) *Without Time Delay*: Without accounting for the computation time delay, our MPC achieves stable locomotion and end-effector tracking with pure feed-forward torques ($K_p = K_d = 0$). We perform several experiments to test the capabilities of the controller.

Fig. 3 illustrates adaptive whole-body behavior across varying manipulator targets. In this experiment, the robot trots forward at 0.2 m/s while tracking a reference trajectory with the arm end-effector. The trajectory is converted into velocity commands using position feedback, enabling highly accurate tracking as visible in Fig. 3. Furthermore, when reaching down low or up high, the MPC automatically adjusts the base pose and leg motions to maintain tracking accuracy for the manipulator.

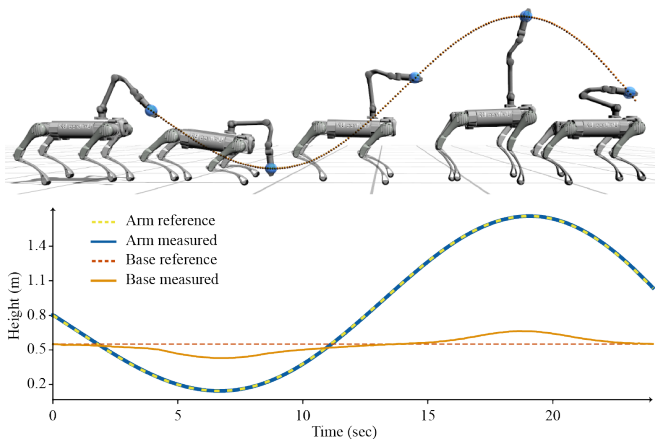


Fig. 3: Whole-body adaptability: The MPC generates coordinated full-body motions to track an arm reference trajectory while trotting, automatically adjusting the base pose to reach both low and high targets.

This behavior emerges from the cost design: base deviations are penalized softly, while the end-effector velocity is enforced as a hard constraint. These results demonstrate the ability of the whole-body MPC to achieve precise arm tracking by optimizing base and joint motions in coordination.

Fig. 4 demonstrates the second simulation experiment, in which a velocity disturbance of 0.8m/s is applied to the robot’s base while trotting in place. The MPC successfully stabilizes this disturbance while maintaining high end-effector tracking accuracy. Additionally, the joint torques always stay within the operational limits, once even pushing the rear left calf joint to its 320Nm limit. This illustrates the benefit of our torque-level MPC formulation in handling whole-body disturbances while enforcing safety constraints.

Finally, we observe that the controller’s response varies depending on when in the gait cycle the disturbance occurs. Due to the robot’s mass distribution, the rear leg joints experience higher loads than the front legs, making rear foot placement critical for disturbance recovery. At the first disturbance, the rear left foot is the next to step and requires minimal position adjustment, albeit pushing the calf torque to its 320Nm limit. In contrast, during the second disturbance, the rear right foot must significantly modify its step location to prevent the base from drifting further left. This experiment highlights the MPC’s ability to reason jointly about torques, footstep adaptation, and whole-body motion in response to external perturbations.

2) *With Time Delay:* With the computation time delay accounted for in the simulator, our controller requires damping of about $K_d = 5$ for stable walking and arm end-effector tracking. In this setting, we test the adaptive time step strategy described in Section IV-C. We run the MPC with 15 nodes and adaptive steps ranging from 10ms to 100ms, totaling a horizon of 0.56s. Comparing this with fixed time steps of 40ms, resulting in the same horizon length, shows a significant reduction in drift of the base while trotting: 0.4cm/s drift for adaptive time steps vs. 1.2cm/s drift for fixed time steps.

