IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

RAMBO: RL-Augmented Model-Based Whole-Body Control for Loco-Manipulation

Jin Cheng[α], Dongho Kang[α], Gabriele Fadini[α], Guanya Shi[β], and Stelian Coros[α]

***Abstract*— Loco-manipulation, physical interaction of various objects that is concurrently coordinated with locomotion, remains a major challenge for legged robots due to the need for both precise end-effector control and robustness to unmodeled dynamics. While model-based controllers provide precise planning via online optimization, they are limited by model inaccuracies. In contrast, learning-based methods offer robustness, but they struggle with precise modulation of interaction forces. We introduce RAMBO, a hybrid framework that integrates model-based whole-body control within a feedback policy trained with reinforcement learning. The model-based module generates feedforward torques by solving a quadratic program, while the policy provides feedback corrective terms to enhance robustness. We validate our framework on a quadruped robot across a diverse set of real-world loco-manipulation tasks, such as pushing a shopping cart, balancing a plate, and holding soft objects, in both quadrupedal and bipedal walking. Our experiments demonstrate that RAMBO enables precise manipulation capabilities while achieving robust and dynamic locomotion.**

I. INTRODUCTION

Modern legged robots have demonstrated impressive mobility over a wide range of terrains [1, 2, 3, 4]. To expand their capabilities beyond conventional locomotion tasks, there is growing interest in loco-manipulation, which enables these machines to actively interact with and manipulate their surroundings. However, whole-body loco-manipulation remains a challenging task for these systems, as it requires coordinated control of both the base and end-effector movements to achieve precise and robust behaviors, which often pose conflicting objectives [5].

Being robust against unmodeled effects and unexpected interactions, reinforcement learning (RL)-based controllers have achieved impressive results in various pedipulation and manipulation tasks [6, 7, 8]. However, training agents for loco-manipulation via RL still remains a challenging task. Often working in joint position space, learned policies tend to produce excessively large targets, which indirectly govern the resulting interaction forces [9, 10]. In practice, this strategy prioritizes robustness at the expense of precision [11, 12]. Moreover, loco-manipulation tasks typically require exploration in a large joint space, necessitating motion prior, reward shaping, or other exploration strategies [13, 14, 15].

[α]The authors are with Computational Robotics Lab in the Department of Computer Science, ETH Zurich, Zurich, Switzerland. {jicheng, kangd, gfadini, scoros}@ethz.ch

[β]Guanya Shi is with the Robotics Institute and the School of Computer Science at Carnegie Mellon University, Pittsburgh, USA. guanyas@andrew.cmu.edu

Project website: https://jin-cheng.me/rambo.github.io/

Supplementary video: https://youtu.be/VdZxhLNG6wQ

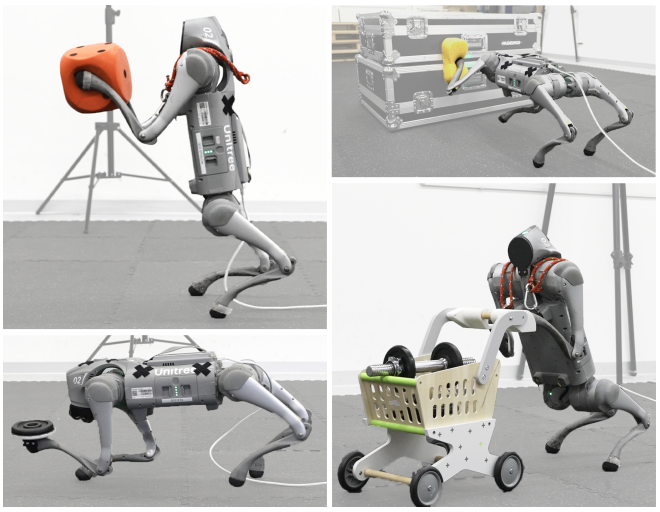

Fig. 1: Various whole-body loco-manipulation tasks enabled by RAMBO on Unitree Go2 [25] in both quadrupedal and bipedal modes.

Model-based control methods, on the other hand, have proven highly effective for contact planning and handling interactions with objects in whole-body loco-manipulation tasks [16, 17, 18]. By explicitly taking contact forces into account, these approaches enable precise control and optimization of torque-level commands [19, 20, 21]. However, their performance heavily depends on how system dynamics is modeled, and on careful parameter identification on real hardware [22]. In addition, model predictive control methods require a trade-off between model complexity, solution accuracy and planning horizon [23, 24].

The ultimate goal of this work is to equip the legged controllers with the capability to perform robust, precise, and efficient whole-body loco-manipulation. We aim to combine the strengths of model-based and learning-based approaches to achieve effective torque-level control while remaining robust against unmodeled effects and disturbances.

To this end, we propose RAMBO—RL-Augmented Model-Based WhOle-body Control—a hybrid control framework for whole-body loco-manipulation tasks on legged systems. Our method generates feedforward torques by optimizing end-effector contact forces through a model-based whole-body controller, formulated as a quadratic program (QP), while ensuring robustness with an RL policy that compensates for modeling errors through its corrective actions.

We demonstrate the effectiveness of our method on a range of loco-manipulation tasks, including pushing a shopping cart, balancing a plate, and holding soft objects—spanning both quadrupedal and bipedal dynamic walking on a

quadruped platform. Through extensive evaluations in simulated and real-world scenarios, RAMBO demonstrates a high degree of precision in tracking end-effector targets while remaining robust in typical locomotion tasks.

II. Related Work

In this section, we review two major groups of the state-of-the-art control approaches for whole-body loco-manipulation applications: model-based and learning-based methods.

A. Model-based Methods for Loco-manipulation

Model-based methods generate control signals by solving optimal control problems using first-principles dynamics models that describe the relationship between system states and control inputs [22]. By explicitly accounting for interaction forces, these methods can effectively optimize motion while incorporating the dynamic effects from contacts into the control loop [20, 26]. When applied to loco-manipulation tasks, model-based approaches offer precise control over the forces exerted on objects and produce effective torque-level commands for each joint [17, 27]. Among these, Model Predictive Control (MPC) is especially popular for legged systems, as it provides a robust feedback mechanism and enables emergent behaviors by forecasting the impact of control inputs over a time horizon [18, 28]. However, the applicability of model-based methods can be limited in complex scenarios, such as locomotion over uneven terrain or manipulation of objects with involved inertial properties, where accurate knowledge of the environment is difficult to obtain and the required modeling and computational effort becomes excessively large [18, 29].

State-of-the-art MPC methods for legged robots often resort to hierarchical frameworks where a middle-layer whole-body controller (WBC) is formulated to translate the high-level plans to torque-level commands through optimization in an instantaneous manner [29, 30]. RAMBO aims to preserve the benefits of torque-level control generated by model-based methods while overcoming their limitations. As such, our framework retains a computationally efficient whole-body control module [31] as a middle layer. The WBC computes feedforward torques by explicitly optimizing reaction forces, and it is augmented with a policy trained via RL to compensate for modeling inaccuracies and unplanned disturbances.

B. Learning-based Methods for Loco-manipulation

Significant progress has been made in training learning-based locomotion policies within physics-based simulators using RL [32, 10]. By leveraging techniques such as domain randomization [33], policies trained in an end-to-end scheme have demonstrated robust performance in a range of loco-manipulation scenarios, including soccer dribbling [7], object transferring [6, 34] and other pedipulation tasks [8, 11].

