Torque-based Deep Reinforcement Learning for Task-and-Robot Agnostic Learning on Bipedal Robots Using Sim-to-Real Transfer

Donghyeon Kim[1], Glen Berseth[2,*], Mathew Schwartz[3] and Jaeheung Park[1,4,5,6,*]

***Abstract*—In this paper, we review the question of which action space is best suited for controlling a real biped robot in combination with Sim2Real training. Position control has been popular as it has been shown to be more sample efficient and intuitive to combine with other planning algorithms. However, for position control gain tuning is required to achieve the best possible policy performance. We show that instead, using a torque-based action space enables task-and-robot agnostic learning with less parameter tuning and mitigates the sim-to-reality gap by taking advantage of torque control's inherent compliance. Also, we accelerate the torque-based-policy training process by pre-training the policy to remain upright by compensating for gravity. The paper showcases the first successful sim-to-real transfer of a torque-based deep reinforcement learning policy on a real human-sized biped robot.**

***Index Terms*—Reinforcement Learning, Humanoid and Bipedal Locomotion, Torque-based Control**

I. Introduction

Control algorithms for robots can be extremely complex and difficult to implement. They change widely based on the robot themselves and tasks, and a continuing line of robotics research has shown the difficulties faced when high-dof multi-link robots, such as biped robots, are controlled [1, 2]. Recently, reinforcement learning (RL) has become a popular method for implementing robotic controllers, in part due to its ability to be applied to challenging tasks and robots. For example, by introducing RL control methods to biped robots, challenging tasks could be performed while effectively utilizing the advantages of biped robots.

Biped robots (and other legged robots) make discontinuous contact with the ground, enabling greater mobility compared to wheeled robots. However, the discontinuous contact requires switching the contact discretely at proper timing and location [2, 3], making the biped robot challenging to control. Additionally, the control challenge varies depending on the task and robot, and it has resulted in designing solutions that work for more specific robots and tasks. For example, although a target pose control (position-based deepRL), which is a widely-used control space in RL, has many advantages, such as it is more explainable and it promotes fast learning [4], it comes at the cost of introducing what is essentially a mass-spring-damper system with gain parameters into the control design. When controlling biped robots, the gain parameters are often tuned for each robot and every task the robot performs [5, 6, 7, 8] as shown in Fig. 1. This tuning complicates the creation of robot-and-task-agnostic algorithms and requires manually searching PD gains, and adding the gains to the optimization space increases the size and difficulty of the optimization [8].

Additionally, since the reality gap exists between simulation and the real-world due to various factors, such as contact dynamics and state estimation gap [7, 9], we should promote control methods that can be used more broadly whether in a simulation or real robot. While transferring the trained policy to the robot hardware, the reality gap would result in a tracking performance difference and the timing of contact with the ground cannot exactly match between the trained environment and the real-world. However, when early contact occurs in the position-based deepRL policy, the PD controller will try to track the target position no matter how much torque is generated, and the robot can lose balance due to high impact force [10]. In this sense, compliance is advantageous to overcome the reality gap because compliance implicitly controls the energy transfer to the environment and reduces the impact force [11].

Therefore, it would be better to develop control methods that require less parameter tuning and reduce the reality gap, because they will scale better, be more reusable, and be independent of the robot platform, task, and environment (simulation or real). One option that fits nicely into this puzzle is to control the motor torques directly as shown in Fig. 1 because there is no PD gain that predetermines the behavior and the torque-based control is known to be compliant [10, 11]. Early work on deep RL [12] has compared position-based and torque-based deepRL and reported that the torque-based deepRL generally shows inferior performance with slow training. However, the performance was evaluated only on a simulator, and the Sim2Real problem was not taken into account. Therefore, it would be worthy to analyze the action space on the real robot from a robotics perspective.

In this paper, we revisit the question of which action-space is the best in general. First, from the learning perspective,

*This work was supported by the National Research Foundation of Korea (NRF) grant funded by Korea government (MSIT) (No.2021R1A2C3005914)

[1]Donghyeon Kim and Jaeheung Park are with the Department of Intelligence and Information, Seoul National University, Seoul, KS013, Republic of Korea kdh0429, park73@snu.ac.kr

[2]Glen Berseth is with the Université de Montréal, Quebec, H3C 3J7, Canada glen.berseth@mila.quebec

[3]Mathew Schwartz is with the College of Architecture and Design, New Jersey Institute of Technology, Newark, 07102, USA cadop@njit.edu

[4,5,6]Jaeheung Park is also with ASRI, RICS, Seoul National University, Republic of Korea, and Advanced Institutes and Advanced Institutes of Convergence Technology(AICT), Republic of Korea.

*Both are corresponding authors.

