Journal of Field Robotics

WILEY

RESEARCH ARTICLE

Learning Accurate and Robust Velocity Tracking for Quadrupedal Robots

Chengrui Zhu[1] | Zhen Zhang[1] | Weiwei Liu[2] | Siqi Li[1] | Yong Liu[3]

[1]Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, China | [2]Huzhou Institute, Zhejiang University, Huzhou, China | [3]State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou, China

Correspondence: Yong Liu (yongliu@iipc.zju.edu.cn)

Received: 30 November 2024 | **Revised:** 10 March 2025 | **Accepted:** 7 July 2025

ABSTRACT

Quadrupedal robots are highly regarded for their superior locomotion capabilities and terrain adaptability, making them competent in a wide range of applications. For autonomous navigation, they must track upper-level trajectories to reach designated locations with flexible obstacle avoidance. This is typically achieved by a planner, which generates a reference velocity, and a controller, which accurately tracks the velocity commands. This article proposes a learning-based controller for quadrupedal robots that is trained in simulation and achieves accurate and robust velocity tracking in the real world. To bridge the gap betwen simulation and reality, an analytical actuator model is introduced to simulation to simulate physical actuator dynamics. We then train a control policy in simulation using Constrained Reinforcement Learning, where symmetry and smoothness constraints are incorporated into Reinforcement Learning. The symmetry constraint promotes coordinated locomotion and consistent velocity tracking performance, while the smoothness constraint reduces jerky actions and generates stable velocity performance. The proposed control policy is zero-shot deployed on the Unitree AlienGo. Experimental results demonstrate a velocity tracking error below 0.084 m/s across the entire operational velocity range while maintaining robust locomotion on natural terrains. To further validate the controller's effectiveness, we integrate it into a pedestrian tracking framework, where it demonstrates precise trajectory following capabilities and long-term reliability.

1 | Introduction

Legged robots are considered to have promising applications due to their high locomotion dexterity. They have the potential to traverse the terrains that humans can traverse and perform the tasks that humans can perform. In particular, quadrupedal robots can replace humans in dangerous and harsh environments, such as searching for survivors in ruins or fires, or patrolling and detecting in factories and electrical substations. Complex operating environments pose a great challenge to quadrupedal controllers. In practice, the robot is often instructed to follow a predefined trajectory, traverse various terrains, avoid collision with obstacles, and reach a designated location. This is typically achieved by a planner specifying the desired velocity of the robot (Shin and Kwak 2017; Zhang et al. 2021), and the controller is responsible for tracking the given velocity accurately and robustly.

The development of quadrupedal controllers in recent years is bringing those visions to reality. Quadrupedal robots can perform agile and robust locomotion in many challenging scenarios. Classical model-based approaches (Di Carlo et al. 2018; Kim et al. 2019; Winkler et al. 2018; Iqbal et al. 2020; Lu et al. 2023) perform online optimization using a dynamics model and exhibit agile behaviors such as balancing, walking, running, and jumping on quadrupedal robots. Those controllers are usually elaborately designed and difficult to automatically adapt to different scenarios. To overcome these challenges, reinforcement learning (RL) offers a promising approach to legged locomotion control. RL algorithms automatically learn a policy through trial and error and require less human intervention. RL-based approaches can outperform the model-based ones in many scenarios. Hwangbo et al. (2019) first demonstrated agile and robust locomotion on a physical robot

with trained controllers. Margolis et al. (2022) and Jin et al. (2022) achieve extremely rapid locomotion with a speed of up to 5.0 m/s. With the inclusion of diverse terrain in the training process, RL policies also learn to traverse challenging terrains such as rough, grassy, or sandy ground with proprioception (Lee et al. 2020; Choi et al. 2023; Aswin Nahrendra et al. 2023) and also environmental perception (Miki et al. 2022; Agarwal et al. 2023).

On the other hand, accurate velocity tracking is a general problem for quadrupedal controllers. First, the robot's dynamics, including friction, payloads, actuator states, etc., are highly uncertain, requiring the controller to estimate and adapt accordingly. For RL controllers trained in simulation, the discrepancy between the simulation and reality domains exacerbates the problem. Techniques such as dynamics randomization (Tobin et al. 2017; Peng et al. 2018a) have been widely used to adapt to uncertain dynamics, but they are still insufficient to bridge the sim-to-real gap. Second, the exploration and learning process of the RL algorithm has some drawbacks. It independently explores each degree of freedom without sharing experience among the limbs, which can result in unbalanced payload distribution, lameness (Yu et al. 2018), and inconsistent tracking accuracy. Also, the unstructured exploration of most RL algorithms leads to a preference for jerky actions (Raffin et al. 2022). All of these issues will eventually affect the tracking performance during deployment. Third, legged robots must exhibit velocity-appropriate gait patterns for accurate and efficient velocity tracking. Natural quadrupeds exhibit distinctive gaits for locomotion at different speeds (Hoyt and Taylor 1981), and quadrupedal robots should also exhibit similar gait diversity for elegant locomotion and low energy consumption.

In this study, we present a learning-based locomotion controller for quadrupedal robots that overcomes the above-mentioned limitations and enables accurate and robust velocity tracking. We first model and identify the robot's actuators using our proposed empirical actuator model (EAM). It contains the principal characteristics of a physical actuator and replicates its response in simulation. It bridges the sim-to-real gap and maximizes the reproduction of the controller's simulation performance in physical robots. Then, a controller network is trained in simulation with our proposed Constrained RL, where two constraint functions, symmetry and smoothness constraints, are incorporated into RL to facilitate consistent, robust, and smooth velocity tracking.

