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

A Modular Residual Learning Framework to Enhance Model-Based Approach for Robust Locomotion

Min-Gyu Kim, *Graduate Student Member, IEEE*, Dongyun Kang, *Graduate Student Member, IEEE*, Hajun Kim, *Graduate Student Member, IEEE*, and Hae-Won Park, *Member, IEEE*

***Abstract*—This letter presents a novel approach that combines the advantages of both model-based and learning-based frameworks to achieve robust locomotion. The residual modules are integrated with each corresponding part of the model-based framework, a footstep planner and dynamic model designed using heuristics, to complement performance degradation caused by a model mismatch. By utilizing a modular structure and selecting the appropriate learning-based method for each residual module, our framework demonstrates improved control performance in environments with high uncertainty, while also achieving higher learning efficiency compared to baseline methods. Moreover, we observed that our proposed methodology not only enhances control performance but also provides additional benefits, such as making nominal controllers more robust to parameter tuning. To investigate the feasibility of our framework, we demonstrated residual modules combined with model predictive control in a real quadrupedal robot. Despite uncertainties beyond the simulation, the robot successfully maintains balance and tracks the commanded velocity.**

***Index Terms*—Legged robots, machine learning for robot control, optimization and optimal control.**

I. Introduction

LEGGED systems have fascinated researchers due to their potential to navigate urban and harsh environments while effectively performing assigned tasks. Model-based approaches (MBA), particularly model predictive control (MPC), have been widely studied for their capacity to handle system dynamics and constraints. Despite the rise of learning-based approaches (LBA), MBA remains central to generate safe and consistent motion.

However, MBA faces challenges, most notably model mismatches resulting from necessary simplifications for real-time feasibility. Since legged systems exhibit complex hybrid dynamics due to contact, the models used for control design are typically simplified, such as the single rigid body model or the linear inverted pendulum model. While this reduces computational cost, it can lead to the loss of critical information such as contact dynamics, degrading performance in unmodeled scenarios.

To address these limitations, LBA can offer a promising complement, excelling in modeling complex behaviors through data. A synergistic integration of MBA and LBA can combine the reliability of MBA with the adaptability of LBA, yielding robust control even under significant uncertainties.

This letter presents a hybrid scheme that leverages LBA to compensate for limitations in existing MBA. While MBA offers refined and reliable motion, they are prone to degraded performance due to minor model inaccuracies or suboptimal heuristics, as well as significant uncertainties such as heavy payload or external disturbances.

To tackle this issue, we introduce *residual modules* into two key components, the footstep planner and the dynamics model. Each residual module is designed using machine learning techniques to provide auxiliary actions. As a result, the hybrid controller offers improved robustness to uncertainty than a conventional MBA, while also preserving consistency outside the training domain compared to the end-to-end LBA. Fig. 1 shows an example of robustness, where a robot handles a payload and disturbance that the nominal MBA alone cannot cope with.

A key advantage of our method is its modular and simplified module design using reinforcement learning (RL) and supervised learning (SL) rather than fully relying on RL, which enables efficient learning while maintaining the nominal

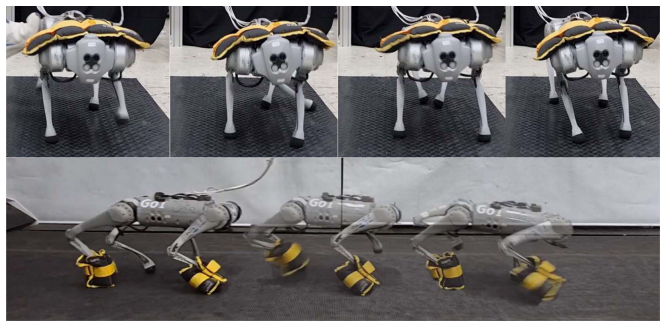

Fig. 1. Snapshots of experimental results. The quadrupedal robot with conventional model-based controller adjusted by residual modules can overcome uncertainties such as unknown payload and disturbances.

Received 31 March 2025; accepted 13 July 2025. Date of publication 23 July 2025; date of current version 31 July 2025. This article was recommended for publication by Associate Editor M. Khadiv and Editor A. Kheddar upon evaluation of the reviewers' comments. This work was supported in part by the Technology Innovation Program(or Industrial Strategic Technology Development Program-Robot Industry Technology Development) (RS-2024-00427719, Dexterous and Agile Humanoid Robots for Industrial Applications) funded by the Ministry of Trade Industry & Energy(MOTIE, Korea), in part by the Korea Evaluation Institute of Industrial Technology (KEIT) funded by the Korea Government (MOTIE) under Grant 20018216, and in part by the Development of mobile intelligence SW for autonomous navigation of legged robots in dynamic and atypical environments for real application. *(Corresponding author: Hae-Won Park.)*

The authors are with the Humanoid Robot Research Center, Department of Mechanical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, South Korea (e-mail: haewonpark@kaist.ac.kr).

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3592067, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3592067

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

controller in the training loop. Specifically, we propose two residual modules: (i) a RL-based residual footstep module to adaptively correct foothold patterns in response to disturbances, and (ii) a SL-based residual dynamics (RD) module to account for discrepancies in the system model.

We address challenges of foothold selection from complex contact dynamics via RL, while compensating continuous-domain dynamics discrepancies through SL, separating them from RL training. This approach reduces the learning search space and simplifies the RL process. To further reduce computational overhead, we employ convex MPC with a simplified dynamics model for 3D motion. Additionally, we apply low-pass filtering to the RD to isolate slowly varying uncertainties, thereby enhancing robustness to noise. By focusing on these slow variations, we assume the residual terms remain constant over the MPC prediction horizon, which further simplifies the optimization problem. Compared to baselines, our method stands out by its architectural simplicity, improved training efficiency, and consistent performance across a wide range of out-of-distribution (OOD) scenarios. We highlight our contributions as follows.

- We propose a hybrid scheme that leverages learning-based residual modules to compensate for performance degradation caused by model inaccuracies and suboptimal heuristics in conventional MBA.
- The framework features a lightweight modular architecture using RL for footstep adaptation and SL for continuous-domain dynamics correction, combined with simplified nominal dynamics and filtering–based RD decoupling to enable efficient learning.
- Extensive experiments show that our method achieves robust and reliable task execution under heavy disturbances and OOD conditions, reduced parameter tuning sensitivity, and improved learning efficiency without compromising performance compared to baselines.