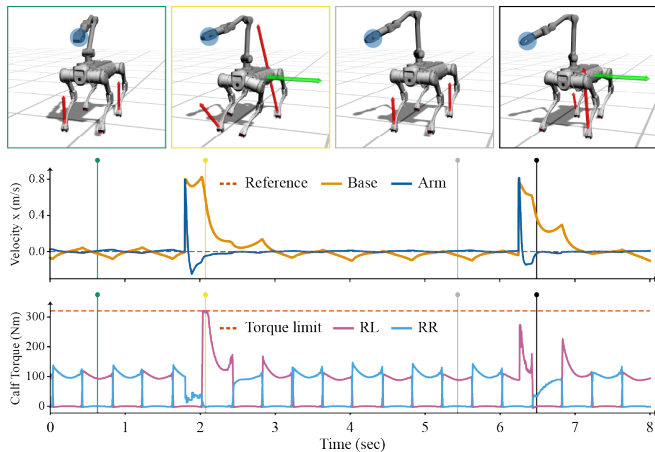


Fig. 4: Disturbance rejection: The base velocity is perturbed laterally by 0.8m/s while trotting in place. The arm end-effector and base lateral velocities are visualized, along with the rear leg calf torques (RL and RR).

C. Hardware Experiments

We deploy the MPC pipeline with the compiled *Fatrop* solver on hardware. The controller runs at 80 Hz off-board, on the same processor as for the simulation experiments, and the control signals are sent over Ethernet to *B2* and *Z1*. *B2* supports *ROS2* for low-level communication, whereas *Z1* provides a UDP-based Software Development Kit.

Unlike in simulation, where the controller performed well with pure feed-forward torques, hardware deployment required a velocity feedback gain of around $K_d = 20$ to avoid high-frequency motor oscillations. Adding the position feedback gain K_p was not necessary for the quadruped, demonstrating the effectiveness of the MPC torque solution. For the manipulator, noisy joint sensor readings sometimes caused the end-effector position to deviate, which was mitigated by adding a small proportional term for the arm joints.

We perform several whole-body loco-manipulation tasks on hardware (Fig. 1 and Fig. 5). To test the MPC near the robot’s operational limits, we demonstrate pulling a 10 kg payload in stance and while walking. This is a significant increase over the *Z1* arm’s rated maximum payload of 3 kg to 5 kg. Despite simulation showing successful pulling up to 14 kg, in a real scenario, overheating of the third arm joint prevents achieving higher loads.

Additional tasks include pushing a box against a wall while trotting in place (Fig. 1), and wiping a whiteboard while standing (Fig. 5). For these experiments, the end-effector force targets were manually specified. As we lacked the equipment to measure the actual applied forces, we could not verify whether the targets were precisely tracked. Nevertheless, the results demonstrate that our MPC can simultaneously handle end-effector force and motion targets on hardware. Finally, by reducing the arm’s PD gains and setting the end-effector target velocity and force to zero, we demonstrate compliant behavior for human interaction (Fig. 1). This experiment indicates the potential for safe human-robot collaboration in real-world loco-manipulation scenarios.

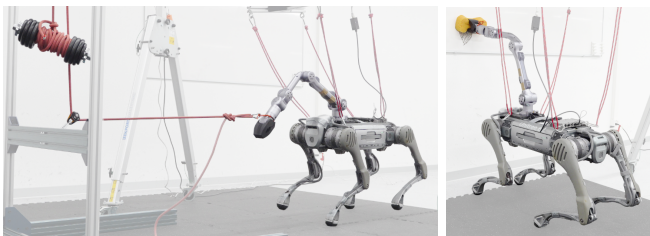


Fig. 5: Hardware demonstration of loco-manipulation tasks: Pulling a 10 kg load while walking (left) and accurately wiping a whiteboard (right).

