RAMBO: RL-augmented Model-based Whole-body Control for Loco-manipulation

Author Names Omitted for Anonymous Review.

***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].

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.

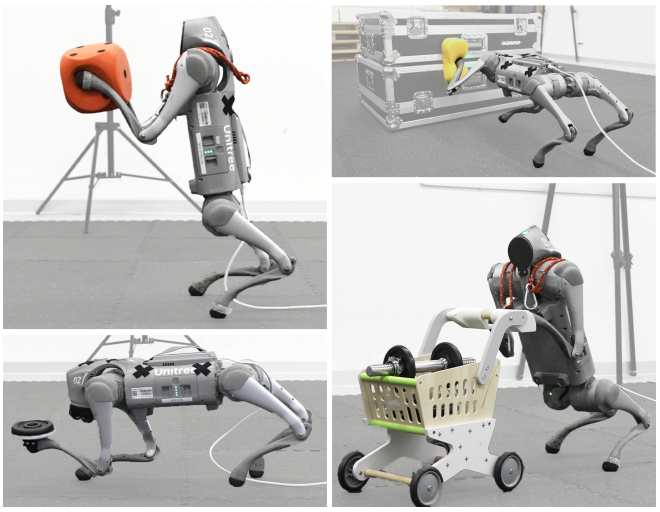

Fig. 1: Various whole-body loco-manipulation tasks enabled by RAMBO on Unitree Go2 [6] in both quadrupedal and bipedal modes.

II. METHOD

As shown in Fig. 2, 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.

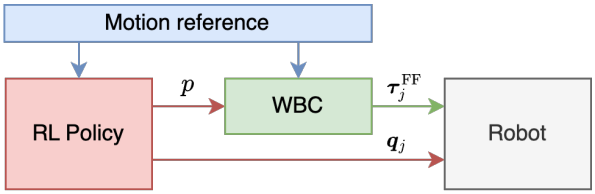

Fig. 2: Overview of the RAMBO architecture. It consists of three core components: (1) a motion reference generator, (2) a WBC module that computes feedforward joint torques $\boldsymbol{\tau}_j^{\text{FF}}$, and (3) an RL policy that generates feedback corrections to both WBC input parameters and motion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

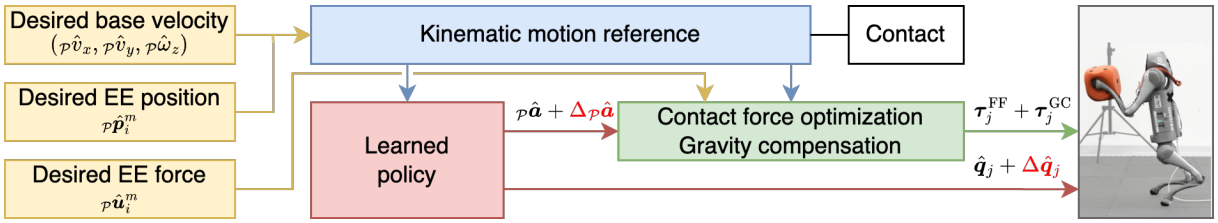

Fig. 3: Detailed architecture of the RAMBO control framework. The desired base velocity and EE positions are used to generate a kinematic motion reference, which is sent to the policy and whole-body control module. The whole-body control module also takes the desired EE force to compute the feedforward 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.

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 $(_{\mathcal{P}}\boldsymbol{p}_{\text{lift}}, _{\mathcal{P}}\boldsymbol{p}_{\text{mid}}, _{\mathcal{P}}\boldsymbol{p}_{\text{land}})_i$. The reference for each joint $\hat{\boldsymbol{q}}_j$ is calculated using inverse kinematics (IK)

$$\hat{\boldsymbol{q}}_j = \text{IK}(_{\mathcal{P}}\boldsymbol{p}, _{\mathcal{P}}\boldsymbol{R}, _{\mathcal{P}}\hat{\boldsymbol{p}}_1, ..., _{\mathcal{P}}\hat{\boldsymbol{p}}_N). \tag{1}$$