II. Related Work

A. Model- and Learning-Based Approaches

Conventionally, heuristic-based schemes have been introduced for controlling legged robots, wherein the original system is represented by simplified models, such as a linear inverted pendulum model (LIPM) or single rigid body model (SRBM) [1], [2], [3]. This scheme has been verified to be simple but practical through a range of demonstrations.

Among these, MPC has gained attention for its ability to handle dynamic tasks and system constraints [3], [4], [5], [6]. However, its computational cost often necessitates simplifications, such as predefined gait sequences or simplified dynamics such as SRBM, leading to performance degradation.

In contrast, data-driven approaches have demonstrated impressive results in challenging tasks [7], [8], [9], [10], but also face issues such as suboptimal convergence. While techniques like system identification [11], [12] and online adaptation [13], [14] improve flexibility, end-to-end RL methods still struggle with issues such as sample inefficiency and a cumbersome reward engineering process.

B. Residual Estimation

To address the performance degradation of MBA caused by model discrepancy, numerous residual estimation approaches have been proposed across various domains such as legged locomotion, drones, and vehicles. These methods vary based on how accurately the residuals represent the actual discrepancies and whether the residual is updated online or designed offline for use in a hierarchical manner.

Specifically, residual estimation techniques range from adaptive control [15] to data-based approaches, including online regression using linear model [16], probabilistic regression based on basis function learning [17] or Gaussian processes [18], and offline-trained neural network models [19], [20], [21]. Each method exhibits a trade-off between accuracy, real-time capability, and ease of integration into optimization-based controllers, depending on its complexity.

While compensating for dynamics gaps via residual estimation is effective in handling various disturbances and external loads, it may be insufficient on its own to achieve robust control in unstructured environments, particularly for unstable and highly dynamic systems such as legged robots.

C. Hybrid Method

To overcome the limitations of MBA and LBA, recent studies explore hybrid strategies to merge their strengths. Some works use LBA to replace heuristically defined high-level references (e.g. commands or gait sequences) for MBA for better adaptability in uncertain environments [22], [23], [24], [25], [26].

Additional approaches apply residual action, such as torque or joint reference, to directly fine-tune the nominal output [26], [27], [28], [29]. While this can improve reactivity by enabling residuals to override nominal degradation, it may pose instability with uncertainties due to the absence of constrained optimization-based feasibility checks in MBA.

Finally, several studies employ MBA to accelerate the initial learning phase of RL, leveraging their ability to generate refined motions [30], [31]. Similarly, MBA capable of long-term planning, such as trajectory optimization, has been used to address challenges of RL in sparse reward learning [32], [33]. While these approaches improve learning efficiency and convergence, they also introduce computational overhead due to repeated MBA computation during training.

III. Residual Modules With Nominal Controller

A. System Overview

Our approach integrates learning-based residual modules into a MBA, as depicted in Fig 2. Each module is designed to modulate the output of its corresponding nominal module, for improving task-specific performance.

A key aspect of the proposed method is the sequential design of residual modules utilizing both RL and SL, depending on the characteristics of each component. In related work on drones, SL has been used to model aerodynamic or actuator dynamics from real-world data [19], [20], [21]. However, legged robots present additional challenges due to the need for choosing footholds.

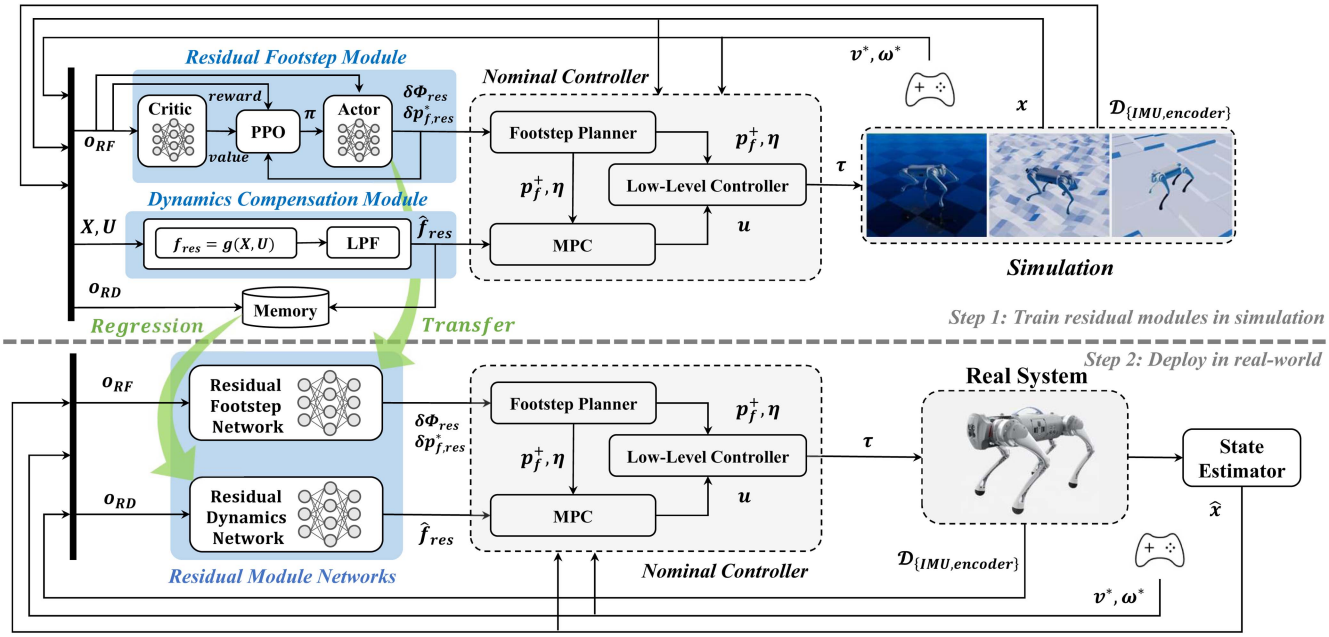

Fig. 2. An illustration of the overall architecture. Each residual module finds auxiliary actions for compensating model mismatches in corresponding nominal modules. The residual footstep module is learned using RL in simulation, while RD data is simultaneously collected and fed into the nominal controller during the learning process. The RD module is then reconstructed into a neural network for real-world deployment.

