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

Efficiently Learning Robust Torque-based Locomotion Through Reinforcement with Model-Based Supervision

Yashuai Yan*,[1], Tobias Egle*,[2], Christian Ott[2,3], and Dongheui Lee[1,3]

***Abstract*—We propose a control framework that integrates model-based bipedal locomotion with residual reinforcement learning (RL) to achieve robust and adaptive walking in the presence of real-world uncertainties. Our approach leverages a model-based controller—comprising a Divergent Component of Motion (DCM) trajectory planner and a whole-body controller—as a reliable base policy. To address the uncertainties of inaccurate dynamics modeling and sensor noise, we introduce a residual policy trained through RL with domain randomization. Crucially, we employ a model-based oracle policy, which has privileged access to ground-truth dynamics during training, to supervise the residual policy via a novel supervised loss. This supervision enables the policy to efficiently learn corrective behaviors that compensate for unmodeled effects without extensive reward shaping. Our method demonstrates improved robustness and generalization across a range of randomized conditions, offering a scalable solution for sim-to-real transfer in bipedal locomotion.**

***Index Terms*—bipedal locomotion, sim-to-real transfer, residual RL, model-based control.**

I. Introduction

LEARNING effective torque-based locomotion policies for bipedal robots remains a central challenge in robotics. While torque control offers low-level access to a robot's actuators—enabling responsive, energy-efficient, and compliant behaviors—it also demands precise handling of complex, nonlinear dynamics, and can lead to unstable behaviors if not carefully regulated, particularly in high-dimensional floating-base systems. This complexity makes direct policy learning at the torque level particularly difficult, especially in the presence of real-world factors such as contact uncertainty, sensor noise, and actuation delays.

Traditionally, bipedal locomotion has been addressed through model-based control methods [1]–[5], which decompose the task into high-level trajectory planning and low-level whole-body control strategies. These controllers depend on accurate models of the robot's dynamics and reliable state estimation to achieve stable walking motions. While model-based approaches often incorporate robustness to model uncertainties and external disturbances [6], their performance can degrade when discrepancies between the model and the physical system become substantial, or when the state estimation is significantly affected by sensor noise. In practice, unmodeled dynamics such as joint friction, actuator nonlinearities, backlash, and structural flexibilities combined with noisy sensory data, can challenge control accuracy and consistency on the real robot.

An alternative strategy to address this challenge is iterative learning control. For example, Hu et al. [7] proposed learning a compensatory Zero-Moment Point (ZMP) trajectory by observing and correcting ZMP errors over repeated executions. This approach refines the reference trajectory to mitigate the impact of unmodeled dynamics during the pattern generation stage. However, it relies on repetitive, consistent motion patterns to accumulate corrections, which limits its applicability in dynamic or highly variable tasks.

To address these challenges, recent advances in deep reinforcement learning (DRL) [8]–[10] have demonstrated promising results. By training policies with domain randomization in simulation, DRL methods [11]–[14] can produce controllers that are robust to modeling errors and environmental variations. Nonetheless, purely learning-based approaches face limitations: they often require extensive data, are sensitive to reward design, and lack the interpretability and safety assurances of model-based methods.

To bridge this gap, recent works have explored hybrid strategies that combine the advantages of model-based and data-driven methods. For instance, Duan et al. [15] integrated the knowledge of the robot system into DRL to learn a task-space policy rather than joint-level controllers. This high-level policy is then connected with a low-level inverse dynamics controller to command joint torques to the robot. Similarly, Castillo et al. [16] incorporated insights from an angular momentum-based linear inverted pendulum model and designed a hierarchical framework in which the RL learns to generate high-level trajectories, followed by a low-level task-space tracking controller. Besides task-space learning, Egle et al. [17] applied RL to learn step location and timing adaptation to enhance the robustness of model-based controllers against strong external disturbances. While these approaches reduce reward engineering and improve sample efficiency, they still rely solely on reward signals for policy learning—requiring a number of trial-error iterations and offering limited guidance during training.

In contrast to standard approaches that rely solely on reward shaping, we propose a supervised reinforcement learning framework that directly incorporates an additional supervised loss term into the training objective. Our control architecture combines the structured reliability of model-based controllers with the adaptability of reinforcement learning. Specifically,

Manuscript received: June 15, 2025; Revised: December 3, 2025; Accepted: Febrary 5, 2026.

This paper was recommended for publication by Editor Abderrahmane Kheddar upon evaluation of the Associate Editor and Reviewers comments.

[1]Yashuai Yan and Dongheui Lee are with the Autonomous Systems Lab, TU Wien, 1040 Vienna, Austria {yashuai.yan, dongheui.lee}@tuwien.ac.at

[2]Tobias Egle and Christian Ott are with the Automation and Control Institute, TU Wien, 1040 Vienna, Austria {tobias.egle, christian.ott}@tuwien.ac.at

[3]Christian Ott and Dongheui Lee are also with the Institute of Robotics and Mechatronics (DLR), German Aerospace Center, Wessling, Germany.

*Contributed equally to the work.

Digital Object Identifier (DOI): see top of this page.