Despite these successes, RL-based approaches still face several notable limitations. First, the vast exploration space spanning diverse end-effector positions and object states renders the learning of meaningful manipulation behaviors sample-inefficient and reliant on complicated exploration strategies [12, 14, 35]. Second, most policies are trained in conjunction with low-level joint proportional–derivative (PD) controllers to translate positional command into desired torques. However, this structure is often exploited by RL policies that output excessively large position targets to generate sufficient contact forces [9], resulting in poor controllability of both end-effector positions and interaction forces [13, 36], which is critical for precise loco-manipulation. While torque-based policies may mitigate this, it requires high-frequency updates to prevent overshooting, further increasing the complexity of training [37, 38].

To address these challenges, recent methods have explored incorporating demonstrations to guide RL agents and improve the motion quality [15, 39, 40, 41], however, they still inherit the limitations by operating on joint positions. Drawing inspiration from prior work [42, 43], our framework integrates model-based WBC module to compute feedforward torque commands while complementing it with a learned policy to enhance robustness, resulting in an effective strategy for precise end-effector position and force control in loco-manipulation.

III. Preliminaries

In this section, we describe the preliminaries for better understanding the framework, including the dynamics model and foot trajectory generation strategy used in RAMBO.

A. Single Rigid Body Dynamics for Legged Robots

In our framework, we employ the single rigid body (SRB) dynamics to model the quadruped robot. It assumes the majority of the mass is concentrated in the base link of the robot, and all the limbs are massless and their inertial effects are negligible [31]. While more complex models such as centroidal model or whole-body model [21, 28] can be in principle leveraged, they still inherit the limitations such as model inaccuracy and sensitivity to disturbances. We choose SRB to trade off complexity for computational efficiency.

As shown in Fig. 2, the state of the single rigid body can be described by

$$\boldsymbol{x} \coloneqq [\boldsymbol{p}\ \boldsymbol{v}\ \boldsymbol{R}\ {}_{\mathcal{B}}\boldsymbol{\omega}], \tag{1}$$

where $\boldsymbol{p} \in \mathbb{R}^3$ is the position of the body Center of Mass (CoM); $\boldsymbol{v} \in \mathbb{R}^3$ is the CoM velocity; $\boldsymbol{R} \in SO(3)$ is the rotation matrix of the body frame $\{\mathcal{B}\}$ expressed in the inertial world frame $\{\mathcal{W}\}$; $\det(\cdot)$ calculates the determinant of a matrix and $\mathbb{I}$ is the 3-by-3 identity matrix. ${}_{\mathcal{B}}\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity expressed in the body frame $\{\mathcal{B}\}$. Variables without subscript on the left are assumed to be in the inertial frame $\{\mathcal{W}\}$. Additionally, we define another coordinate frame, namely the *projected frame* $\{\mathcal{P}\}$, which is centered at the projection of CoM onto the ground plane, and whose x and z axes point forward and upward, respectively.

The input to the dynamics system is the external reaction forces $\boldsymbol{u}_i \in \mathbb{R}^3$ for the locations $\boldsymbol{p}_i \in \mathbb{R}^3$ in contact, where $i \in \{1, 2, .., N\} = \mathcal{N}$ denotes the index for contact locations, and N is the number of contacts, as shown in Fig. 2.

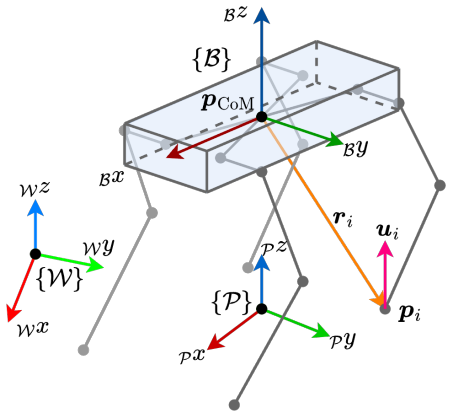

Fig. 2: Illustration of the 3D single rigid body model for a quadruped robot. $\{\mathcal{W}\}, \{\mathcal{B}\}, \{\mathcal{P}\}$ denote the inertial world frame, the body frame, and the projected frame, respectively. $\boldsymbol{r}_i$ is the contact position of limb i relative to the CoM. $\boldsymbol{u}_i$ is the external reaction force at the contact position.

The net external wrench $\boldsymbol{\mathcal{F}} \in \mathbb{R}^6$ exerted on the body is:

$$\boldsymbol{\mathcal{F}} = \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{\tau} \end{bmatrix} = \sum_{i=1}^{N} \begin{bmatrix} \mathbb{I} \\ [\boldsymbol{r}_i]_\times \end{bmatrix} \boldsymbol{u}_i, \tag{2}$$

where $\boldsymbol{F}$ and $\boldsymbol{\tau}$ are the total force and torque applied at the CoM; $\boldsymbol{r}_i = \boldsymbol{p}_i - \boldsymbol{p}$ is the contact positions relative to CoM; the $[\cdot]_\times : \mathbb{R}^3 \to \mathfrak{so}(3)$ operator converts the element from $\mathbb{R}^3$ to skew-symmetric matrices as $[\boldsymbol{a}]_\times \boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$ for all $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$. The inverse of $[\cdot]_\times$ is the vee map $[\cdot]^\vee : \mathfrak{so}(3) \to \mathbb{R}^3$.

The full dynamics of the rigid body can be formulated as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{v}} \\ \dot{\boldsymbol{R}} \\ {}_{\mathcal{B}}\dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v} \\ \frac{1}{M}\boldsymbol{F} + \boldsymbol{g} \\ \boldsymbol{R}[{}_{\mathcal{B}}\boldsymbol{\omega}]_\times \\ {}_{\mathcal{B}}\boldsymbol{I}^{-1}(\boldsymbol{R}^\top \boldsymbol{\tau} - [{}_{\mathcal{B}}\boldsymbol{\omega}]_\times {}_{\mathcal{B}}\boldsymbol{I}\, {}_{\mathcal{B}}\boldsymbol{\omega}) \end{bmatrix}, \tag{3}$$

where M is the mass of the rigid body; $\boldsymbol{g} \in \mathbb{R}^3$ is the gravitational acceleration vector; ${}_{\mathcal{B}}\boldsymbol{I}$ is the fixed moment of inertia in the body frame $\{\mathcal{B}\}$.

The dynamics of the linear and angular velocity can be further derived after ignoring the gyroscopic effect (assuming the angular velocity is small and its second-order term is negligible) and expressed in the body frame $\{\mathcal{B}\}$ as

$$\begin{bmatrix} {}_{\mathcal{B}}\dot{\boldsymbol{v}} \\ {}_{\mathcal{B}}\dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \frac{1}{M}\boldsymbol{R}^\top \boldsymbol{F} + \boldsymbol{R}^\top \boldsymbol{g} \\ {}_{\mathcal{B}}\boldsymbol{I}^{-1}\boldsymbol{R}^\top \boldsymbol{\tau} \end{bmatrix} = \sum_{i=1}^{N} \begin{bmatrix} \frac{1}{M} \\ {}_{\mathcal{B}}\boldsymbol{I}^{-1}[{}_{\mathcal{B}}\boldsymbol{r}_i]_\times \end{bmatrix} {}_{\mathcal{B}}\boldsymbol{u}_i + \hat{\boldsymbol{g}}, \tag{4}$$

where $\hat{\boldsymbol{g}} = [\boldsymbol{g}, 0_3]^\top \in \mathbb{R}^6$ is the total gravity for the base.