For accounting the force command and the dynamics effect from manipulation tasks, the contact state for manipulation $\hat{\boldsymbol{c}}^m = \{\hat{c}_i | i \in \mathcal{M}\}$ 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

$$\begin{aligned} &_{\mathcal{P}}\hat{\boldsymbol{p}} = _{\mathcal{P}}\boldsymbol{p}, _{\mathcal{P}}\hat{\boldsymbol{R}} = _{\mathcal{P}}\boldsymbol{R} \\ &_{\mathcal{P}}\hat{\boldsymbol{v}} = [_{\mathcal{P}}\hat{v}_x, _{\mathcal{P}}\hat{v}_y, 0], _{\mathcal{P}}\hat{\boldsymbol{\omega}} = [0, 0, _{\mathcal{P}}\hat{\omega}_z], \hat{\dot{\boldsymbol{q}}}_j = \boldsymbol{0}. \end{aligned} \tag{2}$$

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

$$_{\mathcal{P}}\hat{\boldsymbol{a}}_l = \kappa_{lp}(_{\mathcal{P}}\hat{\boldsymbol{p}} - _{\mathcal{P}}\boldsymbol{p}) + \kappa_{ld}(_{\mathcal{P}}\hat{\boldsymbol{v}} - _{\mathcal{P}}\boldsymbol{v}) \tag{3}$$

$$_{\mathcal{P}}\hat{\boldsymbol{a}}_a = \kappa_{ap}\log(_{\mathcal{P}}\hat{\boldsymbol{R}}^\top \cdot _{\mathcal{P}}\boldsymbol{R})^\vee + \kappa_{ad}(_{\mathcal{P}}\hat{\boldsymbol{\omega}} - _{\mathcal{P}}\boldsymbol{\omega}), \tag{4}$$

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. 3. 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

$$\min_{_{\mathcal{B}}\boldsymbol{u}_i, i\in\mathcal{N}} \|\Delta_{\mathcal{B}}\boldsymbol{a}\|_{\boldsymbol{U}}^2 + \sum_{i\in\mathcal{M}} \|\Delta_{\mathcal{B}}\boldsymbol{u}_i\|_{\boldsymbol{V}}^2 + \sum_{i\in\mathcal{L}} \|_{\mathcal{B}}\boldsymbol{u}_i\|_{\boldsymbol{W}}^2 \tag{5a}$$

$$\text{subject to : } _{\mathcal{B}}\boldsymbol{a} = \sum_{i=1}^{N} \boldsymbol{A}_i {}_{\mathcal{B}}\boldsymbol{u}_i + \hat{\boldsymbol{g}} \tag{5b}$$

$$(_{\mathcal{B}}\boldsymbol{u}_i)_z = 0, \qquad i \in \mathcal{L}_{\text{swing}} \tag{5c}$$

$$\|(_{\mathcal{B}}\boldsymbol{u}_i)_x\| \le \mu(_{\mathcal{B}}\boldsymbol{u}_i)_z, \qquad i \in \mathcal{L}_{\text{stance}} \tag{5d}$$

$$\|(_{\mathcal{B}}\boldsymbol{u}_i)_y\| \le \mu(_{\mathcal{B}}\boldsymbol{u}_i)_z, \qquad i \in \mathcal{L}_{\text{stance}} \tag{5e}$$

$$u_{z,\min} \le (_{\mathcal{B}}\boldsymbol{u}_i)_z \le u_{z,\max}, \qquad i \in \mathcal{L}_{\text{stance}} \tag{5f}$$

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 **??**; 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

$$\boldsymbol{\tau}_j^{\text{FF}} = \sum_{i=1}^{N} \boldsymbol{J}_i^\top \cdot _{\mathcal{B}}\boldsymbol{u}_i, \tag{6}$$

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

$$(\boldsymbol{\tau}_j^{\text{GC}})_k = -\sum_{l\in\mathcal{D}(k)} \boldsymbol{J}_{lk}^\top \cdot m_l \boldsymbol{g}, \tag{7}$$

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.