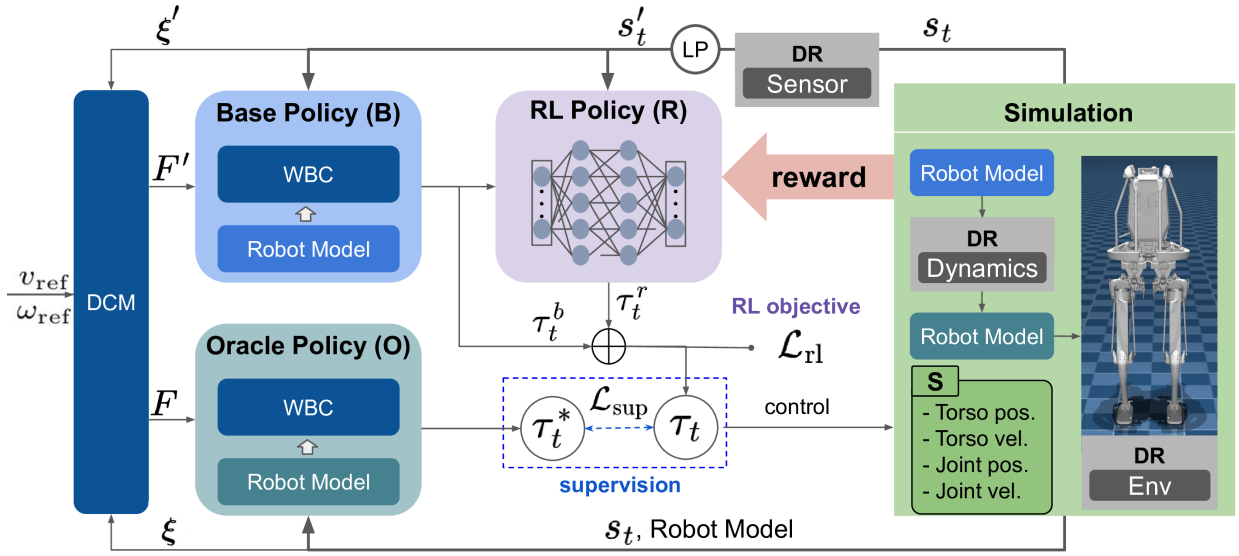

Fig. 1: **Residual Reinforcement Learning with Model-Based Supervision.** Our framework integrates three key components: the Base Policy, the Oracle Policy, and the Residual Policy. The Base and Oracle policies are model-based controllers; however, the Oracle, only used in training, has privileged access to true system information, including the robot model, state, and motor parameters, while the Base Policy operates under an inaccurate dynamics model with realistic assumptions without access to the true system information (LP: low-pass filter). The learnable Residual Policy is trained to compensate for the model inaccuracies of the Base Policy. Crucially, the Residual Policy is guided by both the RL objective $\mathcal{L}_{\text{rl}}$ and direct supervision from the Oracle Policy $\mathcal{L}_{\text{sup}}$, enabling efficient learning and improved robustness under real-world uncertainties.

we augment a model-based bipedal locomotion controller with a residual RL policy that learns to compensate for model inaccuracies. To expose the policy to real-world uncertainties during training, we propose to apply dynamics randomization and introduce a privileged Oracle policy that has full access to ground-truth parameters and generates ideal corrective actions under randomized conditions. Rather than learning solely from sparse or shaped rewards, the RL policy is explicitly guided by the Oracle through the supervised loss, which enhances learning efficiency and convergence speed. By integrating model-based priors with data-driven adaptation, our approach improves efficiency, robustness, and generalization across different bipedal robot platforms.

We summarize the key contributions of this work as follows:

- A control framework that integrates model-based control with data-driven reinforcement learning to learn residual corrective actions for mitigating model inaccuracies.
- A model-based oracle policy that leverages privileged information to compensate for system uncertainties and generate near-optimal control signals, which are used to supervise the learning process.
- A supervised reinforcement learning framework that incorporates Oracle actions into a supervised loss, complementing the standard RL objective and improving learning efficiency without relying exclusively on reward shaping.
- Comprehensive evaluations on three bipedal robots: Kangaroo, Unitree H1-2, and Bruce, demonstrating the robustness and generalization capabilities of our approach across different platforms.

II. Methodology

In this section, we introduce our method, which integrates the strengths of both reinforcement learning and model-based robot control algorithms. An overview is provided in Figure 1.

A. Model-based Robot Control

Our model-based bipedal locomotion controller is composed of two key components: a trajectory generator based on the Divergent Component of Motion (DCM) and an inverse dynamics-based whole-body controller (WBC).

1) DCM Trajectory Generation: The trajectory generation framework leverages the concept of three-dimensional DCM and the Virtual Repellent Point (VRP) [5], [18]. The DCM extends the notion of the capture point (CP) into three dimensions and is defined as:

$$\boldsymbol{\xi} = \boldsymbol{x} + b\dot{\boldsymbol{x}}. \tag{1}$$

where $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ represent the Center of Mass (CoM) position and velocity, respectively. $b = \sqrt{\Delta z/g}$ is derived from the average CoM height Δz and gravitational acceleration g. From the CoM dynamics $\ddot{\boldsymbol{x}} = \boldsymbol{g} + \boldsymbol{F}_{\text{ext}}/m$, we derive the DCM dynamics as:

$$\dot{\boldsymbol{\xi}} = \frac{1}{b}(\boldsymbol{\xi} - \boldsymbol{v}), \tag{2}$$

where $\boldsymbol{v} = \boldsymbol{x} - b^2/m\boldsymbol{F}$ denotes the VRP, which encodes the total force acting on the CoM, i.e., $\boldsymbol{F} = m\boldsymbol{g} + \boldsymbol{F}_{\text{ext}}$.

To generate walking trajectories, we plan a sequence of n preview VRP waypoints $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n\}$, which define $n-1$ transition phases alternating between single and double support phases. Within each phase ψ, over a time interval

Parameter	Unit	Distribution	Operator
Sensory Noise			
Torso position	m	$\mathcal{N}(0, 0.05\beta)$	additive
Torso rotation	rad	$\mathcal{N}(0, 0.05\beta)$	additive
Linear velocity	m/s	$\mathcal{N}(0, 0.1\beta)$	additive
Angular velocity	rad/s	$\mathcal{N}(0, 0.1\beta)$	additive
Joint position	rad	$\mathcal{N}(0, 0.05\beta)$	additive
Joint velocity	rad/s	$\mathcal{N}(0, 0.1\beta)$	additive
Dynamics Uncertainty			
Body mass	-	$\mathcal{U}(1-0.2\beta, 1+0.5\beta)$	scaling
Joint friction	-	$\mathcal{U}(1-0.5\beta, 1+0.1\beta)$	scaling
Joint damping	-	$\mathcal{U}(1-0.5\beta, 1+0.2\beta)$	scaling
Motor efficiency α_{decay}	-	$\mathcal{U}(1-0.2\beta, 1.0)$	scaling
Motor delay Δt_{delay}	ms	$\mathcal{U}(2, 4)$	additive
Environment			
Floor friction	-	$\mathcal{U}(0.5, 1.1)$	scaling

TABLE I: **Domain Randomization.** We capture variability from different sources and use a scaling factor β to control the randomization level during validation. For training, we use $\beta = 1$.