B. Foot Trajectory Generation Strategy

We use a heuristic strategy to generate symmetric foot trajectories for lifting and landing for each limb responsible for locomotion.

Given a specific gait pattern, one can interpolate three keyframe locations $({}_{\mathcal{P}}\boldsymbol{p}_{\text{lift}}, {}_{\mathcal{P}}\boldsymbol{p}_{\text{mid}}, {}_{\mathcal{P}}\boldsymbol{p}_{\text{land}})_i$ expressed in the projected frame $\{\mathcal{P}\}$ for each contact leg i based on the swing timing. The lifting location $({}_{\mathcal{P}}\boldsymbol{p}_{\text{lift}})_i$ is taken from the previous contact location when the leg i switches from contact to swing. The mid-air position $({}_{\mathcal{P}}\boldsymbol{p}_{\text{mid}})_i$ is located at the corresponding hip position $({}_{\mathcal{P}}\boldsymbol{p}_{\text{hip}})_i$ projected towards the ground in the projected frame $\{\mathcal{P}\}$ with some fixed desired foot height. The landing position$({}_{\mathcal{P}}\boldsymbol{p}_{\text{land}})_i$ is calculated according to the hip velocity in the projected frame $\{\mathcal{P}\}$ as

$$({}_{\mathcal{P}}\boldsymbol{p}_{\text{land}})_i = ({}_{\mathcal{P}}\boldsymbol{p}_{\text{hip}})_i + \frac{T_{\text{stance}}}{2}({}_{\mathcal{P}}\boldsymbol{v}_{\text{hip}})_i, \tag{5}$$

where T_{stance} is the stand duration. The desired foot trajectory is described using a cubic Bézier curve connecting the three keyframe positions.

IV. Method

As shown in Fig. 3, RAMBO is composed of three elements: motion reference generation, feedforward torque acquiring through WBC, and feedback policy with RL. We describe each component in detail below.

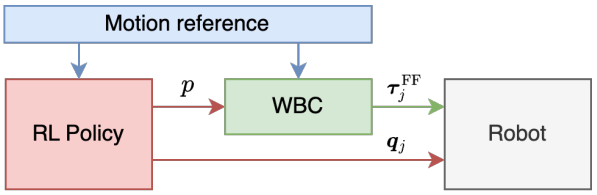

Fig. 3: Overview of the RAMBO architecture. It consists of three core components: (1) a motion reference generator, (2) a WBC module that computes feedforward joint torques $\boldsymbol{\tau}_j^{\text{FF}}$, and (3) an RL policy that generates feedback corrections to both WBC input parameters and motion reference.

A. Motion Reference Generation

For each control time step, RAMBO starts with querying a motion reference for both the base and joint $(\hat{\boldsymbol{q}}, \hat{\dot{\boldsymbol{q}}})$, where $\hat{\boldsymbol{q}}, \hat{\dot{\boldsymbol{q}}}$ are the desired generalized coordinates and velocities respectively; $\hat{\boldsymbol{q}} = [\hat{\boldsymbol{p}}\ \hat{\boldsymbol{R}}\ \hat{\boldsymbol{q}}_j]$ are the desired base position, orientation and joint positions; $\hat{\dot{\boldsymbol{q}}} = [\hat{\boldsymbol{v}}\ \hat{\boldsymbol{\omega}}\ \hat{\dot{\boldsymbol{q}}}_j]$ are the desired base linear and angular velocities, joint velocities. We use the subscript j to denote the joint dimensions.

To build a framework that allows the user to interactively control the robot in a versatile manner across diverse tasks, we implement a reference generation process from the user command, shown in Fig. 4. The input from the user includes the desired base velocity $({}_{\mathcal{P}}\hat{v}_x, {}_{\mathcal{P}}\hat{v}_y, {}_{\mathcal{P}}\hat{\omega}_z)$, which consists of the desired forward, side and turning velocities, all expressed in the projected frame $\{\mathcal{P}\}$. Defining the reference in the projected frame $\{\mathcal{P}\}$ allows RAMBO to operate with only proprioceptive information, without relying on an accurate estimation of coordinates in the inertial frame.

In addition, categorizing the end-effectors $i \in \mathcal{N}$ into two kinds: locomotion $\mathcal{L} \subset \mathcal{N}$, marked by superscript l, and manipulation $\mathcal{M} \subset \mathcal{N}$, marked by superscript m, our kinematic reference generation also accept specification for the desired EE positions ${}_{\mathcal{P}}\hat{\boldsymbol{p}}_i^m \in \mathbb{R}^3$ for the limbs responsible for manipulation. By incorporating predefined gait patterns and their contact schedules, ${}_{\mathcal{P}}\hat{\boldsymbol{p}}_i^l$ is generated by interpolating the keyframes $({}_{\mathcal{P}}\boldsymbol{p}_{\text{lift}}, {}_{\mathcal{P}}\boldsymbol{p}_{\text{mid}}, {}_{\mathcal{P}}\boldsymbol{p}_{\text{land}})_i$ as shown in Section III-B. The reference for each joint $\hat{\boldsymbol{q}}_j$ is calculated using inverse kinematics (IK)

$$\hat{\boldsymbol{q}}_j = \text{IK}({}_{\mathcal{P}}\boldsymbol{p}, {}_{\mathcal{P}}\boldsymbol{R}, {}_{\mathcal{P}}\hat{\boldsymbol{p}}_1, ..., {}_{\mathcal{P}}\hat{\boldsymbol{p}}_N). \tag{6}$$

For accounting the force command and the dynamics effect from manipulation tasks, the contact state for manipulation

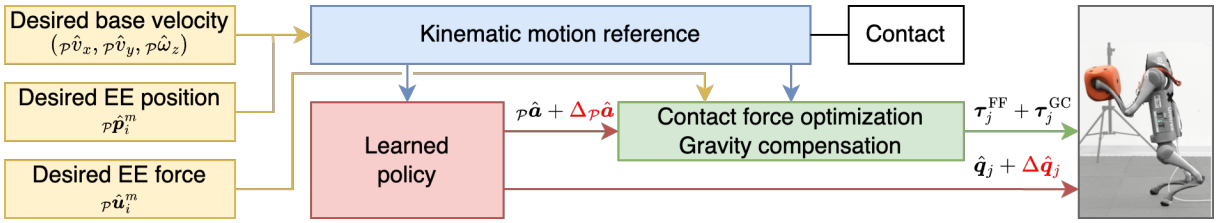

Fig. 4: Detailed architecture of the RAMBO control framework. The desired base velocity and EE positions are used to generate a kinematic motion reference, which is sent to the policy and whole-body control module. The whole-body control module also takes the desired EE force to compute the feedforward joint torques. The learned policy provides corrective feedback to the base acceleration and joint position targets, enabling robust control under modeling errors and dynamic disturbances.

