Robotics: Science and Systems 2024
Delft, Netherlands, July 15-July 19, 2024

RL2AC: Reinforcement Learning-based Rapid Online Adaptive Control for Legged Robot Robust Locomotion

Shangke Lyu, Xin Lang, Han Zhao, Hongyin Zhang, Pengxiang Ding, Donglin Wang*

***Abstract*—Dynamic fast adaptation is one of the basic capabilities that enables the animals to timely and properly adjust its locomotion reacting to the unpredictable changes. Such capability is also essential for the quadruped robot, when working in the unforseen environment. While reinforcement learning (RL) has achieved a significant progress in locomotion control, rapid adaptation to the model uncertainties remains a challenge. In this paper, we seek to ascertain the control mechanism behind the locomotion RL policy, from which we propose a new RL-based Rapid onLine Adaptive Control (RL2AC) algorithm to complementarily combine the RL policy and the adaptive control together. RL2AC is run at a frequency of 1000Hz without the need for simultaneous training with RL. It presents a strong capability against the external disturbances or the sim-to-real gap, resulting in a robust locomotion, which is achieved through proper torque compensation derived from a novel adaptive controller. Various simulation and experiments have demonstrated the effectiveness of the proposed RL2AC against the heavy load, disturbances acted on one leg, lateral torque, sim-to-real gap and various terrains.**

I. Introduction

With the advancement of computing resources, model-free reinforcement learning (RL) has demonstrated significant potential in the motion control of legged robots, successfully accomplishing a range of highly dynamic and complex tasks [1]–[7]. In contrast to model-based control methods [8]–[17], RL liberates humans from the laborious and time-consuming task of modeling, significantly easing the challenges of dynamic motion control, particularly for non-expert users. Despite the impressive performance, it naturally arises the questions about what is the exact control mechanism behind the legged robot RL policy and what distinguishes it from traditional model-based control. In the classical model-based control framework for legged robots [16], the calculation of joint torque commands integrates feedback and feedforward mechanisms. The feedback torque is derived from PD feedbacks on both joint position and velocity tracking errors, with the joint reference signals calculated from Whole Body Control (WBC). The feedforward torque is generated based on the robot dynamics and ground reaction force (GRF) via Jacobian transpose, with GRF optimized by Model Predictive Control (MPC) and further rectified by WBC considering full-body dynamics. Such control architecture aims to generate the appropriate GRF for stance foot to maintain balance while using a PD feedback loop to precisely track the reference trajectory of swing foot and achieve various dynamic gaits.

Inspired by this, one may wonder if RL follows a similar approach when implemented on legged robots. Literature [18] observes that the closed-loop dynamics of a complex bipedal robot, Cassie, controlled by a learned RL policy together with a joint-level PD controller, can be locally captured by a decoupled linear model. This suggests that if the RL policy is well-trained, the torque generated by the PD controller based on action commands can well approximate the system dynamics and decouple the system into a linear model. This essentially resembles the traditional model-based control mentioned earlier. Hence, delving into a comprehensive exploration and understanding of the correlation and distinctions between reinforcement control and model-based optimization control is deemed worthwhile.

Additionally, the RL policy typically undergoes training in a simulator before being transferred to the physical world, while it is always struggling with the sim-to-real issue, stemming from differences in dynamic models and environments between the simulator and reality, which has attracted a lot of attentions [19]–[21]. Furthermore, in scenarios where the quadruped robot operates in unstructured environments, the terrains can vary significantly, and robot may face various external disturbances, exacerbating the sim-to-real issue and posing substantial challenges to the robustness of locomotion control.

In this paper, we propose a **RL**-based **R**apid on**L**ine **A**daptive **C**ontrol (RL2AC) for legged robot robust locomotion as indicated in Fig. 1. This approach involves deriving an adaptive torque compensator, which is combined with an RL policy to mitigate the effects of external disturbances or model mismatches due to the sim-to-real gap, thereby enhancing the robustness of the quadruped robot's locomotion. We observe that locomotion policies often exhibit a decoupling into feed-forward and feedback components, akin to model-based control. The trained feed-forward term counteracts the influence of GRF during policy learning and contributes to balance maintenance, while the feedback part is then responsible for gait tracking. However, the existences of the sim-to-real gap or external disturbances introduce the dynamics shift problem, breaking the balance between the policy and the GRF, that learned from the simulator. On the other hand, we also observe that such dynamics shift can be captured and inferred from the motion differences between the simulation and the reality, and further be identified and compensated in an adaptive manner.

All the authors are with Machine Intelligence Lab (MiLAB), School of Engineering, Westlake University, Hangzhou, 310024, China. (*Corresponding author: wangdonglin@westlake.edu.cn)

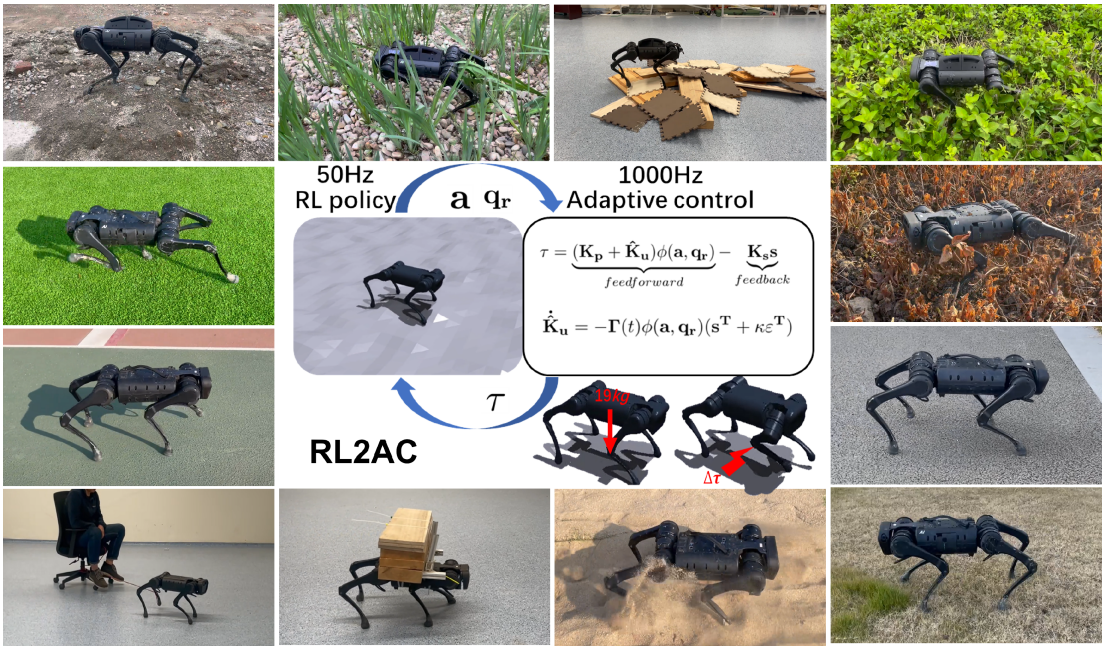


Fig. 1. RL2AC: We propose a new paradigm for legged robot to complementarily combine RL policy and adaptive control for robust locomotion, where a novel adaptive control is developed to generate a feed-forward compensation to eliminate the effects of the external disturbances based on policy actions. It is run on $1000Hz$ without requiring training together with RL and can be directly implemented on the legged robot, enabling rapid online adaptation against the variations or disturbances in dynamics. RL2AC is verified to be robust against the heavy load, disturbances acted on one leg, lateral torque, sim-to-real gap and various terrains.

With the aid of the dynamics compensator, the quadruped robot is capable of performing the locomotion demonstrated in the simulator, even in the presence of the external disturbances or sim-to-real gap and holds promise for achieving robust locomotion.

In summary, the contributions of this work can be unfolded as: 1) the mechanism behind the RL-based locomotion control is explored and some observations behind are presented, which is beneficial for the interpretability and transparency of RL policies and may advance the development of robust locomotion control eventually; 2) a robust locomotion control strategy RL2AC is proposed, which enables the robot to rapidly react and adapt to the variations and enhances its performance against the model uncertainties; 3) the proposed framework finds a new paradigm to closely and complementarily integrate the RL policy and the adaptive control together. Simulations and experiments demonstrate the effectiveness of the proposed method in various terrains and tasks.

II. Related Work

In the initial stages of controlling the quadruped robot system, a preferred approach is the utilization of model-based methods, as indicated by various studies [8], [10], [14]. This involves calculating a torque feed-forward term for the stance leg to maintain balance and commonly employing a PD controller for the swing leg to track the reference gait. The adaptability to diverse GRF is a key element in enhancing quadruped locomotion performance, particularly over rough terrains. In this context, inverse dynamics control emerges as a convenient and effective method for compensating GRF in dynamic locomotion [9], [11], with its performance upon the accuracy of GRF estimation. In challenging terrains like uneven roads or rocky surfaces, the significant variations in GRF between each step pose difficulties in dynamic locomotion control. The inherent strong coupling in quadruped robot kinematics and dynamics, along with additional stability constraints, further complicates highly dynamic maneuvers. Resolving the conflicts between position control and interaction force control of the legged robot, optimization approaches are introduced. These include hierarchical task-space inverse dynamics control [12], WBC [15], and a combination of MPC and WBC [16]. Additionally, studies have demonstrated that appropriately modifying or redistributing interaction forces among the legs leads to coordinated stance and swing leg movements, resulting in robust behaviors such as recovery from pushes [13] or walking on slippery terrains [17].