$t \in [0, T]$, the DCM trajectory is determined by solving (2) with a terminal constraint $\boldsymbol{\xi}_\psi(T)$. The solution is given by:

$$\boldsymbol{\xi}_\psi(t) = \alpha_\psi(t)\boldsymbol{v}_{\psi,0} + \beta_\psi(t)\boldsymbol{v}_{\psi,T} + \gamma_\psi(t)\boldsymbol{\xi}_\psi(T), \quad (3)$$

where the nonlinear coefficients $\alpha_\psi(t), \beta_\psi(t)$ and $\gamma_\psi(t)$ depend on the trajectory of the VRP from the start point $\boldsymbol{v}_{\psi,0}$ to the endpoint $\boldsymbol{v}_{\psi,T}$ [19]. We define equality constraints for the start and end points of adjacent transition phases to ensure the continuity of the trajectory. The complete trajectory is obtained by starting from a DCM endpoint and computing backward in time.

In addition to the DCM trajectory, our model-based method also plans the continuous foot trajectories. Similarly to prior work [20], we apply six-order polynomial functions to parameterize the foot trajectories by conditioning on the planned foot locations.

2) Whole-body Robot Control: To track the planned reference trajectories, we employ an inverse dynamics-based WBC [21]. A key objective of the controller is to stabilize the inherently unstable first-order DCM dynamics in (2). Therefore, we implement the following control law [5]:

$$\boldsymbol{v} = \boldsymbol{v}_{\text{ref}} + (\boldsymbol{I} + b\mathbf{K}_\xi)(\boldsymbol{\xi} - \boldsymbol{\xi}_{\text{ref}}), \quad (4)$$

where $\boldsymbol{v}_{\text{ref}}$ is the reference VRP, $\mathbf{K}_\xi$ is a positive definite diagonal gain matrix, and $\boldsymbol{e}_\xi = \boldsymbol{\xi} - \boldsymbol{\xi}_{\text{ref}}$ is defined as the DCM tracking error. From the output of the DCM controller (4) we can compute the desired force on the CoM as

$$\boldsymbol{F} = m/b^2(\boldsymbol{x} - \boldsymbol{v}), \quad (5)$$

which is commanded as a reference in the CoM task of the whole-body controller. Other tasks include foot trajectory tracking, reference tracking in joint or task space for the arms (if available), and maintaining a desired body orientation. Further details on the WBC formulation can be found in [21].

B. Enhance Robustness through Randomization

To address real-world variability, our method employs domain randomization (DR) during reinforcement learning, targeting three principal sources of uncertainty: sensors, robot dynamics, and environment. Sensor noise is simulated by perturbing the robot state inputs to the model-based controller. Dynamics variability is introduced by randomizing the physical properties of the robot, such as mass, inertia, motor strength, and actuation delay. Since detailed information on actuators and transmissions is not available for all platforms, we apply randomization at the joint level. The randomization ranges are chosen to cover plausible variations induced by the underlying actuators. Additionally, we vary ground friction coefficients and include ground unevenness to reflect the diversity of real-world terrain conditions. A comprehensive list of the randomized parameters used in our simulation environment is provided in Table I.

Algorithm 1 Learning Procedure

Init: RB, M, π_b, π_*, π_r, Δt_{sim}
while π_r not converged **do**
  for each episode **do**
    $M' \leftarrow DR(M)$, set simulator with M'
    $s'_0 \leftarrow DR(s_0)$
    sample $\Delta t_{\text{delay}}, \alpha_{\text{decay}}$
    for each step t **do**
      $\tau_t^b \leftarrow \pi_b(s'_t)$, $\tau_t^r \leftarrow \pi_r(s'_t)$
      $\tau_t \leftarrow \tau_t^b + \tau_t^r$
      simulate $\Delta t_{\text{sim}} - \Delta t_{\text{delay}}$ with $\alpha_{\text{decay}}\tau_t$
      $s'_{t+\Delta t_{\text{sim}}} \leftarrow DR(s_{t+\Delta t_{\text{sim}}-\Delta t_{\text{delay}}})$
      $\tau^b_{t+\Delta t_{\text{sim}}} \leftarrow \pi_b(M, s'_{t+\Delta t_{\text{sim}}-\Delta t_{\text{delay}}})$
      simulate Δt_{delay} with $\alpha_{\text{decay}}\tau_t$
      sample $\Delta t_{\text{delay}}, \alpha_{\text{decay}}$
      $\tau^*_{t+1} \leftarrow \pi_*(M', s_{t+\Delta t_{\text{sim}}})/\alpha_{\text{decay}}$
      $R_t \leftarrow$ reward
      RB $\leftarrow (s'_t, \tau^b_{t+1}, \tau^*_{t+1}, \tau^r_t, R_t, s'_{t+1})$
    end for
    for each update **do**
      sample $\mathcal{B}$ from RB
      compute $\mathcal{L}_{\text{rl}}, \mathcal{L}_{\text{sup}}$
      update policy and critic
    end for
  end for
end while

C. Model-based Supervision

Our framework incorporates the strength of model-based methods by explicitly utilizing knowledge of system uncertainties—information that is accessible with known randomization. To realize this, we introduce three key components: a Base policy, an Oracle policy, and a residual RL policy. The Base and Oracle policies are model-based controllers, while the residual policy learns to control robots with RL algorithms by interacting with the environment.

The Base policy, denoted as π_b, operates without access to the underlying randomization parameters. It operates on noisy state observations and uses an inaccurate robot model. In contrast, the Oracle policy, denoted as π_*, has complete information on the ground-truth randomization settings and the exact robot model within the simulator. This privileged information enables π_* to model uncertainties. For example, π_* has access to the real signal without noise and computes torques based on accurate robot dynamics. Additionally, π_* is aware of variations in motor characteristics—actuation delays Δt_{delay} and motor efficiency α_{decay}—and incorporates this

knowledge when computing torque commands. A detailed learning procedure is illustrated in Algorithm 1.

The oracle policy π_* thus serves as an idealized expert supervisor, leveraging its privileged access to ground-truth dynamics and system parameters to generate ideal control signals. Its role is to guide the residual policy in compensating for the limitations of the Base policy π_b, which arise from unmodeled dynamics and unknown variability. By learning to mimic the corrections provided by π_*, the residual policy enhances the overall control system's robustness and adaptability to real-world uncertainties.

D. Supervised Reinforcement Learning

To improve the robustness and adaptability of model-based control in the presence of real-world uncertainties, we formulate a residual RL problem. In this framework, the residual policy learns corrective actions on top of a model-based Base policy through interaction with a randomized simulation environment. The residual learning allows the agent to adapt to dynamics and disturbances that are difficult to model explicitly. In the following, we describe the design of the observation space and reward functions used to train the residual policy.