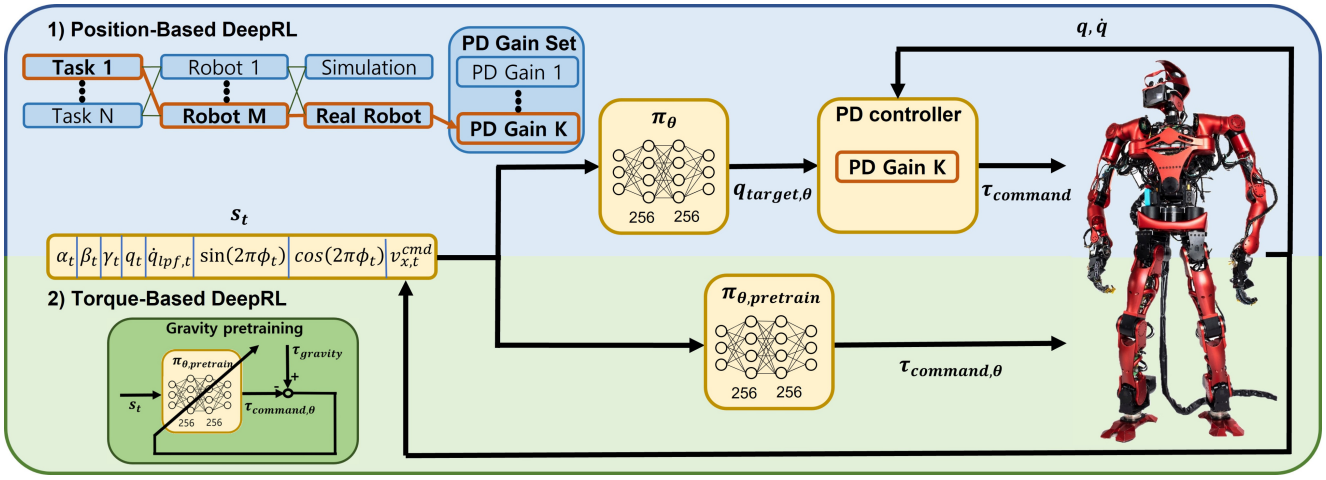

Fig. 1: Diagram of position-based deepRL and torque-based deepRL. For position-based deepRL, the PD gain which is proper for task 1, robot M, and sim-to-real transfer is tuned in advance. In torque-based deepRL, we propose to pre-train the policy with the gravity compensation torque.

we show that the torque-based deepRL is a general algorithm that can be applied out of the box without manually tuning gains for each robot (Atlas, TOCABI) or task (squat, walk, run) compared to position-based deepRL. Second, from the sim-to-real perspective, we include an analysis of real robot hardware and show that the position-based deepRL policy is more vulnerable to the reality gap due to its limited compliance. We also show that the torque-based deepRL policy is compliant by its nature, and effectively overcomes the reality gap. Third, since torque-based policies have been described as exhibiting slow training, we propose to pre-train the torque-based deepRL policy with a gravity compensation torque to accelerate initial training and reduce the sample inefficiency. Lastly, this is the first time that a torque-based deepRL policy is successful on a heavy human-sized biped robot (≈100kg) with stable walking. It is demonstrated that a biped robot can be torque-controlled at various control frequencies with RL, ranging from 250 Hz to as low as 62 Hz, which could suggest a new method to the torque control society. We provide evidence for these claims by comparing position-based and torque-based deepRL policies to each other, and this results in a single control algorithm that achieves a larger space of tasks without additional manual tuning and is adequate for real-world legged robots.

II. Related Work

A. Action Space on Legged Robots

When controlling a legged robot with RL, various action spaces can be designed. Not only action spaces that were compared in early work on RL by [12], but also any other action space such as task space [13] can be adopted by using an additional low-level controller to track the target output by the policy. The most widely used approach is a method that involves using a policy to generate a target joint angle q_{target,π_θ} and tracking it with a PD controller using gain parameters K_p, K_d (position-based deepRL policy). Plenty of works using the position-based deepRL emphasized the importance of selecting a proper gain. However, how to determine the PD gain in RL is not unified and decided by the rule of thumb. Since each task and robot requires a different amount of torque range, the PD controller requires some insight to set properly for each individual robot morphology and task [5, 6]. For example, [14] mentioned that they used a relatively very low PD gain on a humanoid robot that has heavy legs and a high-ratio gearbox. On the other side, [9] emphasized tuning the PD controller to behave similarly on the real robot and the simulation is critical because the gap of the PD controller response can result in non-observed states and increase instabilities.

Alternatively, the action space can be designed to generate the torque command directly from the policy. To the best of the authors' knowledge, the only work in this framework on a legged robot has been implemented by [15] on a quadruped robot. They implemented the torque-based deepRL policy with a relatively high control rate and showed successful sim-to-real outdoor walking. However, their work was implemented on a quadruped robot whose impact force could be low, and it has not been discussed which aspects caused the advantages of torque control. In this paper, we demonstrate that the torque-based deepRL policy can be applied on various robot platforms and tasks with minimal tuning along with a comparison of behavioral differences arising from intrinsic compliance.

B. Sim-to-Real

A fundamental solution to reduce the discrepancies between the real-world and the simulation is to bring the simulator closer to reality by developing a high-fidelity simulator [4], or identifying the robot model accurately [16]. However, since it is hard to completely imitate reality in the simulator, dynamics randomization is applied to train the robot robustly in a certain range of randomized parameters [17, 18]. In addition, implicitly identifying the randomized environment parameter in the latent vector and using the information as part of a state [19, 20] is also being adopted. In this paper, we show a new way to reduce the reality gap by exploiting the intrinsic compliance of the controller in the environment where contact is involved.

III. BACKGROUND

A. Reinforcement Learning

We model a robot control problem as a discrete-time Markov Decision Process (MDP) with a discounted expected return objective. At each time step t, the robot control policy π_θ observes a state s_t, and the policy samples an action $a_t \sim \pi_\theta(\cdot|s_t)$ based on the state. The agent applies the action, transitioning the robot state s_t to a new state $s_{t+1} \sim p(\cdot|s_t, a_t)$, and a reward $r_t = r(s_t, a_t, s_{t+1})$ is calculated accordingly. The agent tries to learn a policy through interaction with the environment that maximizes the expected return $J(\pi_\theta)$, which is the expected cumulative discounted reward over a finite-horizon T

$$J(\pi_\theta) = E_{\tau\sim p(\tau|\pi_\theta)}[\sum_{t=0}^{T-1} \gamma^t r_t], \quad (1)$$

where τ is the trajectory when executing the policy π_θ.