While model-based control and optimization methods have demonstrated robust performance in specific tasks, they face challenges when applied to unknown terrains. The difficulty arises from the inability to adequately model and encode environmental information into the controller design. In such scenarios, model-free RL emerges as a promising alternative, leveraging its learning and adaptation capabilities, especially for agile maneuvers [3], coordination with perception during parkour [4] [5], autonomy in the unknown environment exploration [22]. Typically, diverse motion behaviours require extensive reward design. To release human from the time-consuming and labor-intensity reward engineering, adversarial motion priors (AMP) [23] is introduced, which leverages a generative adversarial network (GAN) [24] manner to learn a style reward from a reference motion dataset and enables the robot to mimic natural movements of the real dog [25].

Teacher-student paradigm [1] is another widely adopted policy learning framework for the quadruped robot locomotion, in which the privileged information such as contact force, terrain profile, is fed into the teacher network to promote the policy improvement. The obtained knowledge is then distilled to a student network for deployment. This enables the robot to acquire more effective information for action generation beyond the proprioceptive measurements, ensuring better alignments with the environment variations [2] [26]. This environment latent representation is also implicitly learned through variational auto-encoder (VAE) [6] or contractive learning [27] for efficient legged locomotion.

Stable locomotion seeks for the reliable leg movements and guaranteed stability. RL-based control policy do present stunning performance in many complex tasks, while the lack of interpretability and transparency significantly limit its potential in commercialization. The combination of the model-based method and RL policy in a complementary manner may take both strengths of them and hold a promise to achieve the performance improvement. They can be settled heretically, where the RL policy is responsible for diverse reference command generation in high-level, while the model based control is used for precise commands tracking in low-level [28] [29]. This can also be carried out in a converse way, in which model-based trajectory optimization (TO) method solve a reference motion prior from some stability constraints for the policy learning so as to regulate the RL action generation towards the stable and trustable movements [30] [7]. In heretical architecture, it prefers to decouple and split some functionalities from RL framework and then to be replaced by the model-based approaches. To enhance the robustness against the external disturbances, the quadruped robot kinematics [31] or dynamics [32] is adopted to correct the policy action, or conversely RL policies is used for corrective whole-body motion tracking and recovery control [33], all of which aim for stable and robust dynamic locomotion.

Pursuing the accurate tracking in the presence of the model uncertainties, adaptive control is one of the effective methodologies in both control area and robotics area [34]. It typically encompasses two distinct schemes: an indirect scheme, where plant parameters are estimated online for calculating controller parameters, and a direct scheme, where the plant model is parameterized in terms of controller parameters that are directly estimated without plant parameter estimation [35]–[37]. Composite adaptive control, on the other hand, represents an integrated approach that combines both direct and indirect adaptive control strategies. This method involves providing feedback from both tracking errors and prediction errors to update parameter estimates [38] [39]. In the context of quadruped robots, precise modeling of contact forces proves challenging, and the resulting model uncertainties can substantially impact locomotion performance. To address external disturbances or model uncertainties, various adaptive control approaches have been developed to enhance robust locomotion [40], [41]. These approaches suggest that the model uncertainties of the quadruped robot can be linearly parameterized. Moreover, they underscore the significance of online adaptation mechanisms in dynamic locomotion control.

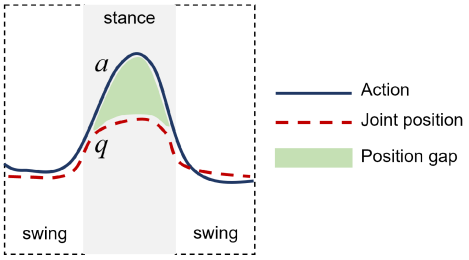


Fig. 2. An illustration of action and joint position of quadruped robot under policy $\pi_{\mathbf{d}}$: PD joint position tracking error $(\mathbf{a}-\mathbf{q})$ presents significant differences between stance phase and swing phase. In swing phase, the joint position $\mathbf{q}$ can almost well track the action $\mathbf{a}$, while there exists an apparent position gap between them in stance phase.

III. Preliminaries

A. RL part

In this paper, similar to [6], the environment is modeled as an infinite-horizon partially observable Markov decision process (POMDP), defined by the tuple $M = (S, \mathcal{O}, \mathcal{A}, d_0, p, r, \gamma)$. The full state, partial observation, and action are continuous, and defined by $s \in \mathcal{S}, \mathbf{o} \in \mathcal{O}$, and $\mathbf{a} \in \mathcal{A}$, respectively. The environment starts with an initial state distribution, $d_0(\mathbf{s}_0)$; progresses with a state transition probability $p(s_{t+1}|\mathrm{s}_t, \mathrm{a}_t)$; and each transition is rewarded with a reward function, $r : S \times \mathcal{A} \to \mathcal{R}$. The discount factor is defined by $\gamma \in [0, 1)$. A temporal observation at time t over the past H measurements is defined as $\mathbf{o}_t^H = \begin{bmatrix}\mathbf{o}_t & \mathbf{o}_{t-1} \dots \mathbf{o}_{t-H}\end{bmatrix}^T$. A context vector, $\mathbf{z}_t$, is defined, which contains a latent representation of the world state.

B. Dynamics

Let q is the actual joint position. The joint level dynamic model of the quadruped robot can be commonly expressed as follows:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) - \mathbf{J}^{\mathbf{T}}\mathbf{F} = \tau, \tag{1}$$

where $\mathbf{M}(\mathbf{q}), \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}), \mathbf{G}(\mathbf{q}), \tau, \mathbf{F}, \mathbf{J}$ denote the inertial matrix, the Coriolis force and gravity force, the joint torque input, the contact force and the contact Jacobian matrix, respectively. For the quadruped robot RL framework, PD controller is commonly adopted as follows:

$$\tau = \mathbf{K_p}(\mathbf{a} - \mathbf{q}) - \mathbf{K_d}\dot{\mathbf{q}}, \tag{2}$$

where $\mathbf{K_p}$ and $\mathbf{K_d}$ represent the gain matrices and $\mathbf{a}$ is sampled from the well-trained policy denoted as $\pi_{\mathbf{d}}$, which is obtained from RL directly. Substituting Eq. (2) in to Eq. (1) yields

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) - \mathbf{J}^{\mathbf{T}}\mathbf{F} = \mathbf{K_p}(\mathbf{a} - \mathbf{q}) - \mathbf{K_d}\dot{\mathbf{q}}. \tag{3}$$

IV. Control Mechanism Analysis Behind Locomotion Policy

A. RL policy for locomotion

Generally, in RL framework, it orchestrates various elements, such as body commands, gait information, GRF, and

tracking errors, together and consolidates them into a simplified action $\mathbf{a}$. To figure out how the RL algorithm regulates these elements together, we conduct a test by following the RL framework in [6] to control the quadruped robot, which will be detailed later. The policy $\pi_{\mathbf{d}}$ in [6] is a neural network that infers 12 joints position commands to control the quadruped robot, given a proprioceptive observation, body velocity, and some latent information. The training process is conducted on the Isaac Gym, and a PD control is implemented at the low level, as indicated in Eq. (3). Domain randomization is adopted during the training so as to enhance the robustness of the policy against diverse testing conditions. The learned policy $\pi_{\mathbf{d}}$ is deployed to execute locomotion tasks based on given commands in rough terrains. In contrast to MPC-WBC frameworks, where the low-level PD controller effectively tracks given joint commands with small joint position tracking errors, the policy $\pi_{\mathbf{d}}$ manifests a different phenomenon. Specifically, despite sending action $\mathbf{a}$ to the PD controller as a reference command, the actual joint position does not always precisely track the action, as illustrated in Fig. 2. Instead, it exhibits obviously distinctions between the swing and stance phases, though it is not carefully and intentionally designed. In swing phase, the joint position can track the action, while there exists a gap between the joint position and the action command in stance phase. Such phenomenon exists in many RL policies [1], [2], [6], [25]–[27] for quadruped robot. The main discrepancy of these two stages is the existence of the contact force, which promotes an exploration of the intrinsic relationship between the presence of contact force and joint position tracking errors.

Let $\tau_{\mathbf{e}} = \mathbf{K_p}(\mathbf{a} - \mathbf{q})$ and $\tau_{\mathbf{m}} = \mathbf{J^T F}$. In the test, the quadruped robot was asked to locomote with different commands in a rough terrain and $\tau_{\mathbf{e}}$ and $\tau_{\mathbf{m}}$ were collected accordingly. Inspired by the model-based method, in which a feed-forward term is designed to regulate the contact force exerted on the stance leg, we compare the normalized values of $\tau_{\mathbf{e}}$ and $\tau_{\mathbf{m}}$, as indicated in Fig. 3. Let $norm(\cdot)$ be the normalization process. It is interesting to note that the normalized torque $norm(\tau_{\mathbf{e}})$ obtained by the specific gain $\mathbf{K_p}$ and the corresponding tracking error $(\mathbf{a} - \mathbf{q})$, can well approximate the measured normalized contact torque $norm(\tau_{\mathbf{m}})$. This presents a phenomenon similar to feedback linearization as observed in [18]. The RL policy $\pi_{\mathbf{d}}$ regulates the feed-forward torques by adjusting the joint tracking errors of the stance legs to generate the suitable GRF for locomotion, while simultaneously generates the reference joint position commands to drive the movements of the swing legs to form different gaits according to the given commands.