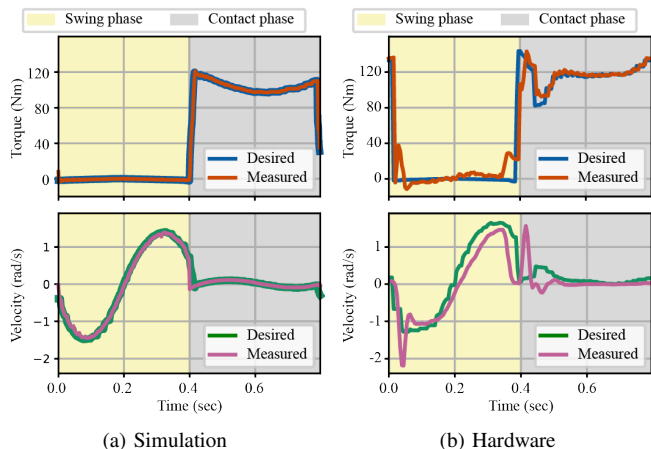


Fig. 6: Rear left calf joint torques and velocities during trot

D. Sim-to-Real Discrepancies

Despite the successful hardware deployment, the experiments revealed several sim-to-real mismatches that affect real-world performance. Most notably, solve times showed greater variability on hardware, occasionally reaching up to 20 ms compared to an average of 12.5 ms in simulation. This may stem from computation delays, causing the MPC prediction to deviate from the true system state, or from model discrepancies.

To investigate the sim-to-real mismatch, we compare joint-level data from the rear-left calf on hardware and in simulation (Fig. 6). Notable differences appear in both torque and velocity profiles, particularly during contact transitions. At the start of the contact phase, the calf torque spikes to support the robot’s weight. In simulation, this transition is stable for both torque and velocity. On hardware, however, a spike in joint velocity indicates delayed ground contact, prompting the MPC to reduce torque until contact is established. This contact mode mismatch is a well-known challenge in model-based control of legged robots.

Beyond contact transitions, other discrepancies also affect performance. First, the dynamics model does not account for motor friction and rotor inertia, which degrade torque tracking on hardware. Prior work addresses this by incorporating these effects directly into the MPC model [23] or compensating via feed-forward torque terms [21, 22]. Applying similar techniques to the *B2* and *Z1* platforms, based on system identification [36], could improve tracking accuracy. Second, we observed that the robot’s mass distribution deviates from the URDF: While the model places the base center of mass within 1 cm of its geometric center,

hardware measurements indicated a rearward weight shift of about 60%. Updating the CoM offset and contact force targets in the MPC significantly reduced backward lean on hardware, and all results reported here were obtained with this modification. This finding suggests that more advanced system identification methods [37] could further improve model accuracy and controller performance.

VII. CONCLUSION AND FUTURE WORK

We present a real-time whole-body MPC framework for loco-manipulation on the *Unitree B2* quadruped equipped with a *Z1* manipulator. Our method directly optimizes joint torques by enforcing whole-body inverse dynamics as path constraints on each node. Using the interior-point solver *Fatrop*, we successfully transfer the MPC to hardware, running the optimization at 80 Hz and interpolating the solution at 500 Hz to interface with a low-level PD controller. Crucially, no separate whole-body controller is required to track the MPC solution, resulting in a simple unified pipeline.

Future work should focus on extending the framework to model object-level dynamics for interaction [15, 16], allowing the MPC to optimize arm end-effector force and motion instead of having them as an external command. Additionally, collision constraints should be incorporated for both the quadruped and the manipulator to ensure safety. A thorough evaluation of alternative solution methods, such as sequential quadratic programming, is also planned, given its promising performance in real-time MPC for locomotion [13, 23]. Finally, to address the sim-to-real mismatches observed in our hardware experiments, system identification methods should be employed to improve low-level torque tracking and refine the robot’s inertial properties.

The sim-to-real limitations of MPC under model inaccuracies also motivate reinforcement learning (RL) as a complementary approach. Through domain randomization [38], RL has shown robustness to modeling errors in real-world deployments, and recent work demonstrates its potential for loco-manipulation [39, 40]. However, exploration in high-dimensional systems remains challenging, and learned policies typically operate at the position level—limiting their ability to plan both motion and force. Integrating RL with MPC is a promising direction for combining learning-based robustness with model-based planning and safety [41–43].

ACKNOWLEDGMENTS

The authors thank Charles Khazoom for discussing the whole-body MPC on the MIT humanoid [23], Jakob Genhart for implementing the *Z1* arm communication interface, and Ajay Sathya for his insights on warm-starting *Fatrop*.