To address this, we first introduce an RL-based residual module to compensate the heuristic footstep planner, enabling adaptive foothold adjustment. In parallel, we compute discrepancies in the nominal continuous dynamics analytically using simulation data. These discrepancies are then reconstructed through SL, using proprioceptive sensor data history, to ensure reliable performance in real-world. This approach enhances sample efficiency by reducing the number of variables optimized during the RL process without compromising performance compared to other hybrid methods.

B. Nominal Hybrid Dynamics

This section explains a hybrid dynamics model of legged system and corresponding residual modules. The discrete hybrid dynamics can be expressed as follows [34].

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \quad \mathbf{x} \notin S \tag{1}$$

$$\mathbf{x}^+ = \Delta(\mathbf{x}^-), \quad \mathbf{x} \in S \tag{2}$$

where $\mathbf{x}$ is a state; $\mathbf{u}$ is a ground reaction force (GRF); S is a switching set which includes every moment of contact; $\mathbf{f}(\mathbf{x}, \mathbf{u})$ is a continuous dynamics; and $\Delta(\mathbf{x})$ is a switching dynamics. Throughout this paper, vector-valued variables are denoted in boldface, while matrix-valued variables are represented using calligraphic font. When contact states do not change, the system follows the continuous dynamics as follows.

$$\mathbf{x} = [\mathbf{p},\ \boldsymbol{\phi},\ \dot{\mathbf{p}},\ \boldsymbol{\omega}] \in \mathbb{R}^{12}, \tag{3}$$

$$\dot{\mathbf{x}} = \mathbf{f}_n(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \dot{\mathbf{p}} \\ \mathcal{M}(\boldsymbol{\phi}_0)\boldsymbol{\omega} \\ \frac{1}{M}\sum_{i=1}^{4} \mathbf{u}_i + \mathbf{g} \\ \mathcal{I}_{\mathbb{W}}^{-1}\sum_{i=1}^{4} (\mathbf{p}_{f,i} \times \mathbf{u}_i) \end{bmatrix}, \tag{4}$$

where $\mathbf{f}_n$ is a nominal continuous dynamics; $\mathbf{p}$ is a position of center of mass (CoM); $\mathbf{p}_f$ is a foothold; $\boldsymbol{\phi}$ is Euler angle; $\boldsymbol{\omega}$ is angular velocity of CoM expressed in the body frame $\mathbb{B}$; $\mathcal{M}(\boldsymbol{\phi}_0)$ is a mapping matrix to transform angular velocity into Euler angle rate at operating point $\boldsymbol{\phi}_0$; M is a lumped mass of the robot; $\mathbf{g}$ is a gravity; $\mathcal{I}_{\mathbb{W}}$ is an inertia matrix in world frame $\mathbb{W}$ with a following relationship $\mathcal{I}_{\mathbb{W}} = \mathcal{R}(\boldsymbol{\phi}_0)\mathcal{I}_{\mathbb{B}}\mathcal{R}(\boldsymbol{\phi}_0)^T$; and $\mathcal{R}(\cdot)$ is a rotation matrix.

In this work, we choose a SRBM-based convex MPC as a nominal controller in [3]. This framework is well-established in many research and has some favorable features. For instance, the convex optimization formulation is advantageous for massive learning procedure by ensuring fast computation time by exploiting state-of-the-art solvers.

C. Residual Footstep Module

For simplification, we assume that footholds instantaneously change whenever contact state changes. Therefore, we hierarchically define the footstep variables in a separate heuristic-based planner as suggested in [2]. The foothold and gait pattern of the i-th leg are defined as follows.

$$\mathbf{p}_{f,i}^{+} = \mathbf{p}_{f,i}^{-} + \delta\mathbf{p}_{f,heuristic,i}, \quad \mathbf{x} \in S \tag{5}$$

$$\Phi_{k,i} = \Phi_{k-1,i} + \frac{\delta t}{T_{step,i}}, \tag{6}$$

TABLE I
LIST OF DOMAIN RANDOMIZATION

Parameter	Notation	Range
Command velocity	$[v_x^*, v_y^*, \omega_z^*]$	$\pm[2.0, 1.0, 1.0]$ $m(rad)/s$
Friction coefficient	μ	$[0.4,\ 1.0]$
Payload	$M_{payload}$	$[-1.0,\ 5.0]$ kg
Bumpiness	h_{env}	$[0, 0.1]$ m
CoM position	$\delta p_{CoM,xyz}$	± 0.05 m
Initial rotation	ϕ_{init}	± 15 deg
Initial height*	δz_{init}	$[0, 0.1]$ m

*deviation from nominal height of CoM.

where $\delta \mathbf{p}_{f,heuristic}$ are reference foothold from the heuristic planner; $\Phi \in [0, 1]$ is a gait phase parameter; δt is a control time step; and T_{step} is a footstep time indicating swing and stance phases. For example, Φ for a leg initialized in stance starts at zero and increments by $\delta t / T_{stance}$ each control cycle. When Φ reaches 1, it is reset to zero, and T_{step} is set to T_{swing}, repeating the same procedure.

Since the heuristic footstep planner comprises a fixed gait pattern and simplified model, which cannot fully reflect actual dynamics such as the inertial effect or non-flat terrain, the residual footstep module compensates it as follows.

$$\mathbf{p}_{f,i}^{+} = \mathbf{p}_{f,i}^{-} + \delta \mathbf{p}_{f,heuristic,i} + \delta \mathbf{p}_{f,res,i}, \quad \mathbf{x} \in S \tag{7}$$

$$\Phi_{k,i} = \Phi_{k-1,i} + \frac{\delta t}{T_{step,i}} + \delta \Phi_{res,k-1,i}, \tag{8}$$

where $\delta \mathbf{p}_{f,res}$, $\delta \Phi_{res}$ are residual footstep and phase.