$\hat{\boldsymbol{c}}^m = \{\hat{c}_i | i \in \mathcal{M}\}$ is always set to 1. The remaining contact state of locomotion end-effector $\hat{\boldsymbol{c}}^l = \{\hat{c}_i | i \in \mathcal{L}\}$ is predefined according to the gait pattern. We fill unspecified references with either current state or zeros

$$\begin{aligned} &_{\mathcal{P}}\hat{\boldsymbol{p}} = {}_{\mathcal{P}}\boldsymbol{p}, {}_{\mathcal{P}}\hat{\boldsymbol{R}} = {}_{\mathcal{P}}\boldsymbol{R} \\ &_{\mathcal{P}}\hat{\boldsymbol{v}} = [{}_{\mathcal{P}}\hat{v}_x, {}_{\mathcal{P}}\hat{v}_y, 0], {}_{\mathcal{P}}\hat{\boldsymbol{\omega}} = [0, 0, {}_{\mathcal{P}}\hat{\omega}_z], \dot{\hat{\boldsymbol{q}}}_j = \boldsymbol{0}. \end{aligned} \tag{7}$$

We note that RAMBO in principle allows any reference trajectory generation process including the ones using trajectory optimization or from offline motion library. The only requirement is to provide the contact state $\hat{c}_i \in \{0, 1\}$ for each end-effector $i \in \mathcal{N}$.

B. Generating Feedforward Torque via WBC

To account for contributions from all contacts and ensure the robot tracks the reference motion, we employ a computationally lightweight whole-body controller to optimize the reaction forces leveraging the single rigid body model.

We firstly calculate the base linear and angular acceleration target $_{\mathcal{P}}\hat{\boldsymbol{a}} = (_{\mathcal{P}}\hat{\boldsymbol{a}}_l, _{\mathcal{P}}\hat{\boldsymbol{a}}_a) \in \mathbb{R}^6$ using a proportional-derivative controller

$$_{\mathcal{P}}\hat{\boldsymbol{a}}_l = \kappa_{lp}(_{\mathcal{P}}\hat{\boldsymbol{p}} - {}_{\mathcal{P}}\boldsymbol{p}) + \kappa_{ld}(_{\mathcal{P}}\hat{\boldsymbol{v}} - {}_{\mathcal{P}}\boldsymbol{v}) \tag{8}$$

$$_{\mathcal{P}}\hat{\boldsymbol{a}}_a = \kappa_{ap}\log(_{\mathcal{P}}\hat{\boldsymbol{R}}^\top \cdot {}_{\mathcal{P}}\boldsymbol{R})^\vee + \kappa_{ad}(_{\mathcal{P}}\hat{\boldsymbol{\omega}} - {}_{\mathcal{P}}\boldsymbol{\omega}), \tag{9}$$

where $\kappa_{lp}, \kappa_{ld}, \kappa_{ap}, \kappa_{ad}$ are the linear and angular, proportional and derivative gains respectively; $\log(\cdot)^\vee : SO(3) \to \mathbb{R}^3$ converts a rotation matrix into an angle-axis representation, which is a vector in $\mathbb{R}^3$.

In addition to the desired acceleration $_{\mathcal{P}}\hat{\boldsymbol{a}}$, RAMBO also accept user's specification of desired EE forces $_{\mathcal{P}}\hat{\boldsymbol{u}}_i$ for the manipulation end-effectors $i \in \mathcal{M}$, as shown in Fig. 4. Using $_{\mathcal{B}}\hat{\boldsymbol{a}}$ and $_{\mathcal{B}}\hat{\boldsymbol{u}}_i$ in the base frame $\{\mathcal{B}\}$, we formulate the following QP to optimize the reaction force

$$\min_{{}_{\mathcal{B}}\boldsymbol{u}_i, i\in\mathcal{N}} \|\Delta_{\mathcal{B}}\boldsymbol{a}\|_{\boldsymbol{U}}^2 + \sum_{i\in\mathcal{M}} \|\Delta_{\mathcal{B}}\boldsymbol{u}_i\|_{\boldsymbol{V}}^2 + \sum_{i\in\mathcal{L}} \|{}_{\mathcal{B}}\boldsymbol{u}_i\|_{\boldsymbol{W}}^2 \tag{10a}$$

$$\text{subject to} : {}_{\mathcal{B}}\boldsymbol{a} = \sum_{i=1}^{N} \boldsymbol{A}_i {}_{\mathcal{B}}\boldsymbol{u}_i + \hat{\boldsymbol{g}} \tag{10b}$$

$$({}_{\mathcal{B}}\boldsymbol{u}_i)_z = 0, \qquad i \in \mathcal{L}_{\text{swing}} \tag{10c}$$

$$\|({}_{\mathcal{B}}\boldsymbol{u}_i)_x\| \le \mu({}_{\mathcal{B}}\boldsymbol{u}_i)_z, \qquad i \in \mathcal{L}_{\text{stance}} \tag{10d}$$

$$\|({}_{\mathcal{B}}\boldsymbol{u}_i)_y\| \le \mu({}_{\mathcal{B}}\boldsymbol{u}_i)_z, \qquad i \in \mathcal{L}_{\text{stance}} \tag{10e}$$

$$u_{z,\min} \le ({}_{\mathcal{B}}\boldsymbol{u}_i)_z \le u_{z,\max}, \qquad i \in \mathcal{L}_{\text{stance}} \tag{10f}$$

where $\Delta_{\mathcal{B}}\boldsymbol{a} = {}_{\mathcal{B}}\hat{\boldsymbol{a}} - {}_{\mathcal{B}}\boldsymbol{a}$, $\Delta_{\mathcal{B}}\boldsymbol{u}_i = {}_{\mathcal{B}}\hat{\boldsymbol{u}}_i - {}_{\mathcal{B}}\boldsymbol{u}_i$; $_{\mathcal{B}}\boldsymbol{a} = [{}_{\mathcal{B}}\dot{\boldsymbol{v}}, {}_{\mathcal{B}}\dot{\boldsymbol{\omega}}] \in \mathbb{R}^6$ is the acceleration of the single rigid body; $\boldsymbol{A}_i$ is the generalized inverse inertia matrix defined in Eq. 4; μ is the friction coefficient; $\mathcal{L}_{\text{swing}}, \mathcal{L}_{\text{stance}}$ are the set of end-effectors for locomotion in swing and stance respectively; $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W} \succ 0$ are positive definite weight matrices. We note that friction checking is only performed on locomotion end-effectors due to the uncertainty of contact surface for manipulation. Leveraging the SRB model, RAMBO efficiently accounts for the dynamic effects from contacts.

The feedforward joint torques $\boldsymbol{\tau}_j^{\text{FF}}$ are calculated using

$$\boldsymbol{\tau}_j^{\text{FF}} = \sum_{i=1}^{N} \boldsymbol{J}_i^\top \cdot {}_{\mathcal{B}}\boldsymbol{u}_i, \tag{11}$$

where $\boldsymbol{J}_i$ is the Jacobian corresponds to the end-effector i.