More significantly, when varying the given commands, the torque $norm(\tau_{\mathbf{e}})$ is still able to well estimate the structure of the actual contact force $norm(\tau_{\mathbf{m}})$. The intuition is that, owing to domain randomization, the policy $\pi_{\mathbf{d}}$ has effectively grasped the features of diverse GRF changes on various terrains in the simulation. Consequently, it can adeptly modify actions in response to different terrain frictions and the required contact forces/torques, thereby maintaining the robot's balance. This, on the other hand, signifies that the tracking error term $(\mathbf{a} - \mathbf{q})$ well captures the structure of the GRF dynamics changes and

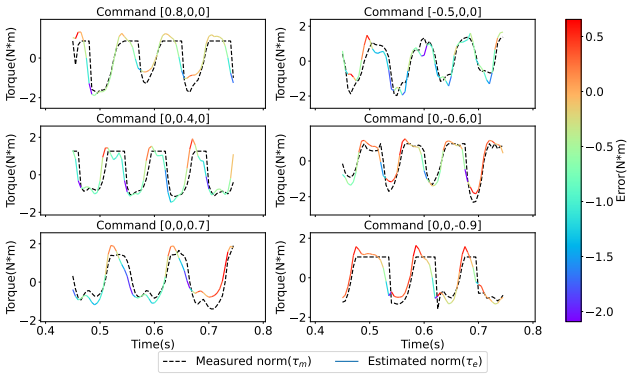


Fig. 3. Illustration of $\tau_{\mathbf{e}}$ and $\tau_{\mathbf{m}}$ under different commands on rough plane: under policy $\pi_{\mathbf{d}}$, the normalized estimated torque $norm(\tau_{\mathbf{e}})$ based on PD position joint tracking error $(\mathbf{a} - \mathbf{q})$ can well approximate the measured normalized contact torque $norm(\tau_{\mathbf{m}})$ across a wide range of locomotion commands. This demonstrates the effectiveness of the control policy $\pi_{\mathbf{d}}$ in approximating GRF for various movements.

can be regarded as certain embodied representation of GRF. More specifically, it is suggested that GRF can be expressed as a linear parameterization involving the gain $\mathbf{K_p}$ and the tracking error $(\mathbf{a} - \mathbf{q})$. This bears resemblance to principles observed in adaptive control [38], where the dynamics of the rigid connected robot can be linearly parameterized by a regressor matrix and a constant parameter vector.

B. Dynamics shift

When the policy $\pi_{\mathbf{d}}$ is deployed in unseen tasks/environments or transferred from the simulation to the reality, it always leads to a performance degradation. This is commonly interpreted as an out-of-distribution (OOD) problem that the policy encounters states or actions during testing that differ significantly from the data it was trained on. In other words, the policy faces difficulties when confronted with samples from a distribution that was not adequately represented in its training set. When referring to RL framework, this is also called dynamics shift problem that the state-action transition model is shifted away from the original one.

In the perspective of the quadruped robot dynamic model, the dynamics shift is equivalent to introduce the additional uncertainties as follows:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) - \mathbf{J^T F} - \Delta\tau = \tau, \qquad (4)$$

where $\Delta\tau$ denote the unknown torque uncertainties that may come from the model mismatch or the external contact force changes. Typically, the dynamic model of a quadruped robot can be mostly simulated and thus the majority of the torque uncertainties come from the variations in the contact forces, that is, $\Delta\tau \approx \mathbf{J^T}\Delta\mathbf{F}$. In scenarios without additional torque uncertainties, the learned policy $\pi_{\mathbf{d}}$ identifies nearly optimal actions solution, $\mathbf{a}$, according to the given commands and the model outlined in Eq. (3). Consequently, the quadruped robot can well perform the given tasks. When $\mathbf{J^T}\Delta\mathbf{F}$ appears, the optimality of $\pi_{\mathbf{d}}$ may not hold, resulting the performance deterioration. As mentioned before, domain randomization is

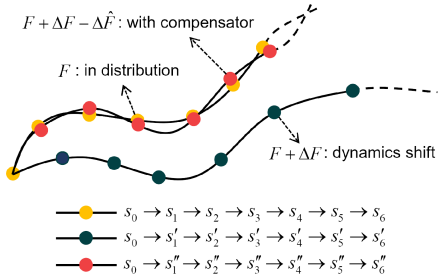


Fig. 4. An illustration of dynamics compensator: the existence of additional model uncertainties ΔF results in the deviation of actual trajectory away from optimal trajectory learned in simulation and an appropriate dynamics compensator may alleviate such issue by narrowing down the state-action distribution gap between simulation and reality.

one effective approach to alleviate the sim-to-real issue by simulating a broader range of scenarios during the learning phase. Here, we consider the case that the external torque uncertainties are unseen in the training stage and significantly different from the range specified for domain randomization.

Leveraging the optimality of the policy $\pi_{\mathbf{d}}$ observed in simulation, if the quadruped robot can replicate its performance in the face of model uncertainties, it holds the promise of bolstering the robot's robustness against dynamics shift. However, the substantial variations in the dynamic model, transitioning from Eq. (3) to Eq. (4), introduce challenges in maintaining the optimality of the policy, leading to performance deviations, as depicted in Fig. 4. A viable strategy to address this challenge involves compensating the extra uncertainties. Due to the optimally of the policy $\pi_{\mathbf{d}}$ with respect to Eq. (3), if additional dynamics $\mathbf{J^T}\Delta\mathbf{F}$ can be approximated and canceled out in a feed-forward manner, the effectiveness of the obtained policy $\pi_{\mathbf{d}}$ therefore remains intact. This assurance ensures the performance of the quadruped robot even in the unseen environment or in the presence of model uncertainties, as illustrated in Fig. 4. However, if the dynamics compensation is not properly designed, it may introduce additional model uncertainties, exacerbating the dynamics shift problem instead. This restricts the direct implementation of some traditional force/torque estimation methods [42]–[44], since it may break the mapping between the policy and the GRF, learned from training.

V. RL2AC: RL-BASED RAPID ONLINE ADAPTIVE CONTROL

During the real-world deployment, there is another commonly adopted trick to relieve the sim-to-real discrepancy by adjusting the PD gains. Essentially, the adjustments of the PD gains change the corresponding feed-forward compensations and further alter the GRF. By choosing a set of PD gains that align more effectively with the specific environment and tasks, the resultant actions generated by the policy $\pi_{\mathbf{d}}$ lead to the performance improvement, thereby narrowing the gap between simulation and reality. This provides a new and convenient perspective to enhance the robustness of the policy $\pi_{\mathbf{d}}$ against the dynamics shift. If the PD gains can be autonomously and appropriately adjusted in a real-time manner, it is promised to obtain a better performance.

The difficulties to autonomously adjust the PD gains lay in the strong coupling between the PD gains and the action $\mathbf{a}$. The modification of the PD gains will cause the corresponding changes in the action. Such implicit and adhesive relationship is hard to precisely captured and hence it is difficult to derive a appropriate criterion to regulate the PD gains. To avoid the explicit representation, some works [21] [26] preferred to use the neural network to learn this mapping directly. An actuator network was trained using the pre-collected data to identify the non-ideal relationship between PD error and realized torque, which is verified to be able to reduce the sim-to-real gap. However, such data collection is time-consuming and labor-intensity and the obtained network is only effective for the specific robot. Besides, it brings some additional issues, such as fragile stability, and limited generation to the unseen situations.

Another approach is to decouple this mapping. According to the analysis in Section IV-A, the action sent to the PD control undertakes two main roles. One is to generate an appropriate GRF for the balance maintenance and locomotion, while the other is to drive the swing of the legs with suitable gaits upon the terrains. Hence, the model defined in Eq. (3) can be rewritten as:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) - \mathbf{J^T}\mathbf{F} = \underbrace{\mathbf{K_p}(\mathbf{a} - \mathbf{q_r})}_{term\ m} + \underbrace{\mathbf{K_p}(\mathbf{q_r} - \mathbf{q}) - \mathbf{K_d}\dot{\mathbf{q}}}_{term\ n}, \quad (5)$$

where $\mathbf{q_r}$ denotes a reference joint position. As seen in Eq. (5), the PD controller is divided into two parts, *term m* and *term n*. Suppose $\mathbf{q_r}$ is a meticulously planned reference joint position, dictating suitable gaits for the provided commands and serving as the equilibrium point for the system in Eq. (5). Hence, it is expected that *term m* can well compensate the dynamics on the left side of Eq. (5) so that *term n* can converged to the equilibrium point at $\mathbf{q_r}$. In simulations with well-trained policy $\pi_{\mathbf{d}}$, the quadruped robot demonstrates precise execution of given commands. Consequently, the obtained actual joint position denoted as $\mathbf{q}'$ can be regarded as the equilibrium point of Eq. (3). Since the quadruped robot is expected to perform similarly as observed in the simulation regardless of the external forces, the actual joint position $\mathbf{q}$ is expected to be as close as possible to the joint position $\mathbf{q}'$ demonstrated in the simulation. Alternatively speaking, it is desired to modify the equilibrium point of Eq. (5) close to $\mathbf{q}'$. To achieve this, $\mathbf{q_r}$ can be specified as $\mathbf{q_r} = \mathbf{q}'$ so that the quadruped robot can act as observed in simulations. Therefore, in the presence of the dynamics shift, the effective design of *term m* to counteract additional torque changes holds the potential for the robot to perform comparably to scenarios without dynamics shifts.

A. RL foundation

To alleviate the dynamics shift issue, we propose a RL2AC algorithm that is composed of the RL part and the adaptive control part, as illustrated in Fig. 5.