B. Biped Robot TOCABI

In this paper, we use a lab-made biped robot TOCABI for hardware verification. TOCABI is a human-sized humanoid that is 1.8 meters tall and weighs 100 kilograms [21]. It is designed according to the proportions of the human body and has 12 actuated joints in the lower body and 21 actuated joints in the upper body. Gears with a high reduction ratio of 100:1 are used due to heavy weight, and the current control-based servo drives, which communicate with the real-time control computer through the EtherCAT interface, control each joint motor with a sampling frequency of 2kHz.

IV. METHOD

To find the best algorithm that can be used to train various tasks on various robot platforms and can be transferred from simulation to the real-robot with minimal tuning, we set two environments. In these environments, the agent generates either target joint angles (position-based deepRL policy) or torque commands (torque-based deepRL policy) following the MDP formulation in Sec. III-A as follows. A diagram of each control method is also shown in Fig. 1, and in this paper, we focus on the lower body, and the upper body is PD controlled along with a gravity compensation to keep its default pose.

A. State Space

The state space $S \in R^{30}$ consists of the robot's base orientation in Euler angles $\alpha_t \in R, \beta_t \in R, \gamma_t \in R$, the joint position $q_t \in R^{12}$, the low-pass filtered joint velocity $\dot{q}_{lpf,t} \in R^{12}$, the phase information represented by sine and cosine $sin(2\pi\phi_t) \in R, cos(2\pi\phi_t) \in R$, and the command velocity $v_t^{cmd} \in R$.

$$s_t = [\alpha_t, \beta_t, \gamma_t, q_t, \dot{q}_{lpf,t}, sin(2\pi\phi_t), cos(2\pi\phi_t), v_t^{cmd}]. \quad (2)$$

The phase variable ϕ_t increments with time from 0 to 1 and then returns to 0 when it reaches 1 to represent the current phase of the cyclic motion. However, the phase ϕ_t can also be modulated with the phase modulation action $a_{\delta\phi,t}$ as it will be described in Sec. IV-B3. The command velocity v_t^{cmd} is given to the robot when the task involves tracking a user-input velocity (e.g. walk, run). To simulate the sensor noise, we inject a Gaussian noise $w_t \sim N(0, \sigma)$ to the joint position according to the encoder resolution of the robot hardware ($\sigma = 1e^{-4} rad$). A joint velocity $\dot{q}_{noise,t}$ is computed from the noise-injected joint position and then low-pass filtered with a cut-off frequency of 4Hz to be used as a state $\dot{q}_{lpf}$.

B. Action Space

The action space $A \in R^{13}$ is composed of 12 actions to generate joint commands and one action to modulate the phase ϕ_t. The action is sampled from a Gaussian distribution whose mean is the output of the policy (μ_θ), and the standard deviation (Σ) is fixed to a predetermined value $\pi_\theta \sim N(\mu_\theta, \Sigma^2)$. Depending on the control method, either position-based or torque-based deep RL policy generates commands as follows.

1) Position-based DeepRL: In the position-based deepRL method, the policy outputs the target joint angle, and the target joint angle is tracked by a low-level PD controller.

$$\tau_{cmd} = K_p(q_{\text{target},\pi_\theta} - q) + K_d(-\dot{q}). \quad (3)$$

In general, the actor-network updates the target joint angle with 30-100Hz, and the PD controller runs faster with 500-2000Hz.

In our implementation, the actor updates the action with 250Hz and the PD controller runs at 2000Hz. The default pose $q_{default}$ is set to a knee-bent standing configuration. Lastly, the standard deviation Σ is set to a scaled value of each joint position limit (q_{limit}/s_q) rather than the same value across all joints. This will respect the hardware design by allowing larger exploration at the joint whose joint limit is designed to be high to move a wider range of motion.

2) Torque-based DeepRL: A torque-based deepRL policy directly outputs the torque,

$$\tau_{cmd} = \pi_\theta. \quad (4)$$

In contrast to the position-based deepRL policy, whose behavior is largely predetermined by the PD gain, there is no inductive bias on the torque-based deepRL. The bias introduced from the gains has made learning policies for specific tasks easier but has also limited the reuse of control methods. Therefore, torque-based deepRL, as we will show, can be used as a more robot-and-task agnostic method for learning control policies. Also, because the generated control input is not restricted to a mass-spring-damper-like system, there is a higher possibility of achieving improved performance than the position-based deepRL policy. Additionally, as opposed to the position-based deepRL policy whose compliance is restricted by the PD controller, the torque does not abruptly change. This compliance results in robust and safe interaction with the environment, especially for unexpected contacts.

In our implementation, the actor updates the command torque with 250Hz. Also, the standard deviation is set to a scaled value of each joint torque limit ($\Sigma = \tau_{limit}/s_\tau$), and this will allow larger torque exploration to the joints whose torque capacity is large, respecting the hardware design.

3) Phase Modulation: The action is also composed with a phase modulation action $a_{\delta\phi,t}$. The phase variable ϕ_t represents the current phase in cyclic motion and the reference motion is updated accordingly. However, if the phase variable advances linearly with time by a fixed value, the period of the learned motion is also determined accordingly and cannot be modulated [6]. This would generate sub-optimal behavior when the robot is commanded to walk at high speed. The robot could generate a wide step with a fixed period T_{ref} to track the target speed, but it would be better if the robot could also reduce the walking period. Therefore, in our implementation, a phase modulation action $a_{\delta\phi,t}$ is additionally added to adjust the period and timing of the motion as follows.

$$\phi_{t+1} = fmod(\phi_t + \frac{\Delta t}{T_{ref}} + a_{\delta\phi,t}, 1.0) \tag{5}$$

The action space of phase modulation action $a_{\delta\phi,t}$ is designed to be within $[0, \Delta t]$.