In addition to the feedforward torque from the reaction force optimization module, we calculate an additional torque term $\boldsymbol{\tau}_j^{\text{GC}}$ to compensate the gravity and account for the limb inertia, for each joint k

$$(\boldsymbol{\tau}_j^{\text{GC}})_k = -\sum_{l\in\mathcal{D}(k)} \boldsymbol{J}_{lk}^\top \cdot m_l \boldsymbol{g}, \tag{12}$$

where $\boldsymbol{J}_{lk}$ is the Jacobian matrix mapped from the CoM velocity of link l to joint k; m_l is the mass of link l, and $\mathcal{D}(k)$ is a set of descendant links of k.

C. Learned Policy

Directly applying the feedforward torque $\boldsymbol{\tau}_j$ may not be enough to accomplish complex loco-manipulation tasks due to the large model mismatch. As a remedy, RAMBO incorporate a learned policy trained in simulated environments using RL to improve the overall robustness of the controller over unconsidered dynamic effects.

An RL problem is formulated as a Markov Decision Process (MDP), represented by a tuple $\langle \mathcal{O}, \mathcal{A}, P, r, \gamma \rangle$, where $\mathcal{O}$ is the observation space; $\mathcal{A}$ is the action space; $P_{O'|O,A}$ is the transition probability; $r : \mathcal{O} \times \mathcal{A} \to \mathbb{R}$ is the reward function; γ is the discounted factor. The policy $\pi_\theta : \mathcal{O} \to \mathcal{A}$, which is parameterized by θ, is trained to maximize the expected sum of discounted reward

$$\mathbb{E}_{o_0\sim p_0(\cdot), o_{t+1}\sim P(\cdot|o_t, \pi_\theta(o_t))} \sum_{t=0}^{T} \gamma^t r(o_t, \pi_\theta(o_t)), \tag{13}$$

over an episode of T, where p_0 is the initial state distribution.

We design the observation space $\mathcal{O}$ to include the proprioceptive information, gait information, kinematic joint position target, and user commands. They are chosen to ensure

reward function can be successfully induced from only the observation. We stack 6-step history of observations as input to the policy [44]. In contrast to previous works [42, 43], the action $a_t \in \mathcal{A}$ is designed to have two separate heads: base acceleration correction $\Delta_{\mathcal{P}}\hat{\boldsymbol{a}}$ and joint position correction $\Delta\hat{\boldsymbol{q}}_j$, providing feedback to both feedforward torque calculation and joint positions. The surrogate targets are

$$\begin{aligned} {}_{\mathcal{P}}\tilde{\boldsymbol{a}} &= {}_{\mathcal{P}}\hat{\boldsymbol{a}} + \Delta_{\mathcal{P}}\hat{\boldsymbol{a}} \\ \tilde{\boldsymbol{q}}_j &= \hat{\boldsymbol{q}}_j + \Delta\hat{\boldsymbol{q}}_j, \end{aligned} \tag{14}$$

where ${}_{\mathcal{P}}\tilde{\boldsymbol{a}}$ is taken as the target for the base acceleration for the reaction force optimization module. The desired joint position $\tilde{\boldsymbol{q}}_j$ is used to calculate the final joint torque command sent to the robot, formulated as

$$\boldsymbol{\tau}_j = \boldsymbol{\tau}_j^{\text{FF}} + \boldsymbol{\tau}_j^{\text{GC}} + k_p(\tilde{\boldsymbol{q}}_j - \boldsymbol{q}_j) + k_d(\dot{\tilde{\boldsymbol{q}}}_j - \dot{\boldsymbol{q}}_j), \tag{15}$$

where k_p, k_d are the proportional and derivative gains for the joint PD controller.

The reward function consists of a combination of task-related rewards and regularizations

$$r = r_{\text{task}} + r_{\text{reg}}, \tag{16}$$

where $r_{\text{task}} = \prod_i r_{\text{task}}^i$ and $r_{\text{reg}} = \prod_j r_{\text{reg}}^j$ are a product of series sub-rewards. Both rewards are designed to ensure the success of tracking user commands and regularized action. For detailed description of the observation space and reward functions, please refer to Table I and Table II.

Term name	Symbol	Dimension
Base height	$\boldsymbol{p}_z$	1
Projected gravity	${}_{\mathcal{B}}\boldsymbol{g}$	3
Base linear velocity	${}_{\mathcal{B}}\boldsymbol{v}$	3
Base angular velocity	${}_{\mathcal{B}}\boldsymbol{\omega}$	3
Joint position	$\boldsymbol{q}_j - \boldsymbol{q}_{j0}$	12
Joint velocity	$\dot{\boldsymbol{q}}_j$	12
Gait phase	$\boldsymbol{\phi}$	4
Gait mode	$\boldsymbol{\psi}$	4
Desired joint position	$\hat{\boldsymbol{q}}_j - \boldsymbol{q}_{j0}$	12
Velocity command	$({}_{\mathcal{P}}\hat{v}_x, {}_{\mathcal{P}}\hat{v}_y, {}_{\mathcal{P}}\hat{\omega}_z)$	3
EE position command	${}_{\mathcal{P}}\hat{\boldsymbol{r}}_i^m$	$3 \cdot \lvert\mathcal{M}\rvert$
EE force command	${}_{\mathcal{P}}\hat{\boldsymbol{u}}_i$	$3 \cdot \lvert\mathcal{M}\rvert$
Last action	a_{t-1}	18

TABLE I: Detailed observation space $\mathcal{O}$. $\boldsymbol{q}_{j0}$ is the default joint position; $\boldsymbol{\phi} \in [-1, 1]$ is the phase for each periodic limb motion; $\boldsymbol{\psi} \in \{-1, 0, 1\}$ is the mode for each limb to distinguish the periodic patterns (*swing*, *gait*, *stance*); $\lvert\cdot\rvert$ is the cardinality of the set.

Reward	Term name	Error $\boldsymbol{\epsilon}$	Sensitivity σ
r_{task}	Base height	$\boldsymbol{p}_z - \hat{\boldsymbol{p}}_z$	0.1 / 0.2
	Base orientation	${}_{\mathcal{B}}\boldsymbol{g}_z - {}_{\mathcal{B}}\hat{\boldsymbol{g}}_z$	0.3 / 0.6
	Linear velocity	${}_{\mathcal{P}}\boldsymbol{v}_{x,y} - {}_{\mathcal{P}}\hat{\boldsymbol{v}}_{x,y}$	0.2 / 0.3
	Angular velocity	${}_{\mathcal{P}}\boldsymbol{\omega}_z - {}_{\mathcal{P}}\hat{\boldsymbol{\omega}}_z$	0.3 / 0.4
	EE position	${}_{\mathcal{P}}\boldsymbol{r}_i^m - {}_{\mathcal{P}}\hat{\boldsymbol{r}}_i^m$	0.1
r_{reg}	Contact mismatch	$\hat{c}_i \neq c_i$	-
	Joint acceleration	$\ddot{\boldsymbol{q}}_j$	700 / 500
	Joint torque	$\boldsymbol{\tau}_j$	100
	Action rate	$a_t - a_{t-1}$	10.0
	Action scale	a_t	8.0

TABLE II: Detailed reward functions. To ensure all reward functions produce values with the range of $[0, 1]$, we map the error term for each reward using $r = \exp(\lVert\boldsymbol{\epsilon}\rVert^2 / \sigma^2)$. The contact mismatch reward is mapped using $r = 0.5^{\lvert\hat{c}_i \neq c_i\rvert}$. $(\cdot)/(\cdot)$ indicates different values for quadruped or biped tasks. We set desired base height $\hat{\boldsymbol{p}}_z$ to 0.3 and 0.45, desired gravity vector ${}_{\mathcal{B}}\hat{\boldsymbol{g}}_z$ to $[0, 0, -1]$ and $[-1, 0, 0]$ for quadruped and biped tasks.