1) Observation Space: To enable the residual policy to infer latent dynamics and compensate for unmodeled variability, we design an observation space that captures both the instantaneous robot state and its temporal evolution. Specifically, we employ a recurrent neural network (RNN) to process the observation history $\boldsymbol{O}_{1:t} = [\boldsymbol{o}_1, \boldsymbol{o}_2, \cdots, \boldsymbol{o}_t]$, allowing the policy to leverage temporal patterns for more informed decision-making. At each timestep t, the observation vector $\boldsymbol{o}_t$ is defined as follows:

$$\boldsymbol{o}_t = \left[\boldsymbol{q}'_t, \dot{\boldsymbol{q}}'_t, \boldsymbol{\tau}^b_t, \boldsymbol{\tau}_{t-1}, \boldsymbol{e}'_{\xi,t}, \boldsymbol{e}'_{\text{foot},t}\right], \tag{6}$$

where $\boldsymbol{q}'_t$ and $\dot{\boldsymbol{q}}'_t$ denote the joint position and velocity after randomization, $\boldsymbol{\tau}^b_t$ is the torque output of the base policy, $\boldsymbol{\tau}_{t-1}$ is the last action, and $\boldsymbol{e}'_{\xi,t}$ and $\boldsymbol{e}'_{\text{foot},t}$ are the tracking errors of the DCM and foot trajectories from the Base policy. This observation reflects the noisy and uncertain sensory input available in real-world scenarios.

To provide richer observation during training, we also define a privileged observation used exclusively by the critic network. The privileged observation vector is given by:

$$\boldsymbol{o}^{\text{privi.}}_t = \left[\boldsymbol{o}_t, \boldsymbol{v}_t, \boldsymbol{\omega}_t, \boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{\tau}^*_t, \boldsymbol{e}_{\xi,t}, \boldsymbol{e}_{\text{foot},t}\right], \tag{7}$$

where $\boldsymbol{v}_t$ and $\boldsymbol{\omega}_t$ are linear and angular velocity of the floating base. $\boldsymbol{q}_t$ and $\dot{\boldsymbol{q}}_t$ are the true joint states, and $\boldsymbol{\tau}^*_t$ is the torque computed by the oracle policy π_*. Besides, the critic has access to the tracking errors $\boldsymbol{e}_{\xi,t}, \boldsymbol{e}_{\text{foot},t}$ from the Oracle policy. This privileged input enables more accurate value estimation during training, without being available to the policy during deployment.

2) Reward Functions: The reward functions used in this work are summarized in Table II. The primary components are tracking rewards that encourage the RL policy to follow the planned DCM and foot trajectories accurately. The torque tracking reward R_τ guides the applied torque $\boldsymbol{\tau} = \boldsymbol{\tau}^b + \boldsymbol{\tau}^r$ to match the output of the oracle policy $\boldsymbol{\tau}^*$, promoting consistency with optimal control behavior.

Reward	Expression	Distance	Parameter (w, λ)
Tracking			
R_ξ	$w \exp(-\lambda d)$	$d = \|\|\boldsymbol{e}_\xi\|\|_2$	$(20, 10)$
R_{foot}		$d = \|\|\boldsymbol{e}_{\text{foot}}\|\|_2$	$(5, 10)$
$R^{\text{rot}}_{\text{torso}}$		$d = \|\|\boldsymbol{e}^{\text{rot}}_{\text{torso}}\|\|_2$	$(1, 10)$
R_τ		$d = \|\|\boldsymbol{\tau} - \boldsymbol{\tau}^*\|\|_2$	$(5, 0.01)$
Regularization			
R_{smooth}		$d = \|\|\boldsymbol{\tau}_{t+1} - \boldsymbol{\tau}_t\|\|_2$	$(0.01, 0.01)$
Punishment			
$R_{\text{termination}}$	-20 if early terminated; 0 otherwise		

TABLE II: **Reward Functions.**

Additionally, R_{smooth} is included to promote smooth transitions in the control signals. Early termination is triggered when the DCM error $||\boldsymbol{e}_\xi||_2$ exceeds 0.2 meters, and this event is penalized through the $R_{\text{termination}}$ term.

Notably, our framework benefits from supervised signals provided by the oracle policy during training, which reduces reliance on extensive reward engineering and hyperparameter tuning typically required in conventional RL approaches.

E. Supervised Loss

To train the residual policy, we adopt the Proximal Policy Optimization (PPO) algorithm [22] with the objective $\mathcal{L}_{\text{rl}}$. Meanwhile, we leverage supervision from the Oracle policy and introduce an additional supervised loss term $\mathcal{L}_{\text{sup}}$ that encourages the policy to assign a higher likelihood to the oracle-corrected residual action. Specifically, the supervised loss is formulated as:

$$\mathcal{L}_{\text{sup}} = -\log(\pi_r(\boldsymbol{\tau}^*_t - \boldsymbol{\tau}^b_t | s_t)), \tag{8}$$

where $\boldsymbol{\tau}^*_t$ is the oracle torque and $\boldsymbol{\tau}^b_t$ is the base policy torque. This loss guides the policy toward imitating the oracle's corrective behavior in response to system uncertainties.

The total training objective combines the PPO loss and the supervised loss:

$$\mathcal{L}_{total} = \omega_{\text{rl}} * \mathcal{L}_{\text{rl}} + \omega_{\text{sup}} \mathcal{L}_{\text{sup}}, \tag{9}$$

where the weights $\omega_{\text{rl}} = 1, \omega_{\text{sup}} = 10$ balance the trade-off between imitation of the oracle policy and autonomous policy exploration.

III. Evaluation

We conduct comprehensive simulations on the different robots, and specifically, we validate the following questions:

- **Q1**: Can our Oracle policy serve as an effective supervisory signal under extensive randomization of robot dynamics and system-level uncertainties?
- **Q2**: Can our residual policy learn to compensate for the unmodeled variability that the base policy fails to address?
- **Q3**: Can our approach outperform standard RL methods that rely solely on reward signals or imitation learning that only learns from supervision?
- **Q4**: Can our framework be applied across different platforms without any parameter tuning?