Specifically, for RL part, it mainly follows the work in [6]. The action is specified as quadruped robot 12 join-

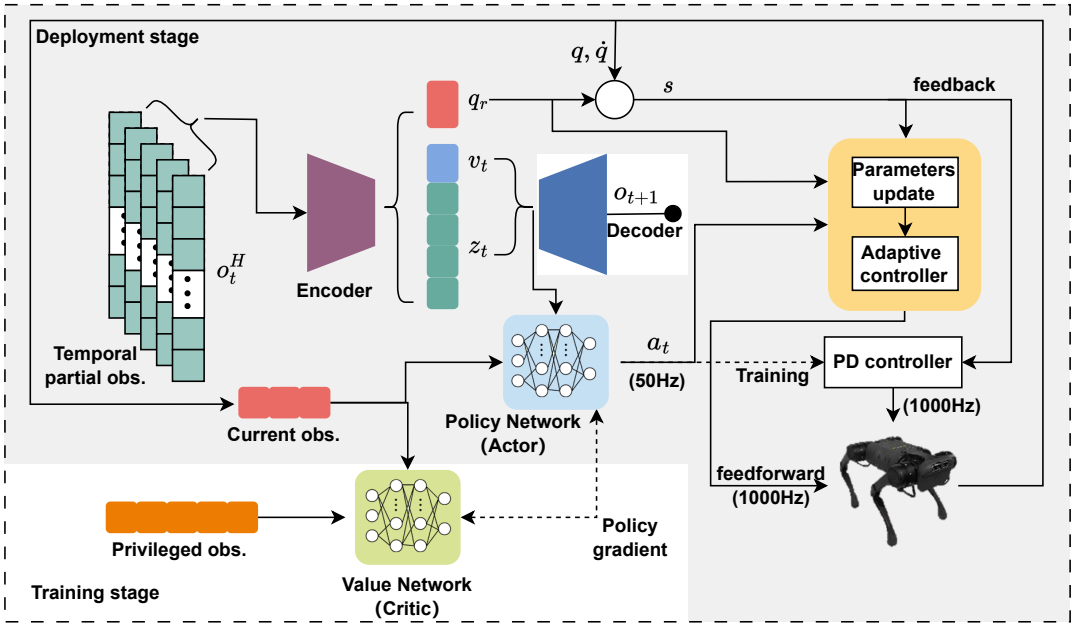


Fig. 5. An illustration of the proposed RL2AC framework: the white background section denotes the modules used only for the training stage, while the grey background section denotes the modules used in deployment stage. A variational auto-encoder (VAE) structure is employed to estimate and infer a latent representation $\mathbf{z}_t$ of the environment based on the historical observations $\mathbf{o}_t^H$. The decoder is only used in training stage and abandoned in the deployment. The framework incorporates a policy network (actor) in blue and a value network (critic) in green: the former processes current observations and latent embedding to output actions, while the latter integrates current and privileged observations to produce state values, aiding the policy refinement during training. During the deployment stage, the action $\mathbf{a}_t$ is sent to the adaptive controller part together with a reference joint position $\mathbf{q_r}$ and a sliding vector $\mathbf{s}$ to generate the control input for the quadruped robot. Note that the controller does not need to train together with the policy and is plug and play directly during deployment.

t position commands. For the policy network, denoted as $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{v}_t, \mathbf{z}_t)$, it receives a proprioceptive observation $\mathbf{o}_t$, body velocity $\mathbf{v}_t$, and latent state $\mathbf{z}_t$ and infers an action output $\mathbf{a}_t$. The observation that feeds into the policy network contains the body angular velocity ω_t, the projected gravity vector $\mathbf{g}_t$, base velocity commands $\mathbf{c}_t$ (including linear velocities and yaw velocity), joint positions $\mathbf{q}_t$, joint velocities $\dot{\mathbf{q}}_t$ and previous actions $\mathbf{a}_{t-1}$. For the value network, except the observation $\mathbf{o}_t$ and body velocity $\mathbf{v}_t$, it receives the additional privileged observation, including $\mathbf{d}_t$, the disturbance force applied randomly on the robot's body and $\mathbf{h}_t$, the height map scan of the robot's surroundings. The reward function encourages the quadruped robot to follow the desired base velocity commands c and generate smooth and efficient motions. We define the reward function as the sum of the terms in table I, where $(\cdot)^{cmd}$ and $(\cdot)^{des}$ denote the commanded and desired values, respectively and $\mathbf{q}, \mathbf{v}_{xy}, \omega_z, \mathbf{h}, p_{f\ z\ k}$ and $v_{f,xy,k}$ denote the gravity vector projected into the robots body frame, linear velocities in the xy plane, yaw rate, body height w.r.t. the ground, foot height, foot lateral velocity, and joint torque, respectively. The policy is optimized using the proximal policy optimization (PPO) algorithm [45].

In [6], CENet architecture, denoted as $\phi_{\mathbf{e}}(\mathbf{o_t^H})$, is developed to estimate and infer a latent representation $\mathbf{z}_t$ of the environment based on the historical observations $\mathbf{o}_t^H$ as shown in Fig. 5. The latent representation is widely adopted in RL policy for the quadruped robot, such as characterizing the terrain properties [2] [46]. In CENet, the usage of the auto-encoding mechanism brings the advantages in robot's forward and backward dynamics learning and thus aligns the robot

TABLE I
REWARD STRUCTURE

Term	Equation	Weight
linear velocity tracking	$\exp\{-\lvert\mathbf{v_{xy}} - \mathbf{v_{xy}}^{cmd}\rvert^2/0.25\}$	1.0
angular velocity tracking	$\exp\{-\lvert\omega_\mathbf{z} - \omega_\mathbf{z}^{cmd}\rvert^2/0.25\}$	0.5
z velocity	v_z^2	-2.0
roll-pitch velocity	$\lvert\omega_{xy}\rvert^2$	-0.05
orientation	$\lvert\mathbf{g}\rvert^2$	-0.2
joint accelerations	$\lvert\ddot{\mathbf{q}}\rvert^2$	$-2.5e-7$
joint power	$\lvert\tau\rvert\lvert\dot{\mathbf{q}}\rvert$	$-2e-5$
body height	$(\mathbf{h} - \mathbf{h}^{des})^2$	-1.0
foot clearance	$(p_{f\ z\ k}^{\text{des}} - p_{f,z,k})^2 \cdot v_{f,xy,k}$	-0.01
action rate	$(\mathbf{a}_t - \mathbf{a}_{t-1})^2$	-0.01
action smoothing	$(\mathbf{a}_t - 2\mathbf{a}_{t-1} + \mathbf{a}_{t-2})^2$	-0.01
power distribution	$var(\tau \cdot \dot{\mathbf{q}})^2$	$-e-5$

dynamics with the environment changes more effectively. The output of the encoder also includes an estimation of the body velocity $\mathbf{v}_t$, which is shown to be able to enhance the locomotion policy's robustness [47] by eliminating the accumulated estimation drift. Different from [6], in the proposed framework, we design another encoder output $\mathbf{q}_r$, which is the estimation of the next joint position $\mathbf{q}_{t+1}$. This is not only beneficial to the dynamics learning, but also decouples the reference joint command from the action. In this manner, $\mathbf{q}_r$ is a more specific goal for PD tracking over the action $\mathbf{a}$ and provides a way to integrate the adaptive control defined later. The loss function of the CENet composes of two parts as follows:

$$\mathcal{L}_{\text{CE}} = \mathcal{L}_{\text{est}} + \mathcal{L}_{\text{VAE}}, \tag{6}$$

where $\mathcal{L}_{\text{est}}$ denotes the MSE estimation loss and is defined as

$\mathcal{L}_{\text{est}} = MSE(\tilde{\mathbf{v}}_t, \mathbf{v}_t) + MSE(\mathbf{q}_r, \mathbf{q}_{t+1})$, and $\mathcal{L}_{\text{VAE}}$ denotes a standard β-variational auto-encoder [48]–[50], which leverages the MSE for the reconstruction loss and Kullback-Leibler (KL) divergence as the latent loss. Details can be found in [6]. Note that for RL2AC, the RL part is not limited to [6] and can be specified as other frameworks by adding an encoder module to learn $\mathbf{q_r}$.

B. Adaptive control

It has been shown in Section IV-B that the term $(\mathbf{a} - \mathbf{q})$ can be regarded as certain embodied representation of the contact force/torque. We assume that this representation is also valid for the unknown torque uncertainty $\Delta\tau$. Further, since $\mathbf{q_r}$ is the estimation of $\mathbf{q}'$ obtained in the simulation, we can have:

$$\Delta\tau \approx \mathbf{K_u}(\mathbf{a} - \mathbf{q_r}), \tag{7}$$

where $\mathbf{K_u}$ denotes the unknown ideal gain matrix that can well fit Eq. (7). Hence, Eq. (4) can be rewritten as:

$$\begin{aligned}\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) - \mathbf{J}^{\mathbf{T}}\mathbf{F} - \Delta\tau \approx \mathbf{K_p}(\mathbf{a} - \mathbf{q_r}) \\ + \mathbf{K_u}(\mathbf{a} - \mathbf{q_r}) + \mathbf{K_p}(\mathbf{q_r} - \mathbf{q}) - \mathbf{K_d}\dot{\mathbf{q}}.\end{aligned} \tag{8}$$

Assuming the validity of the assumption in Eq. (7), the reference joint position $\mathbf{q_r}$ remains the equilibrium point of the system in Eq. (8). This implies that the quadruped robot can perform similarly to the case without model uncertainties if the additional torque can be effectively compensated.

Inspired by the regressor-based adaptive control, let $\phi = (\mathbf{a} - \mathbf{q_r})$ and $\hat{\mathbf{K}}_\mathbf{u}$ be the estimation of the gain matrix $\mathbf{K_u}$. An adaptive controller is proposed as:

$$\tau = \underbrace{(\mathbf{K_p} + \hat{\mathbf{K}}_\mathbf{u})\phi(\mathbf{a}, \mathbf{q_r})}_{feedforward} - \underbrace{\mathbf{K_s}\mathbf{s}}_{feedback} \tag{9}$$

where $\mathbf{K_s}$ denotes a gain matrix and

$$\mathbf{s} = \dot{\mathbf{q}} - \alpha(\mathbf{q_r} - \mathbf{q}). \tag{10}$$

It consists of the feedforward term used to compensate the dynamics and feedback term developed for the reference joint position tracking. Hence, for the proposed controller in Eq. (9), it attempts to compensate the dynamics so as to minimize the tracking errors $(\mathbf{q} - \mathbf{q_r})$.