C. Reward

The reward is formulated to execute a given task while imitating a reference motion. In addition, several regularization rewards are used for real-robot implementation. The overall reward is composed as follows.

$$\begin{aligned} r_t = & r_t^{base,imitate} + r_t^{q,imitate} + r_t^{c,imitate} + r_t^{v,task} \\ & + r_t^{\dot{q},reg} + r_t^{\ddot{q},reg} + r_t^{F,reg} + r_t^{\Delta F,reg} + r_t^{\tau,reg} + r_t^{\Delta\tau,reg} \end{aligned} \tag{6}$$

The reward can be categorized into imitation reward, task reward, and regularization reward, and each term is defined in Table I. Imitation rewards guide the robot to follow the reference base orientation expressed in quaternion $q_{base,t}^{ref}$, reference joint angle q_t^{ref}, and foot contact status. Specifically, the foot contact imitation reward $r_t^{c,imitate}$ is given if the current foot contact status $c_{r,t}, c_{l,t} \in \{0, 1\}$ matches with a predetermined desired foot contact status (double-support phase ϕ_{DSP} or single-support phase $\phi_{SSP,r}, \phi_{SSP,l}$). The task reward encourages the robot to track the command velocity v_t^{cmd} on locomotion tasks. Lastly, regularization rewards penalize joint velocity, joint acceleration, contact force, contact force difference, joint torque, and joint torque difference, respectively for real-world implementation.

D. Gravity Pre-training

Generally, since π_θ is initialized to a small value, the robot must first learn to support its weight in the early stages of learning, which deteriorates sample efficiency. To accelerate the initial training and increase sample efficiency, we propose a relatively weak inductive bias on the torque-based deepRL policy without damaging compliance. This method only requires pre-training a torque-based policy to maintain the initial pose of the robot (gravity compensation).

To collect the pre-training data, the gravity compensation torque-based policy is obtained based on the contact-consistent whole-body model [22]. The gravity torque is computed based on the base orientation $\alpha_t, \beta_t, \gamma_t$, current joint configuration q_t, and contact state of each leg $c_{r,t}, c_{l,t}$, and experience is collected across various configurations by randomly initializing the robot and perturbing the robot. Except for state variables that affect the gravity torque $(\alpha_t, \beta_t, \gamma_t, q_t, c_{r,t}, c_{l,t})$, other state variables are randomly set at each time step. Likewise, the phase modulation action $a_{\delta\phi,t}$ is also set to a random value. $200,000$ experience samples are collected on 8 parallel environments for 5 hours and required less than 10 minutes to pre-train the policy. Note that once the policy is pre-trained, it can be reused as long as the state space and action space are not modified.

TABLE I: Reward Definition

Reward	Definition
$r_t^{base,imitate}$	$0.3 \cdot exp(-13.2\|\|q_{base,t}^{ref} - q_{base,t}\|\|$
$r_t^{q,imitate}$	$0.35 \cdot exp(-4.0\|\|q_t^{ref} - q_t\|\|_2^2)$
$r_t^{c,imitate}$	$\begin{cases} 0.2 \text{ if } \begin{cases} c_{r,t}=1, c_{r,t}=1, \phi_t \in \Phi_{DSP} \\ c_{r,t}=1, c_{r,t}=0, \phi_t \in \Phi_{SSP,r} \\ c_{r,t}=0, c_{r,t}=1, \phi_t \in \Phi_{SSP,l} \end{cases} \\ 0 \text{ else} \end{cases}$
$r_t^{v,task}$	$0.3 \cdot exp(-3.0\|\|v_t^{cmd} - v_t\|\|_2^2)$
$r_t^{\dot{q},reg}$	$0.05 \cdot exp(-0.01\|\|\dot{q}_t\|\|_2^2)$
$r_t^{\ddot{q},reg}$	$0.05 \cdot exp(-20.0\|\|\ddot{q}_t\|\|_2^2)$
$r_t^{F,reg}$	$0.1 \cdot exp(-0.0005(\|\|F_{r,t}\|\|_2 + \|\|F_{l,t}\|\|_2))$
$r_t^{\Delta F,reg}$	$0.1 \cdot exp(-0.0005(\|\|\Delta F_{r,t}\|\|_2 + \|\|\Delta F_{l,t}\|\|_2))$
$r_t^{\tau,reg}$	$0.05 \cdot exp(-0.01(\|\|\tau_{cmd,t}\|\|_2))$
$r_t^{\Delta\tau,reg}$	$0.2 \cdot exp(-0.01(\|\|\tau_{cmd,t} - \tau_{cmd,t-1}\|\|_2))$

E. Training Setup

a) Dynamics Randomization: To narrow the reality-gap and improve the robustness in the real-world implementation, the dynamics parameters are randomized during training, as shown in Table II. The default values of $m_{default}, I_{default}, p_{CoM,default}$ are acquired from the robot CAD model, while the default values of joint friction $v_{joint,default}$ and joint damping $f_{joint,default}$ follow the gear specification sheet. Since our real robot actually controls not a torque but a motor current, as opposed to the simulation, it is also important to randomize the motor constant. Additionally, the delay is randomized between 2ms and 6ms, considering the delay of mechanical response.