	Success Rate ↑ (in %)			DCM Tracking ↓ (in cm)			Foot Tracking ↓ (in cm)			Return ↑ ($\times 10^3$)		
	Kangaroo	Bruce	H1-2	Kangaroo	Bruce	H1-2	Kangaroo	Bruce	H1-2	Kangaroo	Bruce	H1-2
Oracle (O)	**100.0**	**100.0**	**100.0**	3.05 ± 0.30	$\mathbf{1.49 \pm 0.07}$	$\mathbf{2.26 \pm 0.39}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	**27.86**	$\underline{28.16}$	**30.09**
BasePolicy (B)	0.0	0.0	0.0	12.73 ± 1.23	8.33 ± 1.66	12.42 ± 1.53	0.01 ± 0.01	0.02 ± 0.01	0.04 ± 0.03	2.40	2.53	2.56
ResRL (BR)	10.0	0.0	0.0	14.47 ± 1.39	10.18 ± 1.45	11.93 ± 1.43	69.51 ± 0.40	3.99 ± 3.22	7.61 ± 5.61	1.72	0.89	0.20
IL	**100.0**	70.0	80.0	3.86 ± 0.88	3.44 ± 1.30	3.65 ± 1.68	0.30 ± 0.04	0.70 ± 0.63	0.62 ± 0.67	24.9	23.45	25.96
MBC	0.0	0.0	0.0	23.45 ± 2.45	16.14 ± 2.06	22.01 ± 2.19	1.42 ± 0.74	3.05 ± 1.69	2.21 ± 0.94	0.72	0.45	0.63
Ours (OR)	96.7	80.0	**100.0**	4.64 ± 1.85	2.27 ± 0.78	$\underline{2.32 \pm 0.24}$	0.10 ± 0.02	0.54 ± 0.08	$\underline{0.27 \pm 0.04}$	21.94	26.17	$\underline{26.45}$
Ours (BOR)	**100.0**	**100.0**	**100.0**	$\mathbf{2.61 \pm 0.39}$	$\underline{1.57 \pm 0.16}$	2.39 ± 0.32	$\underline{0.08 \pm 0.01}$	$\underline{0.45 \pm 0.04}$	0.28 ± 0.08	$\underline{25.74}$	**28.22**	$\underline{26.45}$

TABLE III: **Comparison of different approaches**. All methods are evaluated with domain randomization ($\beta = 1$). The Oracle and Base in the first section indicate our Oracle and Base policies. The second section presents the performance of our baselines. We showcase the benefits of the supervision from our Oracle policy in the last section via the methods OR and BOR. OR indicates the the RL policy learns the direct troque commands, while BOR learns the residual torques of the Base policy.

	Success Rate		DCM Tracking		Foot Tracking	
level β	MBC	BOR	MBC	BOR	MBC	BOR
0.1	**100.0**	**100.0**	**3.55**	4.74	0.15	**0.08**
0.3	90.0	**100.0**	7.09	**3.94**	0.54	**0.08**
0.5	50.0	**100.0**	13.35	**3.33**	1.08	**0.07**
0.7	10.0	**100.0**	20.37	**2.98**	1.45	**0.07**

TABLE IV: **Randomization level analysis.** During the inference, we evaluate different domain randomization on Kangaroo by adjusting the parameter β, and compare their impacts on MBC and our method BOR. Larger β indicates more uncertainties in the system.

	Total Mass	Height	DoFs per Leg	Foot
Kangaroo	47 Kg	145cm	6	flat
Unitree H1-2	67Kg	178cm	6	flat
Bruce	4.8Kg	70cm	5	line

TABLE V: **Properties of different robots**.

A. Technical Implementation

In our actor-critic framework, the actor network is composed of a two-layer LSTM module followed by a linear output layer. Each LSTM layer contains 256 hidden units, and the final output layer has 512 neurons. The critic network is implemented using a multi-layer perceptron (MLP) with a 512-neuron input layer followed by two hidden layers of 256 neurons each. We employ Exponential Linear Units (ELU) [23] as the activation function for all hidden layers.

We train and evaluate our framework on three bipedal robots: Kangaroo [24], Unitree H1-2, and Bruce [25]. The details of the robots are illustrated in Table V. MuJoCo [26] is used as the physics simulator to simulate the robots. We use PIQP [27] to solve the QP for the whole-body controller, achieving a computation time of approximately 1 ms. While the simulation runs at 1000 Hz, control commands are applied at a frequency of 200 Hz. The RL policies are trained over $10k$ episodes with each episode corresponding to 10 seconds of simulation.

B. Baselines

1) Model-based Controller (MBC): We compare our method with a model-based controller. For a fair comparison, we apply the following steps to handle the unmodeled uncertainties in MBC:

- using a low-pass filter on robot state observations to mitigate the sensor noise;
- operating at a frequency of $1000Hz$.

2) Residual RL (ResRL): To enable a fair comparison with standard RL settings, we train an additional residual policy under the same conditions but without the supervised loss term in Eq. 9 (i.e., $(\omega_{\text{sup}} = 0)$). However, we retain the torque-tracking reward (R_τ), which continues to encourage the learned policy to follow the Oracle. This setup allows us to isolate and examine the impact of using an implicit reward signal versus an explicit loss formulation for incorporating Oracle guidance into the policy. For the same reason, we adopt a torque-based residual learning policy as our baseline, rather than the classical residual RL framework that learns joint positions with an underlying PD controller.

3) Imitation Learning (IL): Since our method shares the similar idea as imitation learning, in which the student (Base+RL policy) learns to mimic the teacher (Oracle policy) behavior, we exploit our framework for imitation learning and compare it with our method. For IL in our framework, we skip the RL objective (i.e. $\omega_{\text{RL}} = 0$ in Eq. 9) and only optimize the policy network based on supervision signal.

C. Quantitative and Qualitative Evaluation

1) Metrics: To compare with the baseline methods, we define the following evaluation metrics:

- **Success Rate**: measures whether the robot can walk successfully for 5 seconds while following random velocity commands. A trial is considered a failure if the robot falls.
- **DCM Tracking**: assesses the distance between the measured and desired DCM positions. Accurate DCM tracking is critical for stable walking; large DCM errors can lead to instability or falls.
- **Foot Tracking**: evaluates the accuracy of foot trajectory tracking, which reflects the robot's ability to follow the commanded velocity inputs.
- **Return**: represents the cumulative reward as defined in Table II, excluding the torque tracking reward R_τ to avoid biasing the evaluation in favor of the oracle policy. This metric captures overall controller performance, including aspects such as trajectory tracking accuracy and motion smoothness.