V. Results

RAMBO offers a general framework for whole-body loco-manipulation on legged systems. We demonstrate its effectiveness on the Unitree Go2 [25], a small-scale quadruped robot, across a variety of tasks involving both quadrupedal and bipedal locomotion.

We implemented two scenarios targeting the quadrupedal and bipedal tasks, respectively. In the quadruped tasks, the robot walks using three legs while lifting the front-left leg to perform manipulation. For the bipedal tasks, it walks solely on its hind legs while using both front legs for manipulation. We design the base orientation to keep flat to the ground for quadruped tasks, while demonstrating more challenging loco-manipulation tasks by making the robot to perform bipedal walking with upright pose. These bipedal demonstrations highlight the potential of RAMBO for applications on humanoids.

We leverage Isaac Lab [45], a massive parallel training framework on GPU, to efficiently train the policy with Proximal Policy Optimization [46]. The detailed training hyperparameters can be found in Table III. During training, we leverage qpth [47], a fast batch QP solver implemented in PyTorch, to solve parallel QPs to generate feedforward torques. Despite the effectiveness of qpth in training, we employ OSQP [48] as a faster QP solver for a single problem to ensure the whole control pipeline runs at 100 Hz in the real-world experiments.

To facilitate the training with force command at end-effectors, we apply virtual external forces acted at the same end-effector in the opposite direction, similarly to the training technique proposed by Portela et al. [36]. We employ various Domain Randomization [33] to ensure successful sim-to-real transfer. Detailed command sampling and domain randomization can be found in Table IV.

Term	Value	Term	Value
# environments	4096	# steps per iter	24
Episode length	10 (s)	γ	0.99
Learning rate	$1e^{-3}$	λ	0.95
Desired KL	0.02	Clip ratio	0.2
Value loss coeff	1.0	Policy network	MLP
Entropy coeff	$1e^{-3}$	Policy hidden	[512, 256, 128]
Action head scale (base acceleration)	5.0	Policy activation	ELU
		Value network	MLP
Action head scale (joint position)	0.15	Value hidden	[512, 256, 128]
		Value activation	ELU
Joint P gain k_p	40	Joint D gain k_d	1.0

TABLE III: Detailed training hyperparameters.

A. Quantitative Evaluation

To evaluate the performance of RAMBO, we compare RAMBO with the following baselines in simulated environments in terms of tracking the desired base velocity, desired EE positions and forces:

- **Vanilla**: Policies trained to track the commands in an end-to-end fashion [36]. We use the same contact information from the gait pattern to facilitate the policy to generate proper gait patterns;

Term	Value
Forward velocity ${}_{\mathcal{P}}\hat{v}_x$	$[-0.5, 0.5]$ (m/s)
Side velocity ${}_{\mathcal{P}}\hat{v}_y$	$[-0.5, 0.5]$ / $[-0.0, 0.0]$ (m/s)
Turning velocity ${}_{\mathcal{P}}\hat{\omega}_z$	$[-0.5, 0.5]$ (rad/s)
EE position x $({}_{\mathcal{P}}\hat{\boldsymbol{p}}_i^m)_x$	$[0.1934, 0.5]$ / $[0.15, 0.30]$ (m)
EE position y $({}_{\mathcal{P}}\hat{\boldsymbol{p}}^m)_y$	$[0, 0.2]$ (m)
EE position z $({}_{\mathcal{P}}\hat{\boldsymbol{p}}^m)_z$	$[0.0, 0.4]$ / $[0.3, 0.9]$ (m)
EE force in x, y, z ${}_{\mathcal{P}}\hat{\boldsymbol{u}}$	$[-30, 30]$ / $[-20, 20]$ (N)
Friction coefficient	$[0.5, 1.5]$
Add mass for base	$[-2.0, 2.0]$ (kg)
CoM offset for base	$[-0.05, 0.05]$ (m)
Actuator random delay	$[0, 20]$ (ms)
Base position noise $\boldsymbol{p}_x, \boldsymbol{p}_y$	$[-0.05, 0.05]$ (m)
Base position noise $\boldsymbol{p}_z$	$[-0.02, 0.02]$ (m)
Base orientation noise $\boldsymbol{R}$	$[-0.05, 0.05]$
Base linear velocity noise $\boldsymbol{v}$	$[-0.1, 0.1]$ (m/s)
Base angular velocity noise $\boldsymbol{\omega}$	$[-0.15, 0.15]$ (m/s)
Joint position noise $\boldsymbol{q}_j$	$[-0.01, 0.01]$ (rad)
Joint velocity noise $\dot{\boldsymbol{q}}_j$	$[-1.5, 1.5]$ (rad/s)

TABLE IV: Detailed command sampling and domain randomization. $(\cdot)/(\cdot)$ represents different values used for quadruped and biped tasks respectively. The orientation noise is estimated from the noise sampled on unit quaternion representations. Besides the terms listed above, we also add random push on the robot base within an episode. Note that the EE position and forces are listed only for the front-left limb of the robot. For biped tasks, we sample the quantities in symmetry for the front-right limb.

- **Imitation**: Policies trained to track the target joint angles from kinematic reference [49] in addition to the vanilla policies;
- **Residual**: Policies trained to produce joint position residuals in addition to the kinematic reference, without WBC to generate feedforward torques;
- **RAMBO-base**: WBC with a feedback policy outputting acceleration correction $\Delta_{\mathcal{P}}\hat{\boldsymbol{a}}$ only;
- **RAMBO-joint**: WBC and a feedback policy outputting joint correction $\Delta\hat{\boldsymbol{q}}_j$ only;
- **RAMBO-ff**: WBC only (no training needed).

Note that the action space of the vanilla and imitation policies is an offset of joint positions relative to the fixed default positions. We set the action scale to 0.25 to facilitate exploration, a common choice in the previous works [32]. In comparison, RAMBO and residual policies have joint actions relative to the kinematic reference with a scale of 0.15.

During evaluation, we randomly sample user commands in base velocity, EE positions and forces. As shown in Table V, RAMBO achieves comparable or superior performance to all baselines across both quadrupedal and bipedal tasks. Notably, our method exhibits a clear advantage in tracking target end-effector positions, significantly reducing tracking errors. These results highlight RAMBO 's precision and effectiveness in whole-body loco-manipulation for both locomotion modes. Note that since we apply virtual external forces at the end-effectors, the baselines' lower performance in tracking end-effector positions indicates that they fail to generate the appropriate desired forces while following user-commanded end-effector positions.