The residual footstep module is learned using RL. The nominal controller employs a heuristic footstep planner that assumes flat terrain and no-slip conditions, which cannot fully handle more complex scenarios, such as rough or stepped terrain with varying friction coefficients. To address these limitations, we train the modules under three terrain types (flat, rough, stepped) with varying friction conditions. Additionally, we randomize system and environmental parameters at each reset, as detailed in Table I, and initialize the robot at random body angles and heights slightly above the ground to further enhance robustness.

The reward function is designed to be similar to the cost function from MPC so that the module can operate as intended by nominal controller. Because the nominal controller can compute the walking pattern and required control inputs, it reduces the initial search space for learning, allowing the reward function to be simply designed as follows.

$$r_{total} = r_{alive} + c_1 ||\mathbf{x}^* - \mathbf{x}|| + c_2 ||\boldsymbol{\tau}||, \tag{9}$$

$$r_{alive} = \begin{cases} -10, \text{ if robot fails} \\ 0, \text{ otherwise} \end{cases} \tag{10}$$

where r_{alive} is a penalty whenever the system fails; $\mathbf{x}^*$ is a user-defined desired state; $\boldsymbol{\tau}$ is a joint torque; and c_i is a weight for each reward.

The training is performed using proximal policy optimization (PPO) [35], resulting in 16-dimensional output including residual foothold and phase of all legs. Details of the PPO setup are provided in Table II. The observation $\mathbf{o}_{RF}$ for the residual

TABLE II
PPO HYPERPARAMETER FOR RESIDUAL FOOTSTEP

Parameter	Value
Network size (actor & critic)	$[256, 128]$
Activation function	LeakyReLU
# of environments	100
# of environment steps per update	200
# of batches	4
# of epochs	4
Learning rate	0.0005
Discount factor	0.996
GAE	0.95
Clip range	0.2

footstep network $\boldsymbol{\pi}_{RF}$ is constructed as follows.

$$[\delta \mathbf{p}_{f,res,k}, \delta \Phi_{res,k}] = \boldsymbol{\pi}_{RF}(\mathbf{o}_{RF,k}), \tag{11}$$

$$\mathbf{o}_{RF,k} = [\dot{\mathbf{p}}_k^*, \dot{\boldsymbol{\omega}}_k^*, \boldsymbol{\phi}_k, \boldsymbol{\omega}_k, \{\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\}_{j,\{k,\ldots,k-h\}}, \delta \mathbf{p}_{f,heuristic}, \Phi_{k,i}, \boldsymbol{\eta}, \boldsymbol{\tau}_k] \in \mathbb{R}^{113}, \tag{12}$$

where $\dot{\mathbf{p}}^*, \dot{\boldsymbol{\omega}}^*$ is a command velocity; $\boldsymbol{\theta}_{j,\{k,\ldots,k-h\}}$ is a history of joint positions with a window size h; $\boldsymbol{\eta}$ is a Boolean vector representing the planned contact sequence for each leg from the nominal gait planner; and $\boldsymbol{\tau}$ is a joint torque. Note that $h = 2$ is used in this paper.

D. Residual Dynamics Module

The nominal model $\mathbf{f}_n(\mathbf{x}, \mathbf{u})$ in (4) contains rotational dynamics which is locally time-invariant and linearized based on Euler angle. The state is then predicted using numerical forward Euler integration through the control horizon.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \delta t \mathbf{f}_n(\mathbf{x}_k, \mathbf{u}_k), \tag{13}$$

In fact, this model does not fully account for actual whole-body dynamics or external disturbances, mainly due to the impact in switching phase, which can lead to performance degradation. This model discrepancy, $\mathbf{f}_{res}$, can be determined using the history of states and control input as follows.

$$\mathbf{f}_{res,k-1} = \delta t^{-1}(\mathbf{x}_k - \mathbf{x}_{k-1}) - \mathbf{f}_n(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}). \tag{14}$$

To deploy this term in real-world applications, several challenges must be addressed. First, since it is used directly as input to the MPC, excessive fluctuations in its value can significantly compromise the performance of the controller. Furthermore, while representing RD as a function of the MPC state and control input could enable more accurate dynamics prediction and potentially allow its use within the MPC optimization horizon [21], this approach would substantially increase computational burden for the RL process.

To mitigate these issues, we design a low-pass IIR filter:

$$\hat{\mathbf{f}}_{res,k} = e^{\frac{-2\pi F_c}{F_s}} \hat{\mathbf{f}}_{res,k-1} + (1 - e^{\frac{-2\pi F_c}{F_s}}) \mathbf{f}_{res,k}, \tag{15}$$

where F_c, F_s are cutoff and sampling frequencies. We set $F_c = 10$ Hz, $F_s = 1$ kHz in this work. The filter is designed to capture only the dominant low-frequency components of uncertainties,

such as payloads or external disturbances, that can be considered quasi-static within each MPC control loop. Additionally, it can prevent the system from being too sensitive to noisy signal.

Nevertheless, the analytical RD in (15) still require accurate information, including linear velocity, contact states and GRF. Estimating this information in a real-world environment is challenging, and the resulting inaccurate measurement of RD can lead to catastrophic failure of the system.

A possible solution is to train a neural network after collecting data from simulations or experiments [19]. We first collect simulation data to accurately label the RD while training the residual footstep module. After training the footstep module, we reconstruct data set of RD and corresponding observations into an neural network model, $\boldsymbol{\pi}_{RD}$, to predict the residual term using only directly measurable proprioceptive sensor data, such as IMU and joint encoders. This model can then be trained using SL.

$$\hat{\mathbf{f}}_{res,k} = \boldsymbol{\pi}_{RD}(\mathbf{o}_{RD,k}, \dots, \mathbf{o}_{RD,k-h}), \tag{16}$$

$$\mathbf{o}_{RD,k} = [\boldsymbol{\phi}_k, \boldsymbol{\omega}_k, \boldsymbol{\theta}_{j,k}, \dot{\boldsymbol{\theta}}_{j,k}, \boldsymbol{\tau}_k, \hat{\mathbf{f}}_{res,k-1}] \in \mathbb{R}^{54}. \tag{17}$$

This network is designed with the same structure as the residual footstep module. We randomly collected a total of 10 million data to train this module. This regression improves robustness in a real-world for inferring RD, compared to relying solely on an analytical design.