2) Comparison with Baselines: To evaluate the effectiveness of our methods in improving robustness and performance for torque-based locomotion, we compare our approach against baseline methods under domain randomization, as detailed in Table I. We validate the proposed key questions **Q1–Q4** and

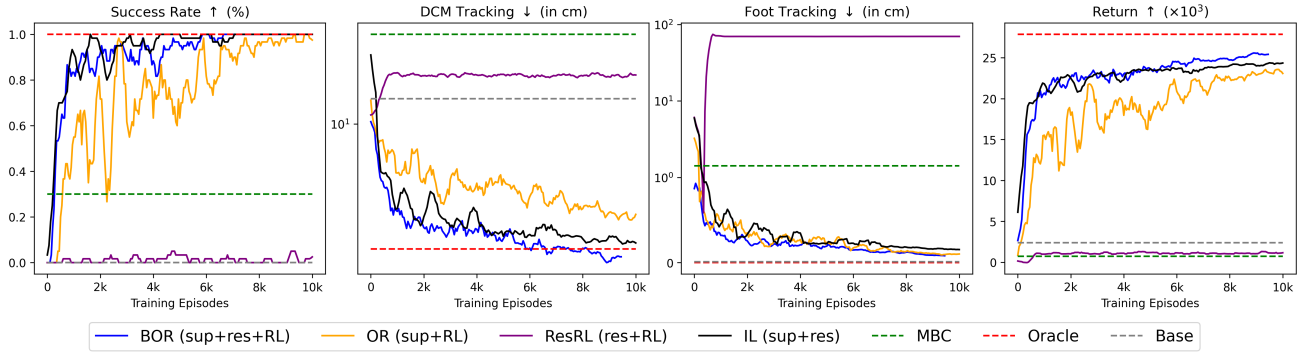

Fig. 2: **Quantitative evaluation on the Kangaroo robot.**We evaluate various methods using predefined metrics throughout training. Our method (BOR) shows that residual learning converges substantially faster than directly learning torque commands, even under identical Oracle supervision. Moreover, comparing BOR/OR with other baselines demonstrates that incorporating the supervision term into the optimization objective significantly improves training efficiency, bringing performance much closer to that of the Oracle policies.

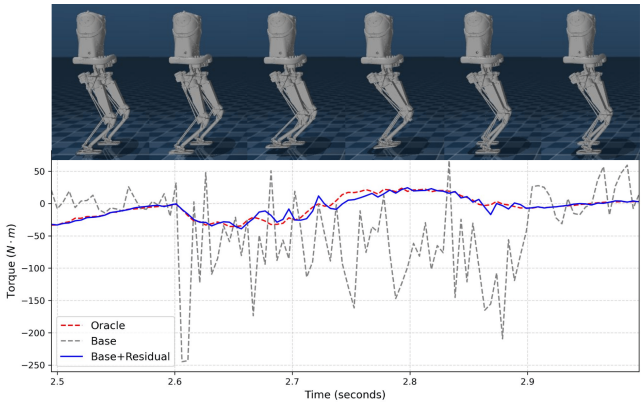

Fig. 3: **Torque tracking on the right hip pitch joint.** The Base policy produces noisy torques due to the domain randomization. Our learned residual policy succeeds in compensating the joint torques, closely tracking the Oracle policy.

compare our methods against the Oracle and Base policies and the baselines in Sec. III-B. For each approach and robot, we conduct 10 simulations, where robots are tasked with following commanded linear and angular velocities. Each simulation covers 5 seconds of walking, and we vary the velocity commands every 2 seconds. We early terminate the simulations if robots are falling.

Table III summarizes the results across the metrics introduced in Sec. III-C1. The Oracle policy (Algorithm 1) achieves the best performance, successfully completing all experiments with minimal tracking errors in both DCM and foot trajectories. This demonstrates its capacity to effectively supervise RL agents during training (**Q1**). In contrast, the Base policy—using the same model-based controller as the Oracle but without knowledge of system uncertainties—fails to execute the tasks, highlighting its brittleness under randomization.

Our method (BOR), shown in the last row, successfully compensates for the Base policy's limitations. After training with Oracle supervision, the residual policy approaches Oracle-level performance, even slightly outperforming it in DCM tracking for Kangaroo and Return metric for Bruce (**Q2**). Regarding **Q3**, our supervised residual RL method significantly outperforms standard residual RL, which frequently fails due to the challenges of learning torque-based locomotion for high degree-of-freedom robots—consistent with prior findings [28], [29]. However, our results also show that the performance of pure imitation learning (IL) varies across robotic platforms. While IL performs comparably to BOR on Kangaroo, it performs significantly worse on Bruce. This suggests that IL is more sensitive to the Oracle policy's reliability. For Bruce with line feet, although the Oracle has access to ground-truth system states, it sometimes results in unstable control during rapid transitions in commanded walking velocity. In such cases, RL helps explore corrective actions that deviate from the Oracle to improve stability and overall performance. Consequently, RL enhances both DCM tracking and foot tracking performance in BOR compared to IL. Finally, in response to **Q4**, our supervised residual RL framework consistently achieves near-Oracle performance across all three bipedal robots under the same training setup.

In addition, Table IV presents the impact of varying domain randomization levels on the performance of the model-based controller (MBC) and our proposed method (BOR). This analysis is conducted by adjusting the parameter β, as defined in Table I. At a low uncertainty level ($\beta = 0.1$), MBC achieves a 100% success rate, demonstrating its robustness under mild system perturbations. However, its performance degrades significantly as uncertainty increases, with failure rates rising to 50% at $\beta = 0.5$ and 90% at $\beta = 0.7$. In contrast, BOR—trained with the highest level of domain randomization ($\beta = 1$)—maintains a 100% success rate and exhibits minimal DCM and foot tracking errors across the entire range $\beta \in [0, 1]$, highlighting its superior robustness to system uncertainties.