TABLE II: Dynamics Randomization Parameters

Parameter	Range	Unit
Link Mass	$[0.6, 1.4] \times m_{default}$	kg
Link Inertia	$[0.6, 1.4] \times I_{default}$	kg·m^2
Link Center of Mass	$[0.6, 1.4] \times p_{CoM,default}$	m
Joint Damping	$[0.6, 1.4] \times v_{joint,default}$	Nm·s/rad
Joint Friction	$[0.6, 1.4] \times f_{joint,default}$	Nm
Motor Constant	$[1.0, 1.1] \times c_{motor,default}$	-
Delay	$[0.5, 1.5] \times t_{delay,default}$	ms

b) RL Algorithm: As an RL algorithm, Proximal Policy Optimization (PPO) is used to train the policy. Both the actor and the critic are modeled as multi-layer perceptions with 2 hidden layers of 256 ReLU units. The agent is updated with a batch size of 128 after $16,384$ samples are collected, and $80,000,000$ samples are collected in total. The learning rate linearly decreases from $5e^{-5}$ to $1e^{-6}$ as the training progresses.

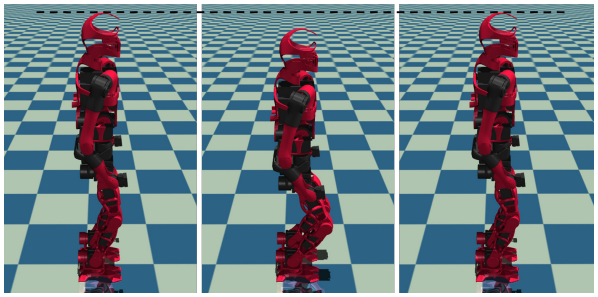

Fig. 2: Trained squat motion

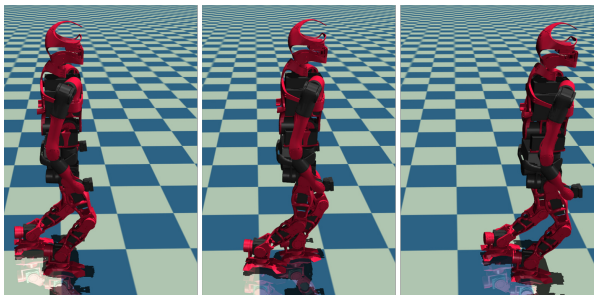

Fig. 3: Trained running motion

The maximum episode length is set to 16s, which is $8,000$ simulation steps, and the early termination occurs when the robot's links, other than both feet, contact the ground.

c) Tasks: In this paper, we selected three tasks (squat, walk, run) so that the difficulty level of each task differs. In the squat motion, the robot bends its knee for 2s, maintains its pose for 2s, and then stretches the knee for 2s without requiring foot contact switching, and the reference motion for the squat is handcrafted. For the walking task, the robot is commanded to follow the target velocity by switching the contact, and a single clip of walking motion with a 1.8s cycle is acquired from the model-based controller [23] for reference motion. For the running task, the robot follows a wider range of target velocity, and the motion capture data from DeepMimic [6] is used as a reference motion.

V. Result

We aim to answer four questions about torque-based policies: (1) Does pre-training with gravity compensation torque accelerate the learning of the torque-based policy (Sec. V-A)? (2) Can the torque-based policy be applied to a wider range of tasks and robots without additional tuning (Sec. V-B)? (3) Does the torque-based policy exhibit compliant behavior (Sec. V-C1)? and (4) Does the compliant behavior of the torque-based policy reduce the reality gap in sim-to-real transfer (Sec. V-C2)?

A. Effect of gravity pre-training

When using the torque-based deepRL policy, the sample efficiency in the early stages of learning is normally low. First, we verified that the pre-trained policy is capable of maintaining its pose on various configurations, indicating that the gravity torque is properly learned and compensated. We then train the torque-based deep RL policy on *squat* and *walking* tasks,

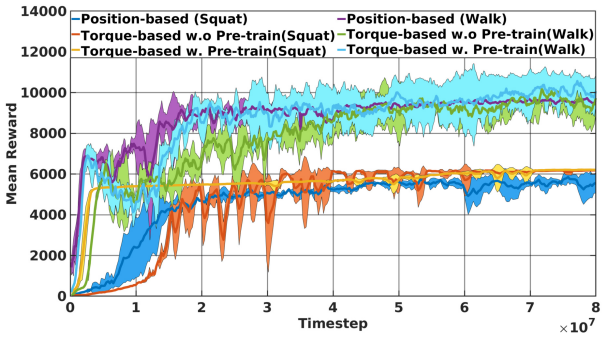

Fig. 4: Effect of gravity pre-training

using the pre-trained model as a starting point, to ensure that pre-training works regardless of whether the task requires foot contact transitions. The learning curves of the position-based and the torque-based deepRL policy with/without gravity pre-training are shown in Fig. 4 with mean and variance of three runs for each task and policy. The pre-trained torque-based policy learns at a comparable or faster speed than the position-based deepRL policy depending on the task because the pre-trained torque policy can stand still from the beginning of training, similar to the position-based deep RL policy. Also, as training progresses, the pre-trained torque policy converges faster than the torque policy trained from scratch, demonstrating that gravity pre-training accelerates the training procedure.

B. Torque-based RL for Task-and-Robot Agnostic Learning

For the position-based deepRL policy, PD gains should be tuned depending on the task and robot to achieve optimal policies, unlike the torque-based deepRL policy. In this subsection, we aim to verify this by showing that tuning gains that work across tasks (Experiment 1) and robot platforms (Experiment 2) is difficult for the position-based deepRL policy compared to the torque-based deepRL policy in simulation.