With aforementioned dynamics and footstep planning, the MPC tries to solve the following optimization problem to obtain control input.

$$\underset{\mathbf{X},\mathbf{U}}{\text{minimize}} \quad \sum_{i=0}^{N-1} l(\mathbf{x}_{k+i}, \mathbf{u}_{k+i}) \tag{18}$$

$$\text{subject to} \quad \dot{\mathbf{x}}_{k+i+1} = \mathbf{f}_n(\mathbf{x}_{k+i}, \mathbf{u}_{k+i}) + \hat{\mathbf{f}}_{res,k}, \tag{19}$$

$$\mathbf{X} \in \mathbb{X},\ \mathbf{U} \in \mathbb{U} \tag{20}$$

where $\mathbf{X}, \mathbf{U}$ is a set of states and control inputs through the horizon N; $l(\mathbf{x}, \mathbf{u})$ is a cost function to minimize; and $\{\mathbb{X}, \mathbb{U}\}$ is a feasible domain of state and control input. Note that the current residual term $\hat{\mathbf{f}}_{res,k}$ is provided by the RD module and is independent of the state and control input. Consequently, it is applied uniformly across the entire prediction horizon at each control iteration. This filtering-based decoupling enables fast computation of MPC during both learning and real-world implementation.

The cost function is formulated as a least-square form.

$$l(\mathbf{X}, \mathbf{U}) = \mathbf{X}^T diag(\mathbf{w}_\mathbf{x})\mathbf{X} + \mathbf{U}^T diag(\mathbf{w}_\mathbf{u})\mathbf{U}, \tag{21}$$

$$\mathbf{w}_\mathbf{x} = [\mathbf{w}_\mathbf{p}, \mathbf{w}_{\boldsymbol{\phi}}, \mathbf{w}_\mathbf{v}, \mathbf{w}_{\boldsymbol{\omega}}], \tag{22}$$

where $diag(\boldsymbol{\zeta})$ is a diagonal matrix consisting of diagonal vector $\boldsymbol{\zeta}$; and $\mathbf{w}_{(\cdot)}$ is a weight for each variable.

The feedforward and feedback joint torques are computed via leg kinematics as follows [3].

$$\boldsymbol{\tau}_{ff,i} = \mathcal{J}(q)_i^T \mathbf{u}_i, \tag{23}$$

$$\boldsymbol{\tau}_{fb,i} = \mathcal{J}(q)_i^T [\mathcal{K}_P(\mathbf{p}^*_{f,i} - \mathbf{p}_{f,i}) + \mathcal{K}_D(\dot{\mathbf{p}}^*_{f,i} - \dot{\mathbf{p}}_{f,i})], \tag{24}$$

TABLE III
EXPERIMENTAL RESULT OF ROBUSTNESS TEST

Name	Normal (A)	With payload (B)	With disturbance (C)
vanilla-MPC	0.0216	*Failed*	*Failed*
res-dyn	0.0118	0.0538	*Failed*
res-all	0.0212	0.0197	0.0695

Values indicate RMS error of roll-angle.

$$\boldsymbol{\tau}_i = \boldsymbol{\tau}_{ff,i} + \boldsymbol{\tau}_{fb,i}, \tag{25}$$

where $\mathcal{J}_i$ is a foot Jacobian, $\mathbf{p}^*_{f,i}$ is a reference foot position from footstep planner, and $\mathcal{K}_{P,D}$ are Cartesian PD gains.

IV. RESULTS

In this section, we conduct comparative experiments to evaluate the advantages of the proposed framework in terms of command tracking and robustness against uncertainties, compared to baseline controllers. These experiments assess the system's ability to reliably handle a range of disturbances and model errors, such as kicks or heavy payloads.

Additionally, we examine whether the proposed method can reduce the control parameter dependency of the nominal MBA controller. We also conduct OOD tests to evaluate whether our framework maintains consistent performance under conditions that differ significantly from those encountered during simulation, in comparison to existing end-to-end RL method. At last, we evaluate how our residual module design affects learning efficiency and convergence compared to baseline methods.

A. Experimental Setup

We validate our framework through simulations and real-world experiments using *Unitree Go1*, a 12 kg quadrupedal robot with 12 DoF. The simulation environment is built using the physics simulator *RAISIM* [36]. MPC and residual modules are updated at 100 Hz, while the remaining low-level parts run at 1 kHz. Note that the MPC predicts the next 0.1 s with a time step of 0.01 s over 10 horizons at each control iteration. We exploit a linear Kalman filter and momentum-based contact detection for the state estimation. The nominal stance and swing times are set to be 0.3 s, corresponding to a trot gait.

B. Effect of Residual Modules

We first investigate the effects of each residual module. In this scenario, the robot is commanded to maintain its posture while external payload of 6 kg is exerted on its head with the following baselines: nominal MPC (*vanilla-MPC*), MPC with only a RD module (*res-dyn*), and MPC with both residual footstep and dynamics module (*res-all*). Due to the payload being applied far from the CoM, a significant model error is expected. If the robot successfully adapts to the given payload, it is subjected to external disturbances.

Table III and Fig. 3 illustrate the results. The time intervals of each condition (normal, with payload, with disturbance) are denoted in Fig. 3. As seen in Table III, *vanilla-MPC* fails to maintain its posture under the given payload.

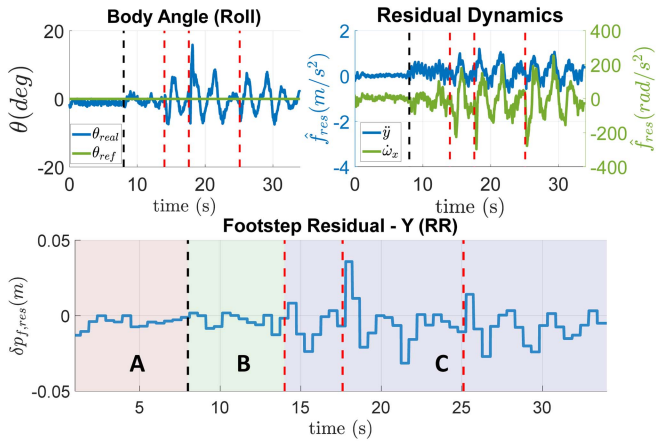

Fig. 3. An experimental result of robustness test with *res-all*. Each graph indicates the roll-angle (top-left), RD (top-right), and y-directional footstep residual of rear-right (RR) leg in the body frame (bottom). The black dotted line marks the moment when the payload is applied, while red lines indicate the disturbances.