Furthermore, we evaluate the training progress of the Kangaroo robot in Fig. 2. Thanks to the design of our Oracle policy and its supervision, our method efficiently learns the torque control, achieving over an 80% success rate after just 1,500 episodes—equivalent to 4.16 hours of simulation time in a single environment. In contrast, standard residual RL shows little to no progress during training, primarily due to the limited design of our reward functions. Our reward structure is intentionally much simpler than in prior work [8], and unlike

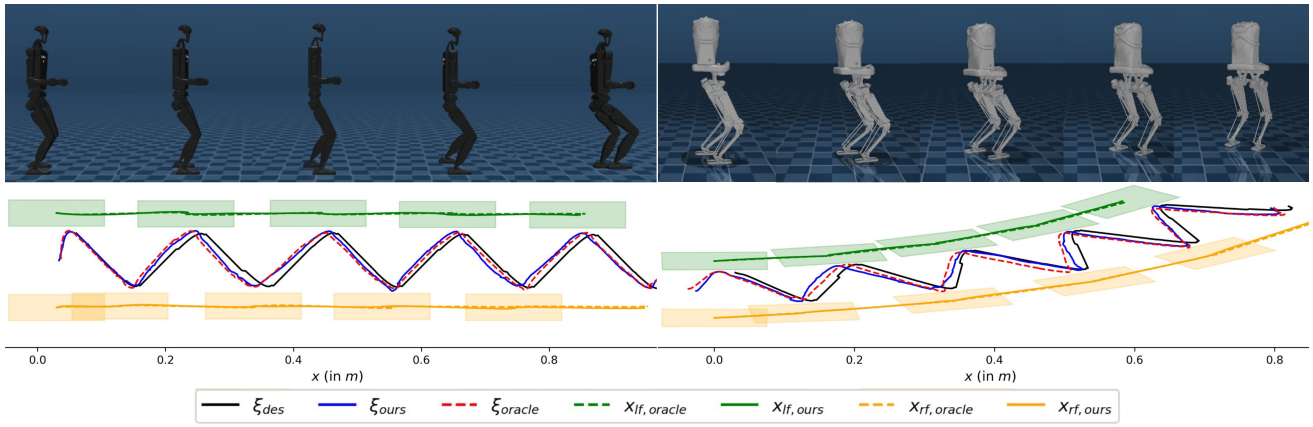

Fig. 4: **Quantitative evaluation on DCM and foot tracking.** The H1-2 robot walks straight forward at a speed of $0.2m/s$, while the Kangaroo is commanded with a linear velocity of $0.2m/s$ and an angular velocity of $0.2rad/s$. ξ_{des} shows the planned DCM trajectory and the footprints are visualized as polygons in green and orange.

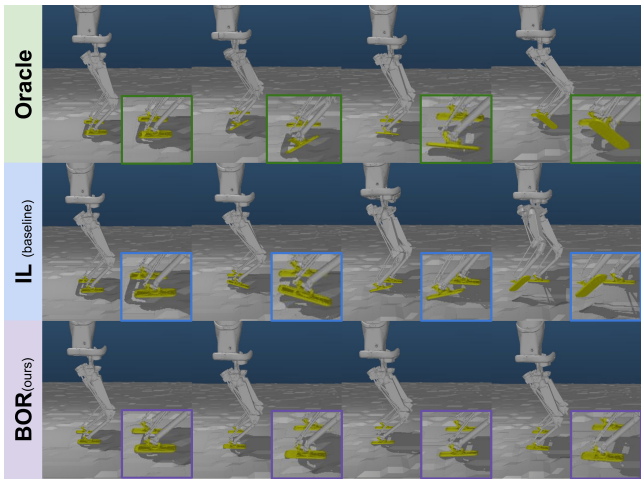

Fig. 5: **Evaluation on uneven terrain.** Uneven terrain introduces additional uncertainties in real-world environments, yet even the Oracle policy struggles to explicitly encode ground irregularities. Thanks to the learning paradigms in our framework, our method learns adaptive behaviors that go beyond simply mimicking the Oracle, unlike the baseline IL approach.

their approach—which leverages thousands of parallel environments—we train using only a single environment. Despite these constraints, our method successfully learns the desired performance by combining supervision from the Oracle and reward signals during training.

3) Robustness on uneven terrains: In addition to uncertainties in the robotic system, ground unevenness introduces another major source of uncertainty in locomotion tasks. Because it is challenging—or even infeasible—to generate optimal planning trajectories for variables such as foot placement and ankle touchdown angles on uneven terrain, these irregularities pose significant challenges for the Oracle in guiding the learning process. In this work, we always assume a flat floor for the Oracle policy while leveraging the learning paradigm to handle uncertainties. Figure 5 presents simulated results using identical walking parameters under different approaches. The Oracle policy leads to unstable control due to unmodeled ground unevenness, and this instability is directly inherited by the imitation learning (IL) baseline, since IL strictly mimics the Oracle's behavior. In contrast, our method (BOR) enables adaptive behavior by allowing the policy to explore its action space through reinforcement learning. The simulation results demonstrate that our method can further optimize the policy beyond merely imitating the Oracle.

4) Command tracking: Leveraging the trajectory generators in our model-based pipeline, we obtain DCM and foot trajectories that enable the robots to follow commanded velocities effectively. To evaluate the velocity tracking performance, we plot the planned and measured trajectories for the H1-2 and Kangaroo robots in Figure 4. In these experiments, the H1-2 robot is commanded to walk forward at $0.2m/s$, while the Kangaroo receives an additional $0.2rad/s$ angular velocity. Our method tracks the planned trajectories as smoothly and accurately as the Oracle policy, demonstrating the ability to closely follow the commanded velocities under domain randomization.

5) Ablation study: We conduct an ablation study to evaluate the role of residual learning in our framework. For that, we combine Oracle supervision with standard RL, in which our RL policy directly learns to command torques rather than residuals. Therefore, the Base policy is not used for training or for inference. As shown in Table III and Figure 2, learning residual torques improves learning efficiency, but the overall performance gains are limited.

To further investigate this behavior, we analyze torque tracking for the Kangaroo robot's right hip joint, as illustrated in Figure 3. Due to the extensive domain randomization, the Base policy produces torque commands that deviate significantly from the desired targets, resulting in poor performance. Consequently, learning residuals on top of the Base policy offers limited benefits compared to directly learning the torque commands themselves.

Despite the large discrepancy between the Base and Oracle policies, our RL agents successfully learn to compensate for these differences, as evidenced by the improved tracking shown in Figure 3.

IV. Discussion

Learning torque-level locomotion policies with reinforcement learning is typically data-inefficient, while model-based optimization methods rely on accurate system models that rarely hold in real-world settings. Our approach bridges these two paradigms by learning a residual torque policy guided by an Oracle while explicitly modeling real-world uncertainties in simulation.