In order to speed up the convergence of both tracking errors and parameter estimation errors, a composite adaptive controller [38] [39] is developed. First, according to Eq. (9), a prediction error is defined as:

$$\varepsilon = \tau_r - \tau \tag{11}$$

where τ_r denotes the actual torque acted on the quadruped robot joints and is measured by the torque sensor. A bounded-gain-forgetting (BGF) composite adaptive law is specified as follows:

$$\dot{\hat{\mathbf{K}}}_\mathbf{u} = -\mathbf{\Gamma}(t)\phi(\mathbf{a}, \mathbf{q_r})(\mathbf{s}^\mathbf{T} + \kappa\varepsilon^\mathbf{T}) \tag{12}$$

where κ is a weight factor and $\mathbf{\Gamma}(t)$ with $\mathbf{\Gamma}(0) = \mathbf{\Gamma}_0$ is a positive-definite matrix updated by [38]

$$\frac{d}{dt}\mathbf{\Gamma}^{-\mathbf{1}}(t) = -\lambda(t)\mathbf{\Gamma}^{-\mathbf{1}}(t) + \phi(\mathbf{a}, \mathbf{q_r})\phi(\mathbf{a}, \mathbf{q_r})^\mathbf{T} \tag{13}$$

and $\lambda(t)$ is a variable forgetting vector specified as:

$$\lambda(t) = \lambda_0(1 - \frac{\|\mathbf{\Gamma}(t)\|}{k_0}) \tag{14}$$

in which λ_0 and k_0 are constant values, denoting the upper bound of $\lambda(t)$ and $\|\mathbf{\Gamma}(t)\|$, respectively. It is shown in [38] [39] that ideally, the BGF composite adaptive law is able to achieve global asymptotic stability in the sense that all closed-loop signals are bounded and the tracking error $\mathbf{s}$ and the prediction error ε asymptotically converge to 0.

For the proposed controller defined in Eq. (9), if $\mathbf{K_s}$ is specified as $\mathbf{K_s} = \mathbf{K_d}$ and α satisfies $\alpha\mathbf{K_d} = \mathbf{K_p}$, it can be rewritten as:

$$\begin{aligned}\tau &= (\mathbf{K_p} + \hat{\mathbf{K}}_\mathbf{u})\phi(\mathbf{a}, \mathbf{q_r}) - \mathbf{K_d}(\dot{\mathbf{q}} - \alpha(\mathbf{q_r} - \mathbf{q})) \\ &= \underbrace{\hat{\mathbf{K}}_\mathbf{u}\phi(\mathbf{a}, \mathbf{q_r})}_{compensator} + \underbrace{\mathbf{K_p}(\mathbf{a} - \mathbf{q}) - \mathbf{K_d}\dot{\mathbf{q}}}_{PD\ control}.\end{aligned} \tag{15}$$

The proposed controller only introduces a compensator term compared with the original PD controller in Eq. (3) and therefore is easy to be directly implemented on most RL policies. In addition, the added compensator is specified in low-level and the update law defined in Eq. (12)-(14) can run as fast as the PD controller. Hence, it can react more quickly to the variations of the external contact forces/torques compared with RL based online fine-tuning methods and exhibits the superiority in dynamic response and rapid adaptability. The whole algorithm is summarized in Algorithm 1.

Algorithm 1 Implementation of RL2AC on quadruped robot

Require: $\pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{v}_t, \mathbf{z}_t)$, $\phi_\mathbf{e}(\mathbf{o_t^H})$, κ, λ_0, k_0.

procedure RL2AC
  while not stop **do**
    get $\mathbf{a} \sim \pi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{v}_t, \mathbf{z}_t)$ and $\mathbf{q_r} = \phi_\mathbf{e}(\mathbf{o_t^H})$;
    repeat
      update $\lambda(t)$ from Eq. (14), $\mathbf{\Gamma}(t)$ from Eq. (13), $\mathbf{s}$ from Eq. (10), ε from Eq. (11), $\hat{\mathbf{K}}_\mathbf{u}$ from Eq. (12);
      get torque τ from Eq. (9);
      send τ to robot and get feedback $\mathbf{q}$;
    until next $\mathbf{a}$ and $\mathbf{q_r}$
    get $\mathbf{o}_t, \mathbf{v}_t, \mathbf{z}_t, o_t^H$;
  end while
end procedure

VI. SIMULATION

The proposed method is evaluated on a legged robot named as *Unitree A1*. To train the policy, we used NVIDIA's Isaac Gym simulator [51] and code adapted from the legged_gym environment [52]. The policy was run on a laptop equipped with a NVIDIA GeForce RTX 3060 GPU and Intel Core i7-12700H CPU. The control frequency of the policy was 50Hz. The value, policy and CENet network were jointly trained for 8000 iterations and PPO was adopted for policy training. The training parameter setting was the same as [6]. For each iteration, the data was collected in parallel from 3000 environments. The Adam optimizer was used to minimize the value and policy loss with a learning rate $1e^{-3}$. The details of

the domain randomization are listed in Table II, which is the same as [6]. In both simulation and experiment, we specify some notations as follows:

- **Original**: using policy $\pi_{\mathbf{d}}$ [6] without disturbances and RL2AC;
- **Baseline**: using policy $\pi_{\mathbf{d}}$ with disturbances but without RL2AC;
- **Proposed**: using both policy $\pi_{\mathbf{d}}$ and RL2AC with disturbances.

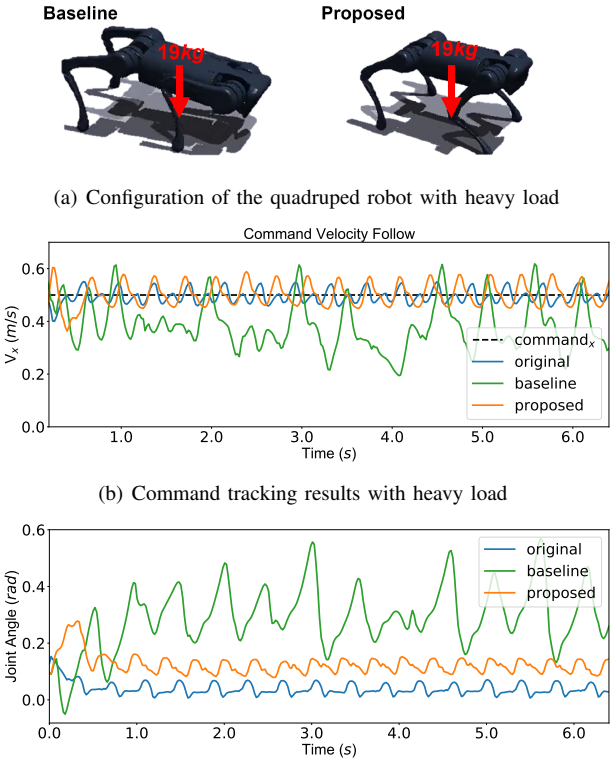
TABLE II
DOMAIN RANDOMIZATION DURING POLICY TRAINING

Parameter	Randomization range	Unit
Payload	[-1,2]	kg
$\mathbf{K_p}$ factor	[0.9,1.1]	Nm/rad
$\mathbf{K_d}$ factor	[0.9,1.1]	Nm/rad
Motor strength factor	[0.9,1.1]	Nm
Center of mass shift	[-0.05,0.05]	m
Friction coefficient	[0.2,1.25]	-

A. Heavy load

The first simulation is to verify the performance of RL2AC against the unknown heavy payload. The robot was required to walk forward with command $[0.5, 0, 0](m/s)$. We intentionally added a $19kg$ heavy payload on the base of the A1 robot, which was much heavier than it encountered during the training ($-1kg \sim 2kg$) and nearly twice of itself weight. The control parameters of the proposed method were specified as: $\kappa = 1.2, \lambda_0 = 3.0, k_0 = 20$.

The existence of the unseen heavy load induces the state-action distribution shift, resulting in the performance degradation. As seen on the left side of Fig. 6(a), the center of mass (CoM) of a quadruped dog is severely lowered with the heavy load, resulting in the poor command tracking performance compared with the original one without payload as illustrated in Fig. 6(b). With the proposed RL2AC, it can resist the heavy load to keep straightly and follow the command much better than the baseline. This is due to the appropriate feed-forward compensator generated to eliminate the affect of the heavy load to the locomotion, which is verified in Fig. 6(c) such that the hip joint of left front (LF) leg with the proposed method is much closer to the original one and changes within small range. This indicates that robot can maintain the same height as before and walk forward steadily. Such improvement is also clearly demonstrated in the comparison of the gait pattern as indicated in Fig. 7, where the colored part represents the contact time of each leg during locomotion. With heavy load, it is difficult for robot to swing its legs by using baseline, resulting in four legs consistently touching the ground for a long duration, making it challenging to track the provided command. This is significantly alleviated when using RL2AC, which implies that RL2AC narrows the gap between the state-action distribution of without/with $19kg$ payload. The gain changes corresponding to the right front (RF) leg are presented in Fig. 8. Since the heavy payload was acted on CoM, it mainly affected the thigh and knee joints and more torque compensators were required in these two joints, which



(a) Configuration of the quadruped robot with heavy load

(b) Command tracking results with heavy load

(c) Joint position comparison with heavy load

Fig. 6. Locomotion with heavy load

is coordinated with the results in Fig. 8, where the gains of the thigh and knee legs are increased more significantly so as to provide sufficient torques to counteract the external disturbances.