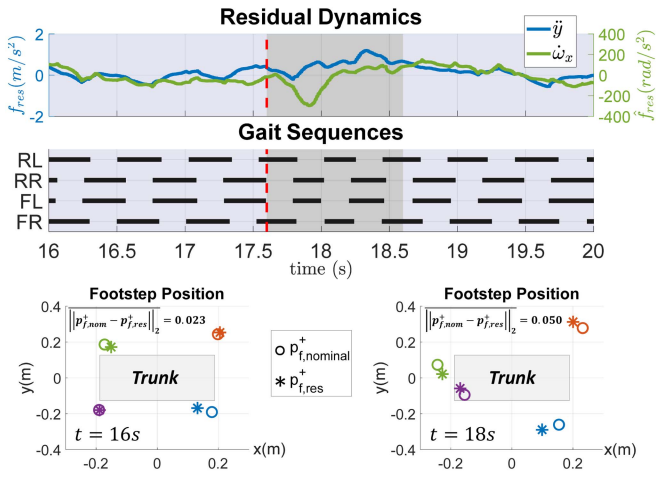

Fig. 4. An detailed results of robustness test using *res-all*. In the gait sequence plot, black lines indicate the stance phases of each corresponding leg, while the gray shaded area denotes the 1-second recovery period immediately following external disturbances applied to the robot. The footstep position graph shows the position of each footstep in a top-down view within the robot's body frame, where the gray rectangle in the graph represents the robot's trunk.

In contrast, *res-dyn* allows the robot to compensate for the model discrepancy once the payload is applied. However, over time, the robot exhibits a sagittal swaying motion. Because the gait pattern remains fixed, it cannot adapt to the resulting motion, leading to failure when additional disturbances are exerted.

On the other hand, *res-all* successfully controls locomotion under both the heavy payload and external disturbances. As seen in Fig. 3, the RD module captures model discrepancies whenever disturbances are exerted, while the residual footstep module modifies footholds to counteract the induced swaying motion.

Fig. 4 describes the detailed results from *res-all*. Without external disturbances, the footholds and phase from the residual footstep module remain close to the heuristic. However, when a disturbance generates angular momentum in the roll direction, the module reduces stance time and adjusts footholds to stabilize the motion.

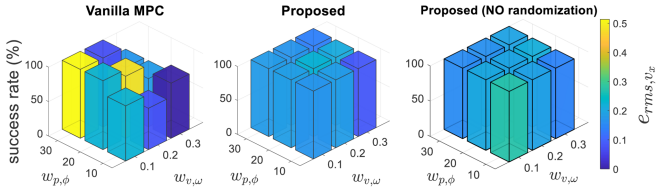

Fig. 5. Results of control parameter robustness test. Each graph indicates success rate (height) and RMS error (color) of velocity tracking from corresponding parameter setup with (1) *vanilla-MPC*, (2) proposed, and (3) proposed without randomization in simulation. Each weight label indicates that all corresponding weights (e.g., $\mathbf{w}_{\mathbf{p},\phi} = [\mathbf{w}_{\mathbf{p}x,y,z}, \mathbf{w}_{\phi x,y,z}]$) share the same value.

The proposed method is also evaluated under relatively high-speed locomotion, where the robot accelerated up to $1m/s$ in the x-direction. In this experiment, a reduction in gait phase is observed during forward acceleration, as shown in the supplementary video.

C. Robustness to Control Parameters

It is widely known that the performance of MPC is highly sensitive to control parameters, such as weight in the cost function [4]. Despite their critical role, these parameters are difficult to optimize directly and are often selected empirically to simplify the problem [24]. Similar to reward engineering in RL, this leads to a cumbersome tuning process. In this experiment, we investigate whether the residual modules can reduce this sensitivity, thereby improving robustness across a range of parameter configurations.

To evaluate this, the robot is tasked with velocity tracking on flat terrain under varying cost function weights. Specifically, during training, the weights in (21) are randomized within $\mathbf{w}_{\mathbf{p},\phi} \in [10, 30]$ and $\mathbf{w}_{\mathbf{v},\boldsymbol{\omega}} \in [0.1, 0.3]$. For testing, we set $\mathbf{w}_{\mathbf{p},\phi} = [10, 20, 30]$, $\mathbf{w}_{\mathbf{v},\boldsymbol{\omega}} = [0.1, 0.2, 0.3]$. For all cases, $\mathbf{w}_{\mathbf{u}}$ is fixed to 10^{-5}. Other system parameters are randomized as in Table I. At each setting, we run 10 simulations for 10 seconds each, and evaluate success rate and RMS velocity tracking error averaged over the entire 10 s time window.

As shown in Fig. 5, *vanilla-MPC* exhibits significant performance sensitivity to weights, while our method demonstrates consistent performance across all configurations. These results suggest that the residual modules effectively reduce dependency on manual tuning, allowing for successful task execution under a broader set of parameter settings.

In addition, we conducted an experiment without weight randomization during training. In this case, weights are fixed to $\mathbf{w}_{\mathbf{p},\phi} = 10$, $\mathbf{w}_{\mathbf{v},\boldsymbol{\omega}} = 0.3$. Fig. 5 indicates that performance remained relatively consistent even without randomization.

D. Consistent Performance Via MPC

Conventional end-to-end RL methods often struggle to handle situation that deviate from their training distribution [12], [13]. In contrast, many model-based controllers are capable of producing consistent motions across a wide range of states. We aim

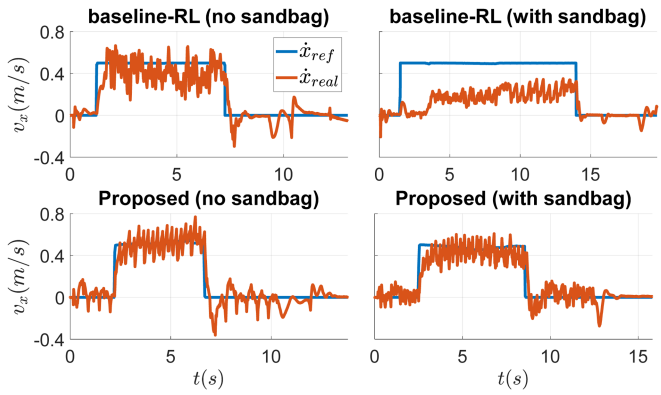

Fig. 6. An experimental result of out of distribution test.