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 [7]. In contrast to previous works [8, 9], 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

$$\begin{aligned} {}_{\mathcal{P}}\tilde{\boldsymbol{a}} &= {}_{\mathcal{P}}\hat{\boldsymbol{a}} + \Delta_{\mathcal{P}}\hat{\boldsymbol{a}} \\ \tilde{\boldsymbol{q}}_j &= \hat{\boldsymbol{q}}_j + \Delta\hat{\boldsymbol{q}}_j, \end{aligned} \tag{8}$$

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

$$\boldsymbol{\tau}_j = \boldsymbol{\tau}_j^{\text{FF}} + \boldsymbol{\tau}_j^{\text{GC}} + k_p(\tilde{\boldsymbol{q}}_j - \boldsymbol{q}_j) + k_d(\dot{\tilde{\boldsymbol{q}}}_j - \dot{\boldsymbol{q}}_j), \tag{9}$$

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

$$r = r_{\text{task}} + r_{\text{reg}}, \tag{10}$$

where $r_{\text{task}} = \prod_i r^i_{\text{task}}$ and $r_{\text{reg}} = \prod_j r^j_{\text{reg}}$ are a product of series sub-rewards. Both rewards are designed to ensure the success of tracking user commands and regularized action.

III. Results

RAMBO offers a general framework for whole-body loco-manipulation on legged systems. We demonstrate its effectiveness on the Unitree Go2 [6], 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 [10], a massive parallel training framework on GPU, to efficiently train the policy with Proximal Policy Optimization [11]. The detailed training hyperparameters can be found in **??**. During training, we leverage qpth [12], 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 [13] 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. [14]. We employ various Domain Randomization [15] to ensure successful sim-to-real transfer.

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 [14]. We use the same contact information from the gait pattern to facilitate the policy to generate proper gait patterns;
- **Imitation**: Policies trained to track the target joint angles from kinematic reference [16] 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 [17]. 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 I, 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 I, 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

	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 I: 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.

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. 4 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. 5. We refer the interested readers to the hardware demonstrations in the supplementary video.

C. 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. 6, 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.

Fig. 4: 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.

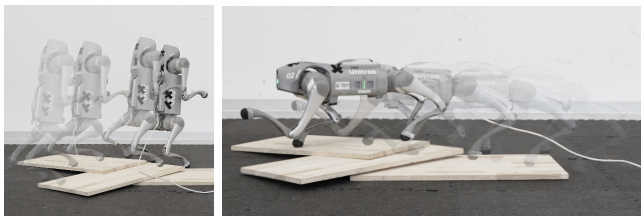

Fig. 5: Snapshots of experiments demonstrating RAMBO's robustness on uneven terrains in bipedal mode (*left*) and quadrupedal mode (*right*).

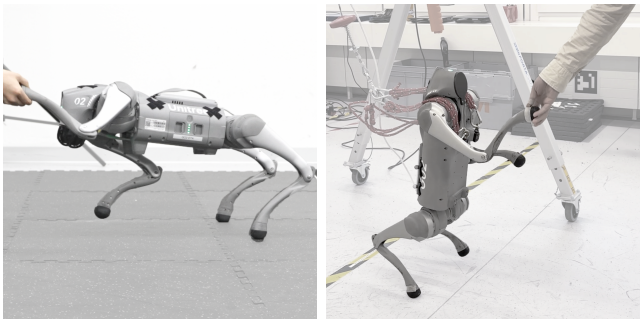

Fig. 6: Compliance enabled by RAMBO in quadruped mode (*left*) and bipedal mode (*right*). The robot is commanded to maintain its end-effector position while allowing external forces to displace it compliantly.

IV. 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.

Currently, RAMBO 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 higher degree-of-freedom 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] Unitree Robotics, "Unitree Go2," https://www.unitree.com/go2, accessed: 2025-03-29.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.