To balance imitation and exploration, we jointly optimize reinforcement and supervision objectives. Oracle guidance significantly improves learning efficiency; however, even an Oracle with access to ground-truth states cannot anticipate all adverse conditions. For instance, abrupt user command changes or unstable ground contacts can cause failures that the Oracle cannot prevent. As shown in our results on the Bruce and H1 robots, over-reliance on the Oracle can therefore limit performance under highly stochastic conditions.

V. Conclusion

In this work, we present a supervised reinforcement learning framework for torque-based humanoid locomotion that addresses two fundamental challenges: the sim-to-real gap inherent in model-based control and the data inefficiency of standard model-free RL. To improve robustness to unknown system uncertainties, we employ domain randomization during training, allowing the learned policies to generalize across a wide range of simulated conditions. To further enhance learning efficiency, we introduce a model-based Oracle policy that provides informative supervision despite being suboptimal under randomized dynamics.

Oracle supervision is incorporated into the RL training process through an additional loss term that explicitly encourages the policy to align with the Oracle during gradient updates. This direct guidance significantly accelerates learning and leads to improved performance compared to standard RL methods that rely solely on reward signals.

We evaluate our framework on three bipedal robots—Kangaroo, Unitree H1-2, and Bruce—and show that the learned policies achieve performance comparable to their respective Oracles across all platforms. Notably, the same training framework is applied without robot-specific modifications. Ablation studies further demonstrate that our method effectively supports both residual torque learning on top of a base controller and direct torque command learning.

Overall, our results demonstrate that Oracle-guided supervised reinforcement learning provides an efficient, robust, and transferable solution for torque-based locomotion control in humanoid robots.

References

[1] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D linear inverted pendulum mode: A simple modeling for a biped walking pattern generation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2001.

[2] T. Sugihara, Y. Nakamura, and H. Inoue, "Real-time humanoid motion generation through ZMP manipulation based on inverted pendulum control," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2002.

[3] P.-b. Wieber, "Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations," in *2006 6th IEEE-RAS International Conference on Humanoid Robots*.

[4] T. Takenaka, T. Matsumoto, and T. Yoshiike, "Real time motion generation and control for biped robot -1st report: Walking gait pattern generation-," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009.

[5] J. Englsberger, C. Ott, and A. Albu-Schäffer, "Three-Dimensional Bipedal Walking Control Based on Divergent Component of Motion," *IEEE Transactions on Robotics*, 2015.

[6] G. Mesesan, J. Englsberger, G. Garofalo, C. Ott, and A. Albu-Schäffer, "Dynamic Walking on Compliant and Uneven Terrain using DCM and Passivity-based Whole-body Control," in *Proc. 19th Humanoids*.

[7] K. Hu, C. Ott, and D. Lee, "Learning and generalization of compensative zero-moment point trajectory for biped walking," *IEEE Transactions on Robotics*, 2016.

[8] 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*, 2022.

[9] Z. Xie, P. Gergondet, F. Kanehiro *et al.*, "Learning bipedal walking for humanoids with current feedback," *IEEE Access*, 2023.

[10] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control," *IJRR*, 2025.

[11] F. Yu, R. Batke, J. Dao, J. Hurst, K. Green, and A. Fern, "Dynamic bipedal turning through sim-to-real reinforcement learning," in *IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*.

[12] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," in *Conference on Robot Learning*. PMLR, 2020, pp. 317–329.

[13] T. He, Z. Luo, W. Xiao, C. Zhang, K. Kitani, C. Liu, and G. Shi, "Learning human-to-humanoid real-time whole-body teleoperation," in *IROS*, 2024.

[14] T. He, Z. Luo, X. He, W. Xiao, C. Zhang, W. Zhang, K. M. Kitani, C. Liu, and G. Shi, "Omnih2o: Universal and dexterous human-to-humanoid whole-body teleoperation and learning," in *8th Annual Conference on Robot Learning*, 2024.

[15] H. Duan, J. Dao, K. Green, T. Apgar, A. Fern, and J. Hurst, "Learning task space actions for bipedal locomotion," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[16] G. A. Castillo, B. Weng, S. Yang, W. Zhang, and A. Hereid, "Template model inspired task space learning for robust bipedal locomotion," in *IROS*, 2023.

[17] T. Egle, Y. Yan, D. Lee, and C. Ott, "Enhancing model-based step adaptation for push recovery through reinforcement learning of step timing and region," in *IEEE-RAS 23rd Humanoids*.

[18] G. Mesesan, R. Schuller, J. Englsberger, C. Ott, and A. Albu-Schäffer, "Unified motion planner for walking, running, and jumping using the three-dimensional divergent component of motion," *IEEE Transactions on Robotics*, 2023.

[19] G. Mesesan, J. Englsberger, C. Ott, and A. Albu-Schäffer, "Convex Properties of Center-of-Mass Trajectories for Locomotion Based on Divergent Component of Motion," *IEEE Robotics and Automation Letters*, 2018.

[20] T. Egle, J. Englsberger, and C. Ott, "Analytical center of mass trajectory generation for humanoid walking and running with continuous gait transitions," in *IEEE-RAS 21st Humanoids*, 2022.

[21] J. Englsberger, G. Mesesan, A. Werner, and C. Ott, "Torque-Based Dynamic Walking - A Long Way from Simulation to Experiment," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms." *CoRR*, 2017.

[23] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *ICLR2016*, 11 2015.

[24] A. Roig, S. K. Kothakota, N. Miguel, P. Fernbach, E. M. Hoffman, and L. Marchionni, "On the Hardware Design and Control Architecture of the Humanoid Robot Kangaroo," in *6th Workshop on Legged Robots during the Int. Conf. Robot. Automat.*, 2022.

[25] Y. Liu, J. Shen, J. Zhang, X. Zhang, T. Zhu, and D. Hong, "Design and Control of a Miniature Bipedal Robot with Proprioceptive Actuation for Dynamic Behaviors," in *2022 ICRA*.

[26] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[27] R. Schwan, Y. Jiang, D. Kuhn, and C. N. Jones, "PIQP: A proximal interior-point quadratic programming solver," in *CDC*, 2023.

[28] D. Kim, G. Berseth, M. Schwartz, and J. Park, "Torque-based deep reinforcement learning for task-and-robot agnostic learning on bipedal robots using sim-to-real transfer," *RA-L*, 2023.

[29] S. Chen, B. Zhang, M. W. Mueller, A. Rai, and K. Sreenath, "Learning torque control for quadrupedal locomotion," in *Humanoids*, 2023.