To further demonstrate the robustness of RL2AC against the heavy load, we conducted another simulation such that we varied the payload from $4kg$ to $17kg$ when the quadruped robot was climbing $0.15m$ high stairs, as illustrated in Fig.9(a). The remaining setting kept the same and the control parameter was slightly modified as $\kappa = 2.2$. For the baseline, the quadruped robot failed to climb the stairs when the payload was increased to $11kg$, while its counterpart is $14kg$ by using

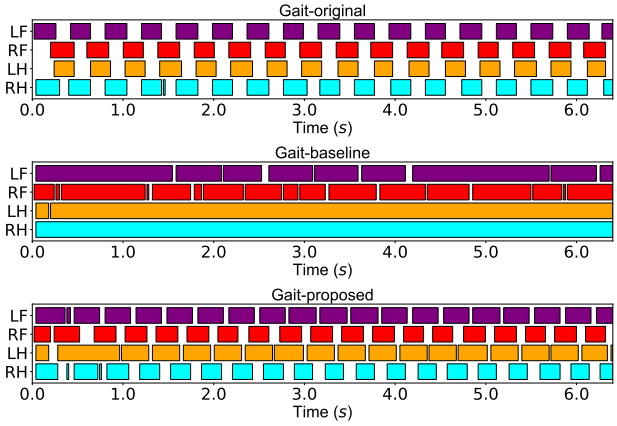


Fig. 7. Gait comparison with heavy load

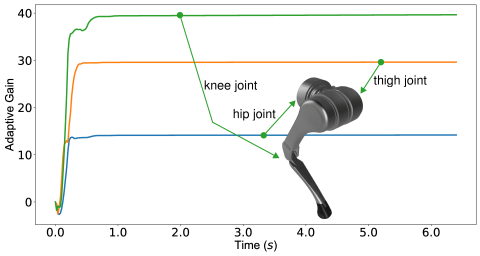
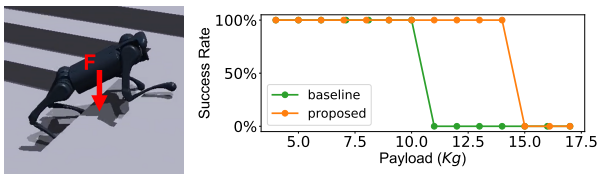


Fig. 8. Gain change of RF leg with heavy load



(a) Up stairs (b) Command tracking results with heavy load

Fig. 9. Climb the stairs with heavy load

RL2AC. The utilization of the proposed method leads to the increased payload up to 40% over the baseline, eventually.

Domain randomization is one of effectiveness way to enhance the robustness of the quadruped robot against various terrains or disturbances. In our framework, domain randomization is also adopted during the training stage. The proposed RL2AC is an augmentation upon the domain randomization and can further enhance the robustness of the robot, especially in the presence of the apparent gap between training and deployment. Such improvement is clearly shown in the simulation where the quadruped robot was asked to follow the given command $[0.5, 0, 0](m/s)$ on the plane when the payload was varied from $[-5, 15](kg)$ with $1(kg)$ increment per test. During the training stage, the payload added to the quadruped robot was randomly sampled from $[-1, 2](kg)$. In the deployment stage, the robot can well track the given velocity commands by using baseline without RL2AC as illustrated in Fig. 10, even if the payload has already out of the range $[-1, 2](kg)$. This illustrates the effectiveness of the domain randomization. However, the tracking performance may gradually descend when the gap between training and deployment continues to widen, in particular, a rapid descent when the payload is greater than $5(kg)$. This signifies that the effective range of the domain randomization is limited and it may lose its power when the environment is significantly different from its training stage. The proposed RL2AC is then used to alleviate such issues and further enhance the robot robustness against diverse environments. As seen in Fig. 10, the tracking performance with the proposed method is observed with a visible improvement over the baseline, even in the presence of significant load differences between training and deployment, which demonstrates that the proposed RL2AC method can further improve the robot performance upon the domain randomization.

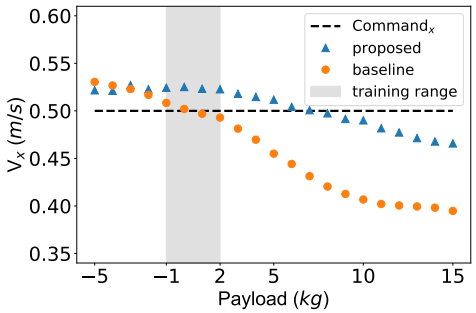
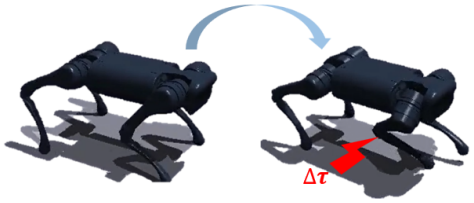
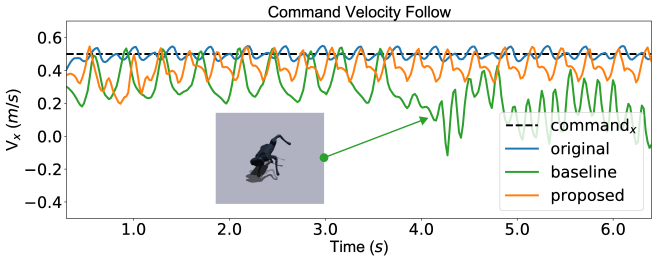


Fig. 10. The comparison of quadruped robot actual velocity in $x-axis$ with various payloads



(a) Quadruped robot configuration under torque disturbances



(b) Command tracking results under torque disturbances

Fig. 11. Locomotion under external torque disturbances

B. *Torque disturbances on RF leg*

For second setting, we manually added an additional torque disturbance $[-10.3, 4.82, 5.88](Nm)$ to the RF leg, which significantly enlarged the gap between the training and the implementation, and pose the extra difficulties in the generalization of the policy. The robot was still required to walk forward with command $[0.5, 0, 0](m/s)$. The control parameter was slightly modified as $\kappa = 1.2$.

The configuration change of the quadruped robot after adding the torque disturbance is shown in Fig. 11(a), resulting in the unnatural inward movement of RF leg. This dramatically

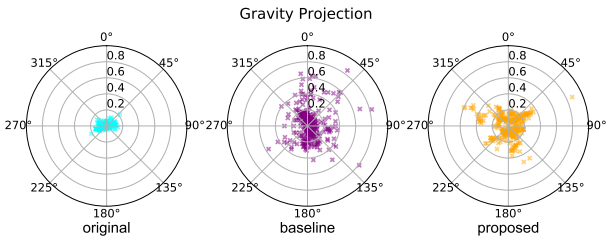


Fig. 12. Comparison of gravity projection

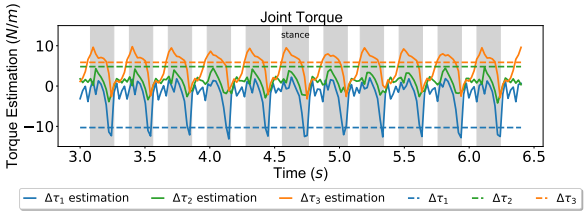


Fig. 13. Torque estimation of the joints on RF leg

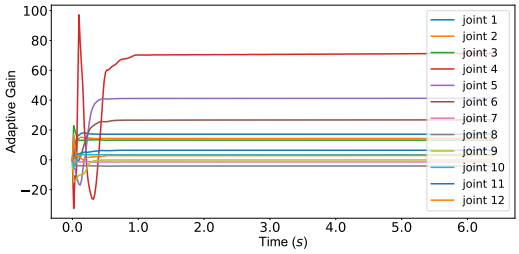


Fig. 14. Gain change in the presence of disturbances on RF leg

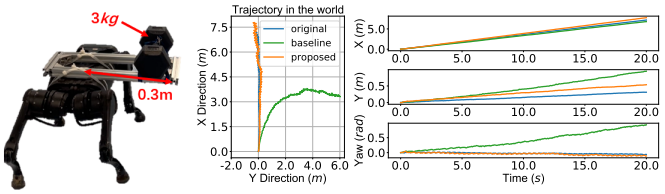
reduces the stable margin during locomotion. Therefore, the quadruped robot failed to follow the given command after $3.7s$ and then fall down as illustrated in Fig. 11(b). For the proposed RL2AC, it was still able to control the robot to move forward around $0.405(m/s)$, in the presence of the external torque and achieved a similar performance to the original case. The improvement is also visible referring to the change of the quadruped robot gravity projection as shown in Fig. 12, where CoM of the robot titled to the front right due to the existence of the torque disturbance acted on RF leg by using baseline. For RL2AC, it corrected the movements of the RF leg by identifying and compensating the disturbance and the corresponding disturbance estimation results of the RF leg are shown in Fig. 13. Note that the torque compensation is only activated during the stance phase and it can be seen that once the RF leg contacted the ground, RL2AC generated a torque to counteract the affect of the disturbance, resulting in the less CoM shift as indicated in Fig. 12. The corresponding gain changes of the 12 joints are presented in Fig. 14 and among them, joints 4, 5, 6 corresponding to the hip, thigh and knee joints of the RF leg change more significantly than remaining in order to alleviate the affects of the external disturbance.

VII. EXPERIMENT

For the physical implementation, the policy is directly used to control the A1 robot and the control frequency of the online adaptation part is 1000Hz.