We trained vanilla and imitation policies using the similar reward structure to those of RAMBO. While we performed reward shaping as best as we could, we found it difficult to balance the various objectives. For those achieving good tracking behavior in velocity and EE position and forces, they often produced much less regularized actions for deployment. We also emphasize the critical role of corrective feedback through the learned policy in RAMBO. As shown in Table V, incorporating residual feedback, particularly at the joint position level, leads to a substantial reduction in EE tracking error in comparison with RAMBO-ff. While the residual policies were able to track certain commands in quadruped tasks without WBC, they failed to generate sufficient contact forces for bipedal walking, further highlighting the importance of feedforward torques.

B. Real-world Experiments

By changing user inputs during runtime, we demonstrate the success execution of diverse loco-manipulation skills such as bipedal cart pushing and dice holding with the same policy trained for bipedal tasks. Similarly with the policy trained for quadrupedal tasks, RAMBO achieves stable object holding and plate balancing while walking with other three legs, as shown in the snapshots of these experiments are included in Fig. 1. In more detail, we overlay the multiple images of shopping cart pushing and sponge holding tasks in Fig. 5 to demonstrate that RAMBO enables the quadruped to apply the desired force stably while walking dynamically.

In addition, we demonstrate the robustness of RAMBO by commanding the robot on uneven terrains and exerting external pushes in both bipedal and quadrupedal tasks, shown in Fig. 6. We refer the interested readers to the hardware demonstrations in the supplementary video.

C. Further Comparison with Vanilla Policy

We further compare the performance of RAMBO against a vanilla policy which is regularized and deployable on hardware in terms of tracking the desired EE position. While both policies achieve smooth end-effector position control during static standing, RAMBO exhibits more accurate EE tracking while walking with the remaining three legs, as shown in Fig. 7. In contrast, the vanilla policy tends to sacrifice EE tracking precision in favor of maintaining overall stability during dynamic locomotion.

Fig. 5: Snapshots of two whole-body loco-manipulation tasks, where the desired EE force are overlaid as pink arrows. *Upper*: pushing a shopping cart while walking in bipedal mode; *bottom*: holding a sponge while walking in quadrupedal mode.

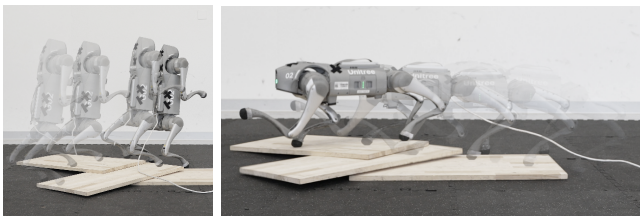

Fig. 6: Snapshots of experiments demonstrating RAMBO's robustness on uneven terrains in bipedal mode (*left*) and quadrupedal mode (*right*).

	Error in tracking from quadruped task			Error in tracking from biped task		
	lin vel (m/s) ↓	ang vel (rad/s) ↓	EE pos (m) ↓	lin vel (m/s) ↓	ang vel (rad/s) ↓	EE pos (m) ↓
Vanilla	0.387 ± 0.022	0.257 ± 0.026	0.254 ± 0.009	0.313 ± 0.026	0.353 ± 0.066	0.431 ± 0.024
Imitation	0.383 ± 0.022	0.253 ± 0.025	0.257 ± 0.010	0.310 ± 0.027	0.334 ± 0.060	0.448 ± 0.018
Residual	0.153 ± 0.027	0.153 ± 0.067	0.077 ± 0.013	0.383 ± 0.054	0.508 ± 0.271	0.347 ± 0.030
RAMBO-base	0.321 ± 0.041	0.293 ± 0.126	0.168 ± 0.036	0.305 ± 0.060	0.389 ± 0.171	0.453 ± 0.025
RAMBO-joint	0.108 ± 0.014	0.101 ± 0.032	0.046 ± 0.007	0.306 ± 0.046	0.383 ± 0.109	0.134 ± 0.044
RAMBO-ff	0.374 ± 0.036	0.404 ± 0.154	0.286 ± 0.048	0.320 ± 0.061	0.389 ± 0.187	0.447 ± 0.026
RAMBO	$\mathbf{0.087 \pm 0.009}$	$\mathbf{0.085 \pm 0.022}$	$\mathbf{0.039 \pm 0.003}$	0.286 ± 0.039	0.352 ± 0.075	$\mathbf{0.036 \pm 0.002}$

TABLE V: Quantitative evaluation of RAMBO compared with baselines. The mean and variance are calculated across 3 different seeds with 1000 episode for each seed. For biped tasks, the end-effector tracking error is calculated as the mean of tracking FL and FR end-effectors.

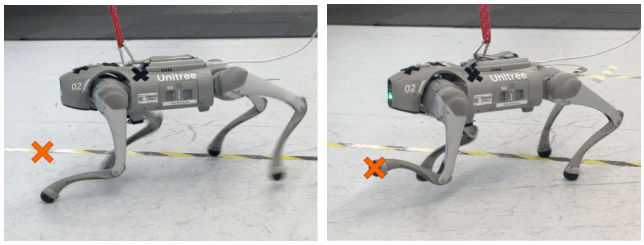

Fig. 7: Comparison between vanilla policy (*left*) and RAMBO (*right*).

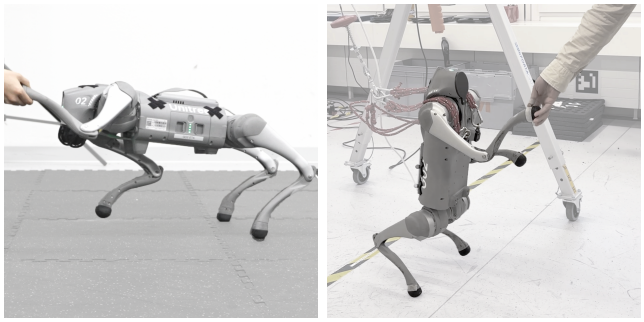

Fig. 8: Compliance enabled by RAMBO in quadruped mode (*left*) and bipedal mode (*right*). The robot is commanded to maintain its end-effector position while allowing external forces to displace it compliantly.

We attribute this discrepancy to the relatively large exploration space in the vanilla training setup, which makes it difficult for the policy to consistently prioritize accurate EE tracking while generating appropriate contact forces through its end-effectors. Instead, RAMBO separates feedforward torque from WBC and joint feedback from policy, which allows the learned policy to provide corrective offsets relative to reference positions with a much smaller exploration space, improving robustness without significantly altering the desired motion.

D. Compliance

Finally, we showcase one of the features enabled by RAMBO, compliance, by lowering the PD gains at the joints associated with the manipulation end-effectors. Thanks to the WBC, the robot is able to maintain a stable end-effector position while walking and being compliant against external pushes. As illustrated in Fig. 8, this compliance is demonstrated through an interactive handshake, where users are able to physically engage with the robot safely. By decoupling the feedforward torque and PD feedback, RAMBO enables a flexible trade-off between compliance and accuracy in end-effector tracking, an essential property for ensuring safe and adaptive interactions.

VI. CONCLUSION

We present RAMBO, a hybrid control framework that combines a model-based whole-body controller with a learned policy to enable robust and precise whole-body loco-manipulation on legged robots. By leveraging a computationally efficient QP based on the SRB model, RAMBO optimizes feedforward torque commands while maintaining robustness through learning-based feedback. Our results in both simulation and on hardware demonstrate RAMBO's advantage in tracking user commands across a range of quadrupedal and bipedal loco-manipulation tasks. Additionally, the framework allows for a flexible trade-off between tracking accuracy and compliance, which is crucial for safe and adaptive interaction with environment.