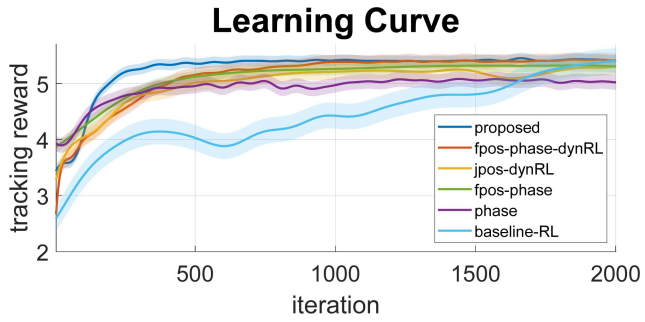

Fig. 7. Comparative results of learning efficiency.

to combine the consistency of MBA with the robustness of RL-based methods, and validate whether our proposed framework offers improved consistency compared to existing approaches.

For comparison, we design an end-to-end RL controller (*baseline-RL*) as a baseline [7]. This framework concurrently learns a state estimator to predict indirect states, such as translational velocity and contact probability, alongside the control policy. The policy is trained using PPO under same conditions and network structure in Table II.

The baseline is then compared with our method. For both cases, the robot is trained in the domain as shown in Table I. Note that the payload is only given on robot's trunk in simulation. In the experiment, a 3 kg payload is attached to both front and rear legs on the right side, creating an OOD situation from simulation. The robot is commanded to follow the given velocity command while maintaining its posture.

The experimental results are shown in Fig. 6. Without any payload, both *baseline-RL* and *res-all* successfully track the given velocity command. However, the velocity tracking performance of *baseline-RL* drops significantly with external payloads. In contrast, the proposed method maintains performance without notable degradation. These results indicate that integrating MBAs with residual modules enhances robustness and enables reliable motion, even in the presence of uncertainties out of training domain.

E. Comparative Analysis With Baselines

Our proposed methodology integrates different types of residual modules into the nominal controller. To analyze each module's contribution to performance, we conduct an ablation study with baseline methods.

TABLE IV
RESULT OF COMPARATIVE ANALYSIS

Name	$M_p = 1.75M$	$M_p = 2M$	$M_p = 2.25M$
res-all (proposed)	0.195 (76%)	**0.185 (74%)**	**0.174 (58%)**
fpos-phase-dynRL	0.200 **(88%)**	0.207 (72%)	0.196 (57%)
jpos-dynRL	**0.185** (64%)	0.189 (55%)	0.194 (3%)
fpos-phase	0.201 (71%)	0.220 (72%)	- (0%)
phase	0.277 (39%)	0.306 (11%)	0.312 (1%)
baseline-RL	0.805 (59%)	0.766 (40%)	0.800 (28%)
vanilla-MPC	0.314 (41%)	0.320 (25%)	- (0%)
resdyn-window	0.309 (39%)	0.293 (30%)	0.282 (14%)

Values indicate 'RMS velocity error (success rate)'. Lowest RMSs and highest success rates are denoted in bold.

In simulations, a robot is commanded to traverse terrain with varying roughness, procedurally generated using Perlin noise. The roughness level h_{env} is defined as the maximum height difference of the terrain. In each simulation, the robot is commanded to follow the given command $v_x^* = 1.0\ m/s$, $\omega_z^* = 1.0\ rad/s$. Each training is conducted with the setup in Table I, except for $M_{payload} = [0.9, 1.8]M$, $h_{env} = [0, 0.2]\ m$. The test environment consists of $h_{env} = 0.2\ m$ and $M_{payload} = [1.75, 2, 2.25]M$, respectively. We set success rate and RMS of velocity tracking error as criteria, and conduct 100 simulations for each combination of environment condition and controller.

For the ablation study, we evaluate a total of 8 controller configurations, each representing a different combination of the residual modules. For instance, *fpos-phase* only compensates footstep position and phase. The baselines include methods from Chen et al. [29], which use RL to find auxiliary actions in the joint space (*jpos*) and dynamics space (*dynRL*); Yang et al. [23], who proposed a hierarchical framework with a learned gait transition module (*phase*); and the online adaptation strategy from [16], which regresses model discrepancies within a sliding window (*resdyn-window*).

As summarized in Table IV, our method consistently achieves a high success rate and stable tracking performance across different environmental conditions, while most baseline methods exhibit greater fluctuations in performance depending on the level of uncertainty.

The analysis revealed specific weaknesses in baselines. For example, the success rate of *fpos-phase* drops sharply under large uncertainties, highlighting the necessity of dynamics compensation. While *fpos-phase-dynRL* shows performance comparable to our method, our framework exhibits improved learning efficiency, which will be discussed later. Conversely, *baseline-RL* adopt an overly conservative strategy, refusing to follow commands under high uncertainty to avoid falling. Furthermore, the poor performance of *phase* and *resdyn-window* highlights that compensating for inaccuracies in both footsteps and dynamics is crucial for robustness.

Lastly, we examine the impact of each residual module on learning efficiency during training. As depicted in Fig. 7, our proposed method not only achieves the highest tracking reward but also demonstrates the stable and fastest convergence. This suggests that the proper combination of residual modules with

nominal MBA leads to more refined locomotion, enabling more sample-efficient learning.

V. Conclusion

In summary, we propose a novel hybrid framework that combines the strengths of both MBAs and LBAs. The residual modules are designed to compensate for the limitations of heuristics in each part of the conventional MBA. The benefits of the proposed framework are validated through both simulations and hardware experiments.