Experiment 1: To apply a control method on diverse tasks, the controller should have the potential to satisfy the required specifications (e.g. tracking performance, compliance) of multiple tasks. To understand the limitations of tuning position-controlled policies compared to a torque-based learning method, our first experiment compares position-based deepRL policies with PD gains to a torque-based deepRL policy across multiple tasks on a single robot. We attempted to train position-based deepRL policies across a set of tasks (squat, walk, run) using a *single* set of PD gain across all tasks. However, it was difficult to find a single set of gains that could be applied to all tasks simultaneously because each task required different levels of tracking performance and compliance. To train the position-based deepRL policy successfully, not only the PD gain but also the action standard deviation should be tuned for each task. Specifically, by examining various PD gains $K_p = K_{default}/s_p$, $s_p = \{1, 2, 4, 8\}$, and action standard deviations $\Sigma = q_{limit}/s_q$, where $s_q = \{100, 200, 400, 800\}$, the squat motion and walking motion could be learned by fixing the gain to $K_p = K_{default}/4, K_{default}/8$ and tuning the action standard deviation for each task ($s_q = 100$ for

squat, and $s_q = 100, 200, 400$ for walking). However, the running motion could not be learned at all with gains of $K_p = K_{default}/4, K_{default}/8$, and $K_p = K_{default}/2$ should be adopted. In contrast, the torque-based control method did not require further parameter tuning across tasks, and it was relatively robust to the action standard deviation by learning all tasks with $s_\tau = 5, 10, 20$. This is because the torque-based method can utilize a wider range of control space and has a higher potential to satisfy the requirements of multiple tasks since it is not pre-restricted and not inductively biased, unlike the position-based deepRL policy that uses the PD controller.

Experiment 2: In the second experiment, we investigate whether the same gains can be used as the robot platform changes and adopt Atlas and TOCABI as the comparison candidates. These robots are chosen to demonstrate that, although they are both human-sized robots with a mass of 90 kg and 104 kg, respectively, and have the same order of lower-body joint direction (YRPPPR), the gain is restricted to a specific robot. Specifically, when using the gains of TOCABI $K_P = K_{default}/4, K_{default}/8$ in the position-based deepRL policy, which is verified in Experiment 1, on Atlas for the walking task, the simulation became unstable and the training could not progress. This is due to the kinematics and dynamics differences between the two robots, making it extremely difficult to find a single set of gains that would fit both robots. Instead, we had to tune a separate gain for Atlas. In addition, the gain had to be tuned for each joint by allocating a larger gain for joints that would sustain gravity. In contrast, the same parameter (action standard deviation) used in TOCABI $s_\tau = 5, 10, 20$ could be simply transferred to Atlas, and no trial-and-error was required in the torque-based deepRL policy. Moreover, setting the action standard deviation Σ to a scaled value of the torque limit of each joint as proposed in Sec. IV-B2 was successful without exception.

C. Torque-based RL for Sim2Real Transfer

In this subsection, we evaluate the analysis of simulation and real robot hardware to determine the generality of torque-based control and how its built-in compliance can assist in Sim2Real transfer.

1) Compliance Analysis:

We start by demonstrating how the intrinsic compliance of position-based and torque-based deepRL policies differ by examining the reaction of both policies to unexpected contact (Experiment 3) and how the PD gain affects compliance in the position-based deepRL policy (Experiment 4) on the TOCABI robot while performing a walking task.

Experiment 3: In this experiment, we demonstrate how the position-based deepRL policy and torque-based deepRL policy respond to unexpected contact and analyze compliance behavior. To simulate the unexpected contact, the robot is trained on flat terrain, and after training is completed, an obstacle is placed in the environment in the simulation. The obstacle is modelled as a box with a one cm height lying on the ground 60 cm from the robot's starting position and results in early contact with the ground. In Fig. 5, the command torques of the ankle pitch joint of a foot stepping on the obstacle, which is the most influenced joint by perturbation, are plotted. In Fig. 5(a), when the robot steps on an unexpected obstacle at around 4s with position-based deepRL policy, the command torque abruptly increases compared to when the robot walks on flat terrain. This is because the low-level PD controller generates large torque when the tracking error increases due to early contact. The sudden impact is propagated from the foot to other links and the robot loses balance eventually. However, as seen in Fig. 5(b), the torque-based deepRL policy does not show any peak torque although the robot steps on an unexpected obstacle and smoothly passes the obstacle.

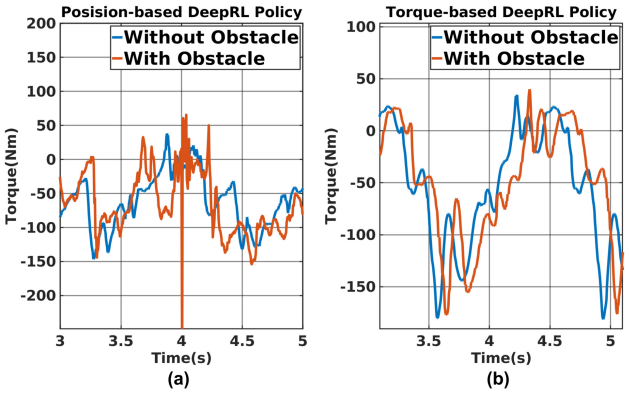

Fig. 5: Command torque with/without obstacle in simulation

We conducted the same experiment on the real robot and the time-lapse of both policies encountering an obstacle is shown in Fig. 6. By examining the contact force of both policies when stepping on the obstacle, the result showed that the position-based deepRL policy produced a larger impact force when stepping on an unexpected obstacle and fails to overcome the obstacle, eventually losing its balance as in the simulation. Conversely, the torque-based deepRL policy did not show any sudden torque-change when the robot encounters an unexpected obstacle. This compliant behavior is consistent with the simulation result and demonstrates that the torque-based deepRL policy shows greater compliance than the position-based deepRL policy.