RAMBO is not without limitations. Currently, it relies solely on proprioceptive information, which negatively affects performance due to drift in state estimation. As a next step, we aim to incorporate additional sensing modalities to enhance robustness and accuracy. Nevertheless, we see strong potential for RAMBO in future loco-manipulation research, including the integration of full-order WBC dynamics models and its application to humanoids. Other promising directions include extending the framework with online model adaptation to further improve generalization and precision.

REFERENCES

[1] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[2] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[3] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *Science Robotics*, vol. 9, no. 88, p. eadi7566, 2024.

[4] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Real-world humanoid locomotion with reinforcement learning," *Science Robotics*, vol. 9, no. 89, p. eadi9579, 2024.

[5] Z. Gu, J. Li, W. Shen, W. Yu, Z. Xie, S. McCrory, X. Cheng, A. Shamsah, R. Griffin, C. K. Liu *et al.*, "Humanoid locomotion and manipulation: Current progress and challenges in control, planning, and learning," *arXiv preprint arXiv:2501.02116*, 2025.

[6] Z. Fu, X. Cheng, and D. Pathak, "Deep whole-body control: learning a unified policy for manipulation and locomotion," in *Conference on Robot Learning*. PMLR, 2023, pp. 138–149.

[7] Y. Ji, G. B. Margolis, and P. Agrawal, "Dribblebot: Dynamic legged manipulation in the wild," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5155–5162.

[8] P. Arm, M. Mittal, H. Kolvenbach, and M. Hutter, "Pedipulate: Enabling manipulation skills using a quadruped robot's leg," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 5717–5723.

[9] S. Lyu, X. Lang, H. Zhao, H. Zhang, P. Ding, and D. Wang, "Rl2ac: Reinforcement learning-based rapid online adaptive control for legged robot robust locomotion," in *Proceedings of the Robotics: Science and Systems*, 2024.

[10] S. Ha, J. Lee, M. van de Panne, Z. Xie, W. Yu, and M. Khadiv, "Learning-based legged locomotion: State of the art and future perspectives," *The International Journal of Robotics Research*, p. 02783649241312698, 2024.

[11] X. Cheng, A. Kumar, and D. Pathak, "Legs as manipulator: Pushing quadrupedal agility beyond locomotion," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5106–5112.

[12] C. Zhang, W. Xiao, T. He, and G. Shi, "Wococo: Learning whole-body humanoid control with sequential contacts," in *8th Annual Conference on Robot Learning*, 2024.

[13] S. Jeon, M. Jung, S. Choi, B. Kim, and J. Hwangbo, "Learning whole-body manipulation for quadrupedal robot," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 699–706, 2023.

[14] C. Schwarke, V. Klemm, M. Van der Boon, M. Bjelonic, and M. Hutter, "Curiosity-driven learning of joint locomotion and manipulation tasks," in *Proceedings of the 7th Conference on Robot Learning*, vol. 229. PMLR, 2023, pp. 2594–2610.

[15] H. Ha, Y. Gao, Z. Fu, J. Tan, and S. Song, "Umi on legs: Making manipulation policies mobile with manipulation-centric whole-body controllers," in *8th Annual Conference on Robot Learning*, 2024.

[16] C. D. Bellicoso, K. Krämer, M. Stäuble, D. Sako, F. Jenelten, M. Bjelonic, and M. Hutter, "Alma-articulated locomotion and manipulation for a torque-controllable robot," in *2019 International conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 8477–8483.

[17] 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.

[18] 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.

[19] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, 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)*. IEEE, 2018, pp. 1–9.

[20] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous robots*, vol. 40, pp. 429–455, 2016.

[21] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 295–302.

[22] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-based control for dynamic legged robots," *IEEE Transactions on Robotics*, vol. 40, pp. 43–63, 2023.

[23] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, planning, and control for quadruped locomotion over challenging terrain," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 236–258, 2011.

[24] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback mpc for torque-controlled legged robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4730–4737.

[25] Unitree Robotics, "Unitree Go2," https://www.unitree.com/go2, accessed: 2025-03-29.

[26] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[27] A. Rigo, Y. Chen, S. K. Gupta, and Q. Nguyen, "Contact optimization for non-prehensile loco-manipulation via hierarchical model predictive control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9945–9951.

[28] 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*, 2024.

[29] 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.

[30] F. Jenelten, R. Grandia, F. Farshidian, and M. Hutter, "Tamols: Terrain-aware motion optimization for legged systems," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3395–3413, 2022.

[31] Y. Ding, A. Pandala, and H.-W. Park, "Real-time model predictive control for versatile dynamic motions in quadrupedal robots," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8484–8490.

[32] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 91–100.

[33] 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)*. IEEE, 2017, pp. 23–30.

[34] J. Dao, K. Green, H. Duan, A. Fern, and J. Hurst, "Sim-to-real learning for bipedal locomotion under unsensed dynamic loads," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 10 449–10 455.

[35] Y. Yuan, J. Cheng, N. A. Urpí, and S. Coros, "Caiman: Causal action influence detection for sample efficient loco-manipulation," *arXiv preprint arXiv:2502.00835*, 2025.

[36] 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)*. IEEE, 2024, pp. 15 366–15 372.

[37] S. Chen, B. Zhang, M. W. Mueller, A. Rai, and K. Sreenath, "Learning torque control for quadrupedal locomotion," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*. IEEE, 2023, pp. 1–8.

[38] P. Li, H. Li, G. Sun, J. Cheng, X. Yang, G. Bellegarda, M. Shafiee, Y. Cao, A. Ijspeert, and G. Sartoretti, "Sata: Safe and adaptive torque-based locomotion policies inspired by animal learning," *arXiv preprint arXiv:2502.12674*, 2025.

[39] J.-P. Sleiman, M. Mittal, and M. Hutter, "Guided reinforcement learning for robust multi-contact loco-manipulation," in *8th Annual Conference on Robot Learning*, 2024.

[40] D. Kang, J. Cheng, M. Zamora, F. Zargarbashi, and S. Coros, "Rl+ 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.

[41] F. Jenelten, J. He, F. Farshidian, and M. Hutter, "Dtc: Deep tracking control," *Science Robotics*, vol. 9, no. 86, p. eadh5401, 2024.

[42] Z. Xie, X. Da, B. Babich, A. Garg, and M. v. de Panne, "Glide: Generalizable quadrupedal locomotion in diverse environments with a centroidal model," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2022, pp. 523–539.

[43] Y. Yang, G. Shi, X. Meng, W. Yu, T. Zhang, J. Tan, and B. Boots, "Cajun: Continuous adaptive jumping using a learned centroidal controller," in *Conference on Robot Learning*. PMLR, 2023, pp. 2791–2806.

[44] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *Conference on Robot Learning*. PMLR, 2023, pp. 22–31.

[45] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar *et al.*, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.

[46] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[47] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International conference on machine learning*. PMLR, 2017, pp. 136–145.

[48] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[49] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions On Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.