A. Lateral torque disturbances

To verify its performance against the unknown disturbances, in this experiment, we put a shelf on the robot base and added a $3kg$ dumbbell (one quarter of A1 robot weight) on the right side of the shelf about $0.3(m)$ away from its base origin as shown in Fig.15(a). Such configuration has not be seen in the



(a) Experiment setup (b) Comparison of robot actual path

Fig. 15. Robot actual path

training stage. The control parameters of the proposed method were specified as: $\kappa = 4.5, \lambda_0 = 1.0, k_0 = 20$.

The existence of the dumbbell applies a lateral torque on the robot body and causes the robot deflecting to the right. Hence, it cannot walk straightly even if the command was $[0.5, 0, 0](m/s)$, as shown in Fig. 15(b), where the original, baseline and propose denote the quadruped robot paths without dumbbell, with dumbbell but without RL2AC and with both dumbbell and RL2AC, respectively. It demonstrates that without RL2AC, robot cannot resist the lateral torque disturbances and was deviated to the right gradually. The lateral torque had a significant impact on the robot yaw, resulting in the path deviation. For RL2AC, it corrected the movements of the legs through compensating the lateral torque by updating the gains of the related joints, as illustrated in Fig. 16. Since the dumbbell was located at the right side of the base, the RF and right hind (RH) legs undertook more pressures. Consequently, the gains of the knee joints 3, 9 on the RF and RH change more significantly. Note that the order of legs in simulator and reality is reversed. This is also detected on the tracking errors of $\mathbf{q} - \mathbf{q_r}$ in Fig. 17, where the corresponding joint position tracking errors of the RF and RH legs fluctuate more severely. However, RL2AC can always reduce the tracking errors through proper feed-forward compensation once it increased and all the joints position tracking errors converge to be within a bound. This verifies the effectiveness of the proposed method against the lateral torque disturbance. We further compare the RF leg gait frequency and joint amplitude of the original, baseline and proposed. The closer the gait frequency and amplitude are to the original one, the more effective the controller is in dealing with and compensating for lateral force disturbances. This implies that the quadruped robot can maintain similar gaits even in the presence of external disturbances. For Fig. 18(b), the middle point denotes the joint position mean value and up and down bar represent the maximum and minimum values, respectively. Therefore, we can conclude that RL2AC can well handle the dynamics shift issue by properly compensating the disturbances, achieving the similar performance with the original one without disturbances.

B. Slow motion

When the RL policy is transferred to the reality, normally it is hard to control the quadruped robot to perform accurate slow locomotion, i.e., moving forward at a small speed less than $0.2(m/s)$. One possible reason is that the smaller the speed is, the greater the impact of friction and model

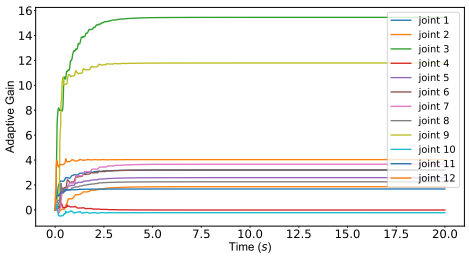


Fig. 16. Gain change with lateral torque disturbance

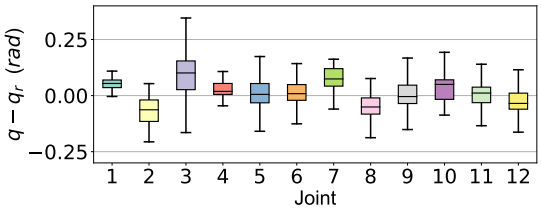
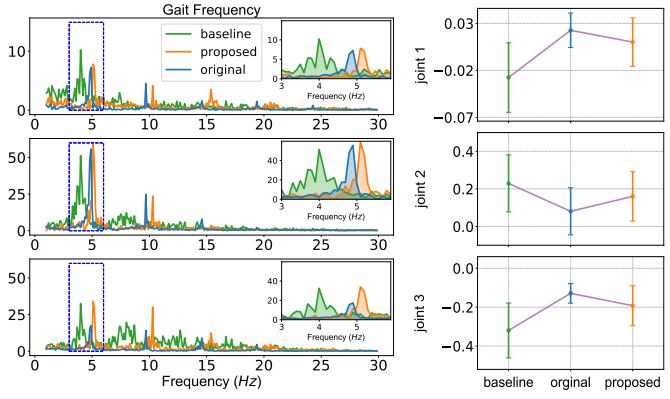


Fig. 17. $\mathbf{q} - \mathbf{q_r}$ tracking results

uncertainty have in reality. This results in a more serious sim-to-real issue and therefore can be used to verify the performance of the proposed RL2AC against the sim-to-real gap. The control parameters in this experiments were specified as:$\kappa = 10, \lambda_0 = 1.0, k_0 = 20$. Note that there was no any disturbance considered in the test and the comparison was between the baseline and the proposed one by following a small speed command.

The performance of both two methods for tracking command $[0.2, 0, 0](m/s)$ is given in Fig. 19, where the mean actual speed of the baseline and the proposed are $0.040(m/s)$ and $0.127(m/s)$, respectively, exceeding 200% improvement. Due to the existence of the proper feed-forward torque compensation by using RL2AC, each joint received sufficient power to overcome the friction and model uncertainties and therefore was able to move smoothly for catching up the given speed command. This is illustrated in Fig. 20, where joints with the proposed method rotate periodically. As for the baseline, the joints 1-4 situated on the RF and LF legs exhibited irregular movement, appearing to encounter resistance before



(a) Gait frequency analysis (b) Gait amplitude analysis

Fig. 18. Gait analysis

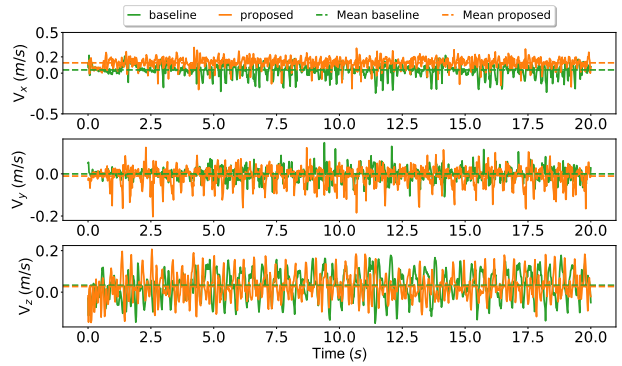


Fig. 19. Command tracking results at small speed $0.2(m/s)$

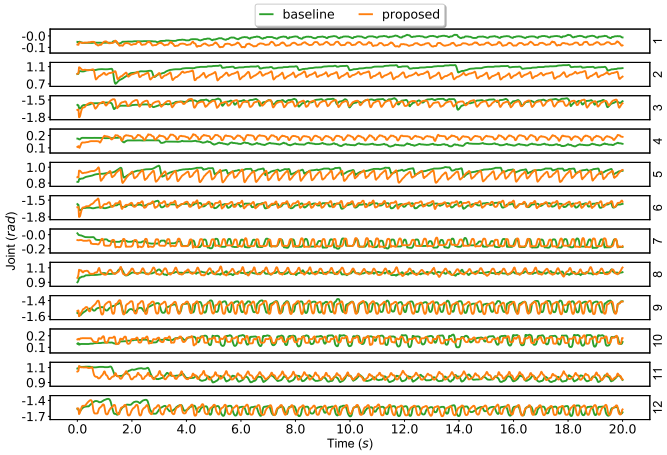


Fig. 20. Comparison of 12 joints' motions

suddenly progressing, as depicted in Fig. 20. This behavior resembled a scenario where the back legs pushed the front legs forward, as observed in the accompanying video. Moreover, we conducted more tests by specifying the desired command from $0.1(m/s)$ to $0.3(m/s)$ with $0.02(m/s)$ increment and the results are presented in Fig. 21, which demonstrates a significant improvement of RL2AC over the baseline and verifies the performance of RL2AC against the sim-to-real gap.

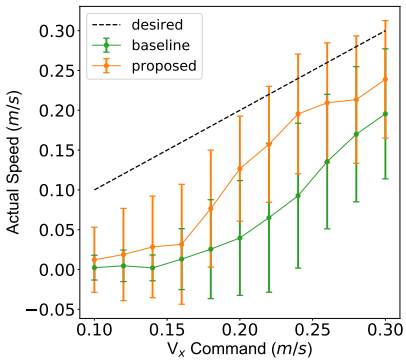
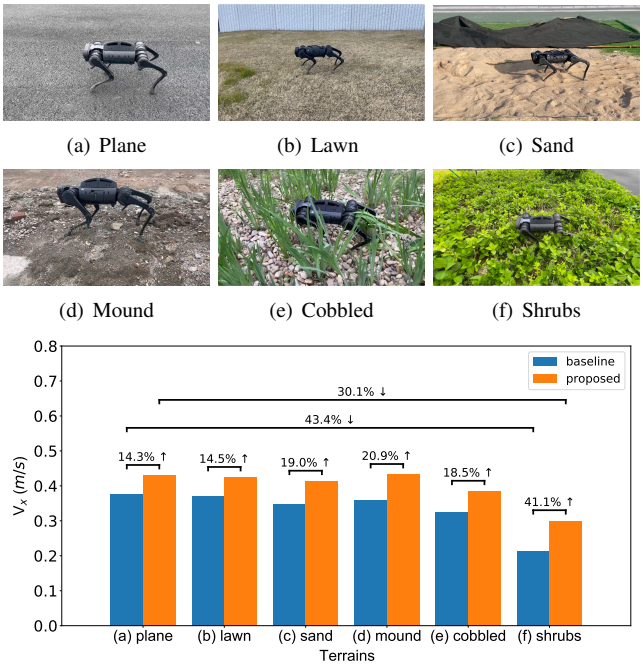