Experiment 4: In the position-based deepRL policy, the low-level PD controller restricts compliance, and the level of compliance is predetermined by the PD gain. In this experiment, we examine how compliance in the position-based deepRL policy varies as we adjust the gain in the same simulation environment as Experiment 3. To achieve this, we evaluate the success rate as the target velocity changes when implementing position-based deep RL policies trained with various gains and the torque-based deep RL policy. Success is defined as not falling for 16 seconds, which is the maximum episode duration of the training environment. The target velocity increases by 0.02 m/s in each episode, with 25 trials ranging from 0.02 m/s to 0.5 m/s, which fall within the range of target velocities used during training.

The position-based deepRL policy with $K_p = K_{default}/8$ ($s_p = 8$), which was verified in Sec. V-B, resulted in the robot losing balance 10 times out of 25 trials. Then, we inspected

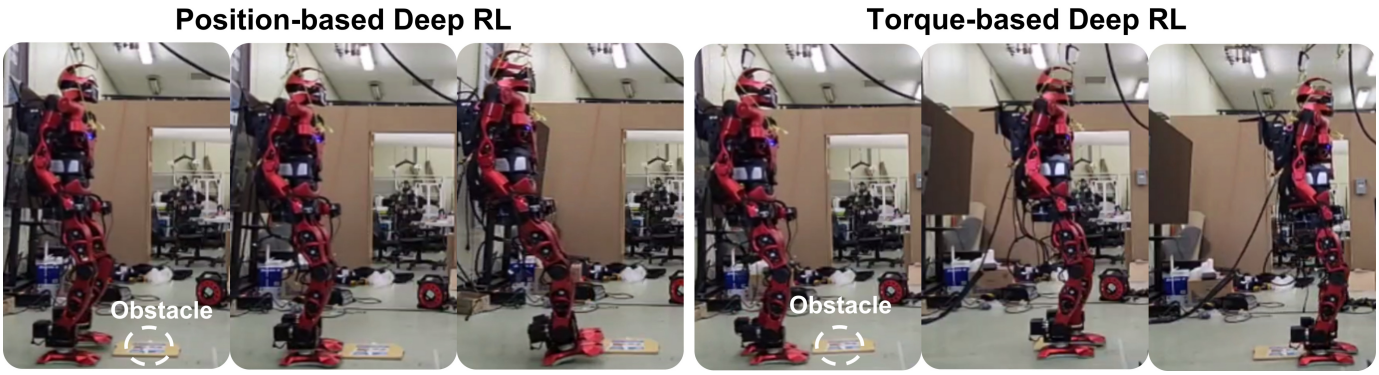

Fig. 6: Time-lapse of position-based deepRL policy and torque-based deepRL policy stepping on an unexpected obstacle

various PD gains of the position-based deepRL policy by continuously decreasing the P gain by half, starting from the proportional gain $K_p = K_{default}/8$. Each policy is trained from scratch with the corresponding gain, and the results in Table III indicate that the success rate increases as more compliance is introduced by using a low gain requiring trial and error to find a proper gain for the desired level of compliance. In contrast, the torque-based deepRL policy never fell and remained stable at all target velocities regardless of which action standard deviation s_τ was used.

Gain Scale Factor s_p	8	16	32	64	128
Success Rate	15/25	18/25	22/25	22/25	25/25

TABLE III: Success rate of position-based deepRL policy according to the gain

2) Benefits of Compliance for Overcoming Reality Gap: Since we have verified the inherent compliance of the torque-based deepRL policy in Sec. V-C1, we demonstrate its benefits when deploying the trained policy to environments other than those used for training. To this end, we show that the torque-based deepRL policy is advantageous for Sim2Real transfer by exploiting its intrinsic compliance (Experiment 5, 6).

Experiment 5: To examine how compliance contributes to robust sim-to-real transfer, we first *simulate* the reality gap by training both policies without dynamics randomization and then randomizing the parameters in simulation after training is completed. After training both policies with default parameters in Table. II, the parameters are randomly scaled between $[0.7, 1.3]$ from their default values. Additionally, a parameter that scales leg length is also randomized to effectively simulate early or late contact with the ground. At the start of each episode, the parameters are uniformly randomized, and the sampled parameters are logged with the episode reward and a success indicator at the end of the episode. The success indicator is set to *True* when the robot walks for the maximum episode length (16s), and the episode reward is a cumulative reward during an episode to measure motion quality.

Both policies encountered $4,870$ sets of randomized parameters, and the position-based deepRL policy could withstand $3,477$ sets of parameters without falling, while the torque-based deepRL policy exhibited greater robustness by successfully handling $3,604$ sets of parameters. When examining the episode rewards of the successful runs, as shown in Fig. 7, the torque-based deepRL policy exhibited a higher mean reward with less variance than the position-based deepRL policy. This indicates that the torque-based deepRL policy produces higher-quality motion by more accurately tracking the reference motion and generating smaller contact forces as a result of its compliance.