REFERENCES

- [1] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, “The pinocchio c++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives,” in *2019 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2019, pp. 614–619.
- [2] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, “CasADi – A software framework for nonlinear optimization and optimal control,” *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

- [3] L. Vanroye, A. Sathya, J. De Schutter, and W. Decré, “Fatrop: A fast constrained optimal control problem solver for robot trajectory optimization and control,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 10036–10043.
- [4] J. Di Carlo, P. M. Wensing, B. Katz, G. Bleedt, and S. Kim, “Dynamic locomotion in the mit cheetah 3 through convex model-predictive control,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.
- [5] D. Kim, J. Di Carlo, B. Katz, G. Bleedt, and S. Kim, “Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control,” *arXiv preprint arXiv:1909.06586*, 2019.
- [6] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, “Gait and trajectory optimization for legged systems through phase-based end-effector parameterization,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.
- [7] S. Hong, J.-H. Kim, and H.-W. Park, “Real-time constrained nonlinear model predictive control on so (3) for dynamic legged locomotion,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 3982–3989.
- [8] D. Kim, J. Lee, J. Ahn, O. Campbell, H. Hwang, and L. Sentis, “Computationally-robust and efficient prioritized whole-body controller with contact constraints,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–8.
- [9] J. Li and Q. Nguyen, “Kinodynamics-based pose optimization for humanoid loco-manipulation,” *arXiv preprint arXiv:2303.04985*, 2023.
- [10] A. Rigo, M. Hu, S. K. Gupta, and Q. Nguyen, “Hierarchical optimization-based control for whole-body loco-manipulation of heavy objects,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 15 322–15 328.
- [11] H. Dai, A. Valenzuela, and R. Tedrake, “Whole-body motion planning with centroidal dynamics and full kinematics,” in *IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 295–302.
- [12] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, “Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2261–2268, 2018.
- [13] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, “Perceptive locomotion through nonlinear model-predictive control,” *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3402–3421, 2023.
- [14] E. Dantec, W. Jallet, and J. Carpentier, “From centroidal to whole-body models for legged locomotion: a comparative analysis,” in *2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids)*, 2024, pp. 335–342.
- [15] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, “A unified mpc framework for whole-body dynamic locomotion and manipulation,” *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4688–4695, 2021.
- [16] J.-P. Sleiman, F. Farshidian, and M. Hutter, “Versatile multicontact planning and control for legged loco-manipulation,” *Science Robotics*, vol. 8, no. 81, p. eadg5014, 2023.
- [17] J. Cheng, F. Abi-Farraj, F. Farshidian, and M. Hutter, “Haptic teleoperation of high-dimensional robotic systems using a feedback mpc framework,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 6197–6204.
- [18] I. Dadiotis, A. Laurenzi, and N. Tsagarakis, “Whole-body mpc for highly redundant legged manipulators: Experimental evaluation with a 37 dof dual-arm quadruped,” in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, 2023, pp. 1–8.
- [19] M. Neunert, M. Stäuble, M. Gifthalder, 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, 2018.
- [20] A. Assirelli, F. Risbourg, G. Lunardi, T. Flayols, and N. Mansard, “Whole-Body MPC without Foot References for the Locomotion of an Impedance-Controlled Robot,” 2022, working paper or preprint.
- [21] E. Dantec, R. Budhiraja, A. Roig, T. Lembono, G. Saurel, O. Stasse, P. Fernbach, S. Tonneau, S. Vijayakumar, S. Calinon, M. Taix, and N. Mansard, “Whole body model predictive control with a memory of motion: Experiments on a torque-controlled talos,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 8202–8208.