Fig. 21. Slow motion tests

C. Various terrains

In this experiment, we show the performance of the proposed RL2AC method against rough terrains. Specifically, the robot was required to follow the velocity command $[0.5, 0, 0](m/s)$ when locomoting on various terrains and the command velocity tracking errors were evaluated. The control parameters in this experiments were specified as:$\kappa = 25, \lambda_0 = 1.0, k_0 = 20$. We tested the quadruped robot on various terrains including plane, lawn, sand, mound, cobbled and shrubs, as illustrated in Fig. 22(a)-Fig. 22(f) and the experimental comparison with the baseline, that solely used learned policy, is illustrated in Fig. 22. The velocity of quadruped robot was estimated by using a Kalman filter [53], which may not be very accurate. However, it can roughly reflect the actual velocity and is valid to evaluate the velocity tracking performance of the quadruped robot. As seen in Fig. 22(g), the actual velocity of the quadruped robot with the proposed RL2AC is much closer to the desired velocity and demonstrates an apparent improvement on the velocity tracking with about 21.38% promotion in average over the baseline on 6 terrains. Additionally, when the rough level of terrains is increased, for example from (a)plane to (f)shrubs, the velocity tracking performance gradually descends, implying the enlargement of the sim-to-real gap. Such performance decline mainly stems from varied GRF on different terrains. In particular, when the robot walked into the shrubs, its legs were often entangled by intricate branches and vines, imposing additional forces/torques. The existence of the adaptive mechanism in proposed method, was able to adjust the feedforward torques to alleviate the influence of varied GRF and additional disturbances, resulting in a better performance in various terrains with less velocity decrease than its counterpart (30.1% vs 43.4%) as depicted in Fig.22(g). This verifies the effectiveness of the proposed RL2AC against rough terrains.



(g) Actual body velocity on various terrains

Fig. 22. Comparison of command tracking performance on various terrains

VIII. Conclusion

We aim to delve into and comprehend the control mechanism underlying the RL policy employed in quadruped robot locomotion. As a result of this exploration, we introduce a new framework named RL2AC, designed for robust locomotion by seamlessly integrating RL and traditional adaptive control. Simulation and experimental results substantiate the effectiveness of RL2AC in mitigating various disturbances, both in simulated environments and the physical world.

The improvements of RL2AC over the baseline primarily stem from the feedforward torque compensation calculated based on the joint position tracking errors $(\mathbf{q} - \mathbf{q_r})$, assisting the quadruped robot to handel the model uncertainties during the deployments. As seen in Fig.2, during the swing phase, the tracking errors are small, resulting in the limited torque compensation. Therefore, RL2AC mainly affects the stance phase and is hard to deal with the model uncertainties exerted on the swing phase. Essentially, it is a trajectory tracking control problem during leg swing, which has been extensively explored in the existing literatures. There are a lot of methods used to tackle the model uncertainties for trajectory tracking tasks. In future work, these methods can be further integrated into the proposed control framework to improve the robustness against the disturbances during the swing phase. In addition, RL2AC method is able to effectively alleviate the sim-to-real issue and enhance the robot robustness against the disturbances, while its capability to handle totally unknown tasks is limited. The RL2AC method aims to assist the robot to recovery its original performance when suffering from the sim-to-real gap or the external disturbances. If its original performance is poor, no much improvement can be earned with RL2AC. Generalization to totally unknown tasks is crucial for robot, especially when deployed in un structured environment. It is also one of key issues in robot learning domain, that we will continuous to explore and contribute in future.

IX. Acknowledgement

This work was supported by the National Science and Technology Innovation 2030-Major Projects (Grant No. 2022ZD0208800) and the National Natural Science Foundation of China (Grant No. 62003018 and 62176215).

References

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

[2] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," in *Robotics: Science and Systems*, 2021.

[3] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," *arXiv preprint arXiv:2205.02824*, 2022.

[4] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," *arXiv preprint arXiv:2309.14341*, 2023.

[5] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *arXiv preprint arXiv:2306.14874*, 2023.

[6] I. M. A. Nahrendra, B. Yu, and H. Myung, "Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5078–5084.

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

[8] J. R. Rebula, P. D. Neuhaus, B. V. Bonnlander, M. J. Johnson, and J. E. Pratt, "A controller for the littledog quadruped walking on rough terrain," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 1467–1473.

[9] J. Buchli, M. Kalakrishnan, M. Mistry, P. Pastor, and S. Schaal, "Compliant quadruped locomotion over rough terrain," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 814–820.

[10] A. Spröwitz, A. Tuleu, M. Vespignani, M. Ajallooeian, E. Badri, and A. J. Ijspeert, "Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 932–950, 2013.

[11] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. De Pieri, and D. G. Caldwell, "A reactive controller framework for quadrupedal locomotion on challenging terrain," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2554–2561.

[12] M. Hutter, C. Gehring, M. A. Höpflinger, M. Blösch, and R. Siegwart, "Toward combining speed, efficiency, versatility, and robustness in an autonomous quadruped," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1427–1440, 2014.

[13] M. Hutter, H. Sommer, C. Gehring, M. Hoepflinger, M. Bloesch, and R. Siegwart, "Quadrupedal locomotion using hierarchical operational space control," *The International Journal of Robotics Research*, vol. 33, no. 8, pp. 1047–1062, 2014.

[14] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 38–44.

[15] C. D. Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic locomotion and whole-body control for quadrupedal robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3359–3365.

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

[17] D.-E. Argiropoulos, D. Papageorgiou, M. Maravgakis, D. Drosakis, and P. Trahanias, "Two-layer adaptive trajectory tracking controller for quadruped robots on slippery terrains," *arXiv preprint arXiv:2304.00804*, 2023.

[18] Z. Li, J. Zeng, A. Thirugnanam, and K. Sreenath, "Bridging Model-based Safety and Model-free Reinforcement Learning through System Identification of Low Dimensional Linear Models," in *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.

[19] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[20] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[21] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

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

[23] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics (ToG)*, vol. 40, no. 4, pp. 1–20, 2021.

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[25] A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel, "Adversarial motion priors make good substitutes for complex reward functions," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 25–32.

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

[27] Q. L. J. G. L. C. J. P. Junfeng Long, Zirui Wang, "Hybrid internal model: Learning agile legged locomotion with simulated robot response," *Arxiv*, 2023.

[28] Q. Yao, J. Wang, D. Wang, S. Yang, H. Zhang, Y. Wang, and Z. Wu, "Hierarchical terrain-aware control for quadrupedal locomotion by combining deep reinforcement learning and optimal control," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4546–4551.

[29] X. Da, Z. Xie, D. Hoeller, B. Boots, A. Anandkumar, Y. Zhu, B. Babich, and A. Garg, "Learning a contact-adaptive controller for robust, efficient legged locomotion," in *Conference on Robot Learning*. PMLR, 2021, pp. 883–894.

[30] J. Wu, G. Xin, C. Qi, and Y. Xue, "Learning robust and agile legged locomotion using adversarial motion priors," *IEEE Robotics and Automation Letters*, 2023.

[31] H. Shi, B. Zhou, H. Zeng, F. Wang, Y. Dong, J. Li, K. Wang, H. Tian, and M. Q.-H. Meng, "Reinforcement learning with evolutionary trajectory generator: A general approach for quadrupedal locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3085–3092, 2022.

[32] S. Lyu, H. Zhao, and D. Wang, "A composite control strategy for quadruped robot by integrating reinforcement learning and model-based control," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 751–758.

[33] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 2908–2927, 2022.

[34] J. J. E. Slotine, W. Li *et al.*, *Applied nonlinear control*. Prentice hall Englewood Cliffs, NJ, 1991.

[35] P. A. Ioannou and J. Sun, *Robust adaptive control*. PTR Prentice-Hall Upper Saddle River, NJ, 1996, vol. 1.

[36] C. C. Cheah, C. Liu, and J.-J. E. Slotine, "Adaptive tracking control for robots with unknown kinematic and dynamic properties," *International Journal of Robotics Research*, vol. 25, no. 3, pp. 283–296, 2006.

[37] S. Lyu and C. C. Cheah, "Human-robot interaction control based on a general energy shaping method," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2445–2460, 2019.

[38] J.-J. E. Slotine and W. Li, "Composite adaptive control of robot manipulators," *Automatica*, vol. 25, no. 4, pp. 509–519, 1989.

[39] Y. Pan and H. Yu, "Composite learning robot control with guaranteed parameter convergence," *Automatica*, vol. 89, pp. 398–406, 2018.

[40] M. V. Minniti, R. Grandia, F. Farshidian, and M. Hutter, "Adaptive clf-mpc with application to quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 565–572, 2021.

[41] M. Sombolestan and Q. Nguyen, "Adaptive force-based control of dynamic legged locomotion over uneven terrain," *arXiv preprint arXiv:2307.04030*, 2023.

[42] A. De Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger, "Collision detection and safe reaction with the dlr-iii lightweight manipulator arm," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 1623–1630.

[43] S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, 2017.

[44] Y. Wei, S. Lyu, W. Li, X. Yu, Z. Wang, and L. Guo, "Contact force estimation of robot manipulators with imperfect dynamic model: On gaussian process adaptive disturbance kalman filter," *IEEE Transactions on Automation Science and Engineering*, 2023.

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

[46] G. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," in *Robotics: Science and Systems*, 2022.

[47] 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 Robotics and Automation Letters*, vol. 7, no. 2, pp. 4630–4637, 2022.

[48] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[49] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "β-vae: Learning basic visual concepts with a constrained variational framework," in *International conference on learning representations*, 2016.

[50] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in β-vae," *arXiv preprint arXiv:1804.03599*, 2018.

[51] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance GPU-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

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

[53] L. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine, "Legged robots that keep on learning: Fine-tuning locomotion policies in the real world," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1593–1599.