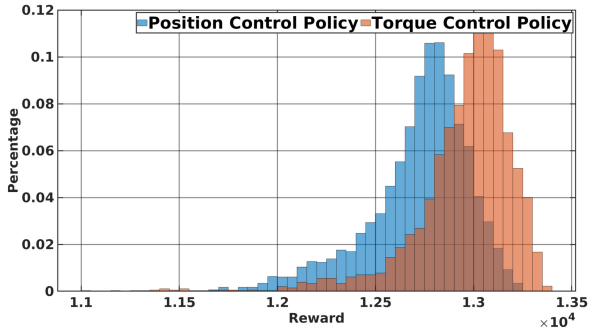

Fig. 7: Histogram of episode reward of alive runs

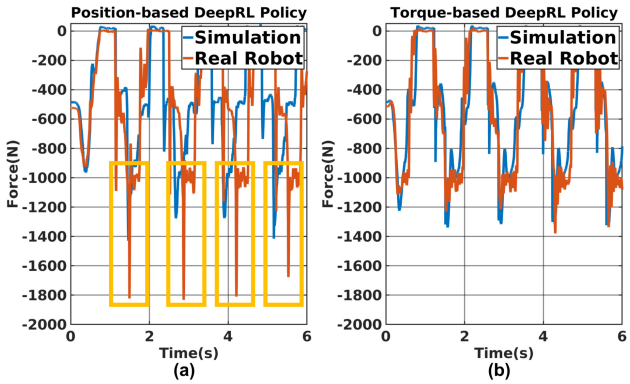

Fig. 8: Contact force comparison in simulation and real robot

Experiment 6: In this experiment, we demonstrate how compliance is helpful for Sim2Real transfer. We transfer the trained policy to the real robot and commanded it to walk on flat ground with a target velocity of 0.2m/s. Fig. 8 shows the contact force of both policies on the real robot along with the contact force in the simulation. In Fig. 8(a), the impact

force of the position-based deepRL policy on the real robot ($\approx$ 1800N) is much larger than in simulation (<1400N), while in Fig. 8(b), the impact force of the torque policy is very similar both on the real robot and in simulation due to the compliance of the torque policy. Although the robot could walk on flat terrain with both policies, the reality gap on the position-based deepRL policy continuously resulted in a stamping motion of the foot due to high impact force, and the motion was less stable. Also, the contact timing gradually mismatched with the simulation in the position-based policy as the walking progressed. Additionally, as demonstrated in Fig. 6, when a larger reality gap is introduced by an unexpected obstacle, the position-based deepRL policy could not withstand the reality gap. These comparisons indicate that torque-based policies can more smoothly handle differences between the simulation and the real world.

D. Torque-based DeepRL Policy Control Frequency

Lastly, we argue that the torque-based deepRL policy can be applied with various frequencies. In traditional humanoid torque control, it is common to use high control frequencies of up to 1-4 kHz. However, the torque-based deepRL policy was implemented on the real robot with control frequencies of 62.5 Hz, 125 Hz, and 250 Hz, and it was not difficult to reduce the control rate without episode reward drop. The main difference was that the training time was longer at lower control rates to collect the same amount of training samples.

VI. Conclusion

In this work, we investigated which action space is suitable not only for task-and-robot agnostic learning but also for reducing the reality gap on biped robots. By analyzing the proposed torque-based deepRL policy alongside the widely-used position-based deepRL policy, it is demonstrated that the torque-based deepRL policy can learn to squat, walk, and run with minimal tuning. Additionally, the torque-based deepRL policy did not require further parameter tuning when the robot platform changes from TOCABI to Atlas, making it suitable for task-and-robot agnostic learning. Furthermore, it is shown that the torque-based deepRL policy is inherently compliant, and this compliance is beneficial when the trained policy encounters an environment different from the one it was trained on by reducing the unexpected impact force. Lastly, we have accelerated the training of the torque-based deepRL policy by pre-training it with a gravity torque. This is the first successful attempt to implement a torque control method with deep RL on a human-sized humanoid, and we believe this could suggest a new way to actively take advantage of torque-based control methods in RL.

References

[1] J. Englsberger, G. Mesesan, A. Werner, and C. Ott, "Torque-based dynamic walking-a long way from simulation to experiment," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 440–447.

[2] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[3] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS international conference on humanoid robots*. IEEE, 2014, pp. 279–286.

[4] 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.

[5] G. Feng, H. Zhang, Z. Li, X. B. Peng, B. Basireddy, L. Yue, Z. Song, L. Yang, Y. Liu, K. Sreenath, *et al.*, "Genloco: Generalized locomotion controllers for quadrupedal robots," *arXiv preprint arXiv:2209.05309*, 2022.

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

[7] 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.

[8] X. B. Peng, G. Berseth, and M. Van de Panne, "Dynamic terrain traversal skills using reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, pp. 1–11, 2015.

[9] D. Rodriguez and S. Behnke, "Deepwalk: Omnidirectional bipedal gait by deep reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 3033–3039.

[10] 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.

[11] A. Calanca, R. Muradore, and P. Fiorini, "A review of algorithms for compliant control of stiff and fixed-compliance robots," *IEEE/ASME transactions on mechatronics*, vol. 21, no. 2, pp. 613–624, 2015.

[12] X. B. Peng and M. van de Panne, "Learning locomotion skills using deeprl: Does the choice of action space matter?" in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2017, pp. 1–13.

[13] H. Duan, J. Dao, K. Green, T. Apgar, A. Fern, and J. Hurst, "Learning task space actions for bipedal locomotion," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1276–1282.

[14] R. P. Singh, M. Benallegue, M. Morisawa, R. Cisneros, and F. Kanehiro, "Learning bipedal walking on planned footsteps for humanoid robots," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*. IEEE, 2022, pp. 686–693.

[15] S. Chen, B. Zhang, M. W. Mueller, A. Rai, and K. Sreenath, "Learning torque control for quadrupedal locomotion," *arXiv preprint arXiv:2203.05194*, 2022.

[16] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

[17] 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.

[18] 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.

[19] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *arXiv preprint arXiv:2107.04034*, 2021.

[20] 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.

[21] M. Schwartz, J. Sim, J. Ahn, S. Hwang, Y. Lee, and J. Park, "Design of the humanoid robot tocabi," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*. IEEE, 2022, pp. 322–329.

[22] J. Park and O. Khatib, "Contact consistent control framework for humanoid robots," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 1963–1969.

[23] M.-J. Kim, D. Lim, G. Park, and J. Park, "Humanoid balance control using centroidal angular momentum based on hierarchical quadratic programming," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 6753–6760.