We observe that our framework can produce more adaptive locomotion while still providing consistent performance, even beyond the training domain. Additionally, our approach can alleviate the hyperparameter sensitivity of nominal MBA. Furthermore, our method demonstrates improved learning efficiency compared to the baselines.

However, to fully validate the general applicability of our method, broader evaluations are necessary under various conditions. We expect that with further module design and training in more diverse domains, the proposed framework can be extended to handle a wider range of uncertainties, for example, uncertain terrain properties such as softness, compliance, and slipperiness.

References

[1] M. H. Raibert, *Legged Robots That Balance*. Cambridge, MA, USA: MIT Press, 1986.

[2] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *Proc. 6th IEEE-RAS Int. Conf. Humanoid Robots*, 2006, pp. 200–207.

[3] J. D. Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–9.

[4] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. D. Prete, "Optimization-based control for dynamic legged robots," *IEEE Trans. Robot.*, vol. 40, pp. 43–63, 2023.

[5] D. Kim, J. D. Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," 2019, *arXiv:1909.06586*.

[6] S. Hong, J.-H. Kim, and H.-W. Park, "Real-time constrained nonlinear model predictive control on SO(3) for dynamic legged locomotion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 3982–3989.

[7] G. Ji, J. Mun, H. Kim, and J. Hwangbo, "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 4630–4637, Apr. 2022.

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

[9] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Robust and versatile bipedal jumping control through multi-task reinforcement learning," in *Proc. Robotics: Sci. Syst.*, Jul. 2023, doi: 10.15607/RSS.2023.XIX.052.

[10] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 11443–11450.

[11] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," in *Proc. Robot.: Sci. Syst.*, Jul. 2017, doi: 10.15607/RSS.2017.XIII.048.

[12] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "RMA: Rapid motor adaptation for legged robots," in *Proc. Robot.: Sci. Syst.*, Jul. 2021, doi: 10.15607/RSS.2021.XVII.011.

[13] W. Yu, J. Tan, Y. Bai, E. Coumans, and S. Ha, "Learning fast adaptation with meta strategy optimization," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 2950–2957, Apr. 2020.

[14] L. Smith, I. Kostrikov, and S. Levine, "A Walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning," 2022, *arXiv:2208.07860*.

[15] M. V. Minniti, R. Grandia, F. Farshidian, and M. Hutter, "Adaptive CLF-MPC with application to quadrupedal robots," *IEEE Robot. Automat. Lett.*, vol. 7, no. 1, pp. 565–572, Jan. 2022.

[16] Y. Sun et al., "Online learning of unknown dynamics for model-based controllers in legged locomotion," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 8442–8449, Oct. 2021.

[17] E. Arcari et al., "Bayesian multi-task learning MPC for robotic mobile manipulation," *IEEE Robot. Automat. Lett.*, vol. 8, no. 6, pp. 3222–3229, Jun. 2023.

[18] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 3363–3370, Oct. 2019.

[19] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza, "NeuroBEM: Hybrid aerodynamic quadrotor model," in *Proc. Robot.: Sci. Syst. XVII, Robot.: Sci. Syst. Found.*, Jul. 2021, doi: 10.15607/RSS.2021.XVII.042.

[20] K. Y. Chee, T. Z. Jiahao, and M. A. Hsieh, "KNODE-MPC: A knowledge-based data-driven predictive control framework for aerial robots," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 2819–2826, Apr. 2022.

[21] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time neural MPC: Deep learning model predictive control for quadrotors and agile robotic platforms," *IEEE Robot. Automat. Lett.*, vol. 8, no. 4, pp. 2397–2404, Apr. 2023.

[22] Z. Xie, X. Da, B. Babich, A. Garg, and M. V. d. Panne, "GLiDE: Generalizable quadrupedal locomotion in diverse environments with a centroidal model," in *Proc. Algorithmic Found. Robot. XV*, Cham, Springer International Publishing, 2023, pp. 523–539.

[23] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots, "Fast and efficient locomotion via learned gait transitions," in *Proc. 5th Conf. Robot Learn.*, 2022, pp. 773–783.

[24] Y. Song and D. Scaramuzza, "Policy search for model predictive control with application to agile drone flight," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2114–2130, Aug. 2022.

[25] Y. Yang, X. Meng, W. Yu, T. Zhang, J. Tan, and B. Boots, "Continuous versatile jumping using learned action residuals," in *Proc. 5th Annu. Learn. Dyn. Control Conf.*, 2023, pp. 770–782.

[26] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "RLOC: Terrain-aware legged locomotion using reinforcement learning and optimal control," *IEEE Trans. Robot.*, vol. 38, no. 5, pp. 2908–2927, Oct. 2022.

[27] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "TossingBot: Learning to throw arbitrary objects with residual physics," *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 1307–1319, Aug. 2020.

[28] G. Bellegarda, C. Nguyen, and Q. Nguyen, "Robust quadruped jumping via deep reinforcement learning," *Robot. Auton. Syst.*, vol. 182, 2024, Art. no. 104799.

[29] Y. Chen and Q. Nguyen, "Learning agile locomotion and adaptive behaviors via RL-augmented MPC," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 11436–11442.

[30] 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 Robot. Automat. Lett.*, vol. 8, no. 10, pp. 6619–6626, Oct. 2023.

[31] D. Youm, H. Jung, H. Kim, J. Hwangbo, H.-W. Park, and S. Ha, "Imitating and finetuning model predictive control for robust and symmetric quadrupedal locomotion," *IEEE Robot. Automat. Lett.*, vol. 8, no. 11, pp. 7799–7806, Nov. 2023.

[32] F. Jenelten, J. He, F. Farshidian, and M. Hutter, "DTC: Deep tracking control," *Sci. Robot.*, vol. 9, no. 86, 2024, Art. no. eadh5401.

[33] Y. Fuchioka, Z. Xie, and M. V. D. Panne, "OPT-Mimic: Imitation of optimized trajectories for dynamic quadruped behaviors," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 5092–5098.

[34] E. Westervelt, J. Grizzle, and D. Koditschek, "Hybrid zero dynamics of planar biped walkers," *IEEE Trans. Autom. Control*, vol. 48, no. 1, pp. 42–56, Jan. 2003.

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

[36] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 895–902, Apr. 2018.