- [22] E. Dantec, M. Naveau, P. Fernbach, N. Villa, G. Saurel, O. Stasse, M. Taix, and N. Mansard, “Whole-body model predictive control for biped locomotion on a torque-controlled humanoid robot,” in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 2022, pp. 638–644.
- [23] C. Khazoom, S. Hong, M. Chignoli, E. Stanger-Jones, and S. Kim, “Tailoring solution accuracy for fast whole-body model predictive control of legged robots,” *IEEE Robotics and Automation Letters*, vol. 9, no. 12, pp. 11 074–11 081, 2024.
- [24] C. Mastalli, S. P. Chhatoi, T. Corbères, S. Tonneau, and S. Vijayakumar, “Inverse-dynamics mpc via nullspace resolution,” *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 3222–3241, 2023.
- [25] H. Ferrolho, V. Ivan, W. Merkt, I. Havoutis, and S. Vijayakumar, “Roloma: Robust loco-manipulation for quadruped robots with arms,” *Autonomous Robots*, vol. 47, no. 8, pp. 1463–1481, 2023.
- [26] R. Featherstone, *Rigid Body Dynamics Algorithms*. Berlin, Heidelberg: Springer-Verlag, 2007.
- [27] H. Ferrolho, V. Ivan, W. Merkt, I. Havoutis, and S. Vijayakumar, “Inverse dynamics vs. forward dynamics in direct transcription formulations for trajectory optimization,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 12 752–12 758.
- [28] H. Li and P. M. Wensing, “Cafe-mpc: A cascaded-fidelity model predictive control framework with tuning-free whole-body control,” *IEEE Transactions on Robotics*, 2024.
- [29] S. Shin and V. M. Zavala, “Diffusing-horizon model predictive control,” *IEEE Transactions on Automatic Control*, vol. 68, no. 1, pp. 188–201, 2021.
- [30] L. Vanroye, J. De Schutter, and W. Decré, “A generalization of the riccati recursion for equality-constrained linear quadratic optimal control,” *Optimal Control Applications and Methods*, vol. 45, no. 1, pp. 436–454, 2024.
- [31] A. Wächter and L. T. Biegler, “On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming,” *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.
- [32] M. V. Minniti, F. Farshidian, R. Grandia, and M. Hutter, “Whole-body mpc for a dynamically stable mobile manipulator,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3687–3694, 2019.
- [33] W. Jallet, A. Bambade, S. El Kazdadi, C. Justin, and M. Nicolas, “aligator.” [Online]. Available: <https://github.com/Simple-Robotics/aligator>
- [34] G. Bleedt, 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)*. IEEE, 2018, pp. 2245–2252.
- [35] R. Smith *et al.*, “Open dynamics engine,” <http://www.ode.org>, 2005, accessed: 2025-05-30.
- [36] L. Schwendeman, A. SaLoutos, E. Stanger-Jones, and S. Kim, “Improving domain transfer of robot dynamics models with geometric system identification and learned friction compensation,” in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*. IEEE, 2023, pp. 1–8.
- [37] T. D. Nguyen, V. Bonnet, M. Sabbah, M. Gautier, P. Fernbach, and F. Lamiroux, “Figaroh: a python toolbox for dynamic identification and geometric calibration of robots and humans,” in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*. IEEE, 2023, pp. 1–8.
- [38] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. 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)*, 2017, pp. 23–30.
- [39] T. Portela, G. B. Margolis, Y. Ji, and P. Agrawal, “Learning force control for legged manipulation,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 15 366–15 372.
- [40] M. Zhang, Y. Ma, T. Miki, and M. Hutter, “Learning to open and traverse doors with a legged manipulator,” in *Proceedings of The 8th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 270. PMLR, 2025, pp. 2913–2927.
- [41] D. Kang, J. Cheng, M. Zamora, F. Zargarbashi, and S. Coros, “RI+ model-based control: Using on-demand optimal control to learn versatile legged locomotion,” *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6619–6626, 2023.
- [42] F. Jenelten, J. He, F. Farshidian, and M. Hutter, “Dtc: Deep tracking control,” *Science Robotics*, vol. 9, no. 86, p. eadh5401, 2024.
- [43] J. Cheng, D. Kang, G. Fadini, G. Shi, and S. Coros, “Rambo: RI-augmented model-based optimal control for whole-body loco-manipulation,” *arXiv preprint arXiv:2504.06662*, 2025.