The efficacy of our method is demonstrated using the Unitree AlienGo, a medium-dog-sized quadruped robot. The robot exhibits omnidirectionally accurate velocity tracking over a wide range of commands. It can automatically adjust its gait to velocity changes, making it competent at both low and high velocities. Additionally, it achieves the highest reported running velocity for this type of robot while maintaining high energy efficiency and robustness on natural terrain. Our method establishes a foundation for precise trajectory tracking and upper-level task execution such as human-following, as shown in Figure 1.

Our main contributions are concluded as follows:

- An effective actuator model is proposed to simulate the actuator's response and mitigate the sim-to-real gap.
- A Constrained RL framework is proposed for quadrupedal robots to facilitate accurate and smooth velocity tracking.
- Extensive testing of tracking performance for omnidirectional velocities is conducted. To the best of our knowledge, this is the highest velocity tracking performance achieved by quadrupedal robots.

FIGURE 1 | Our proposed controller completes an automatic human-following tour in Zhejiang University assisted by a planner (Zhang et al. 2021) specifying its velocity. The robot finished the 3.1-km-tour within 37 min without human intervention, during which it encountered terrains such as slopes, vegetation, and gravel. Our controller demonstrates omnidirectionally accurate velocity tracking and long-term robustness.

2 | Related Work

2.1 | Deep Reinforcement Learning (DRL)-Based Quadruped Locomotion

Since DRL techniques have been proposed, there has been an amount of work by researchers on locomotion control in quadrupeds. First in the field of animation and simulation, Peng et al. (2015, 2016) employed a RL method to control a 21-link planar dog to navigate complex 1D terrains. Their methods work well with continuous and high-dimensional states, and show how RL methods can be used for such problems.

Tan et al. (2018) deployed DRL technology for the first time on a home-built 8-degree-of-freedom quadruped robot named Minitaurs, which learned two agile locomotion gaits of trotting and galloping. Hwangbo et al. (2019) then successfully deployed the locomotion policy learned in the simulation on a large 12-degree-of-freedom quadruped robot, achieving locomotion performance that exceeded existing model-based controllers. To enhance the sample efficiency of RL, Haarnoja et al. (2018) introduced the soft actor-critic DRL algorithm to achieve training of locomotion gait within 2 h. Ha et al. (2020) strive to reduce human intervention in the training process of RL-based locomotion policy and achieve stable and efficient learning.

2.2 | Sim-to-Real Transfer

A major problem of control policies learned in simulation to deploy to a real robot is to overcome the sim-to-real discrepancy. Typically, they are bound to underperform or even fail to control without careful design of the simulation environment or the policy. This is because there is a gap between the physical properties of the real world and those in the simulation, or rather, the simulator is not able to fully simulate the state of the robot in the real world. Neunert et al. (2017) note that most robot simulators do not model the dynamics, noise, and latency associated with jointed actuators. They believe that ignoring these factors is one of the reasons why controllers do not transfer well from simulation to hardware. Li et al. (2013) applied carefully measured physical parameters to the simulation to realize the deployment of open-loop locomotion control. In addition, physical simulation parameters can be obtained by employing system identification. In Zhu et al. (2018), Lee and Park (2018), Tan et al. (2016), Moeckel et al. (2013), and Yu et al. (2017), the fidelity of the simulator is complemented by system identification. Peng et al. (2018b) implicitly embedded system identification function in the policy learning process.

Another effective technique for enhancing the sim-to-real transfer of control policies is simulation randomization. It makes the controller robust to changes in system properties, leading to better adaptability. Pinto et al. (2017) trained a robust control policy by learning a destabilization policy. Tan et al. (2018), Peng et al. (2018b), Jakobi et al. (1995), and Mordatch et al. (2015) developed policies capable of adapting to different dynamics by randomizing the dynamics of the simulator during the training as well as adding random noises to observation.

3 | Methodology

This section describes the proposed RL-based locomotion framework that enables quadruped robots to track a velocity command with high accuracy and efficiency. Our framework concentrates on two critical problems: maximizing the policy performance at training and minimizing the performance degradation during the sim-to-real transfer. An overview of our method is illustrated in Figure 2.

3.1 | Actuator Modeling

One of the most effective ways to bridge the sim-to-real gap is to replicate the underlying physical process as realistically as possible in the simulation. The robot's links, joints, and inertia can be modeled as a multi-body system and easily solved by modern simulators. It takes the generalized force and outputs the generalized velocity and position. However, the actuator, which translates the joint commands into the generalized force, has different mechanisms and structures for different robots and is difficult to model. Actuator dynamics include communication delays, low-level controller response dynamics, motor peak performance, and so forth. These characteristics profoundly affect the robot dynamics and should be included in the simulation so that the controller can adapt accordingly to maximize its performance on the robot. The most common approach is to use a proportional-derivative (PD) controller with drastically randomized gains and system delay to simulate the actuator's behavior (Aswin Nahrendra et al. 2023; Wu et al. 2023; Kumar et al. 2021; Feng et al. 2023). It can help with a successful sim-to-real transfer, but it is not sufficient to tailor the controller to a specific robot.

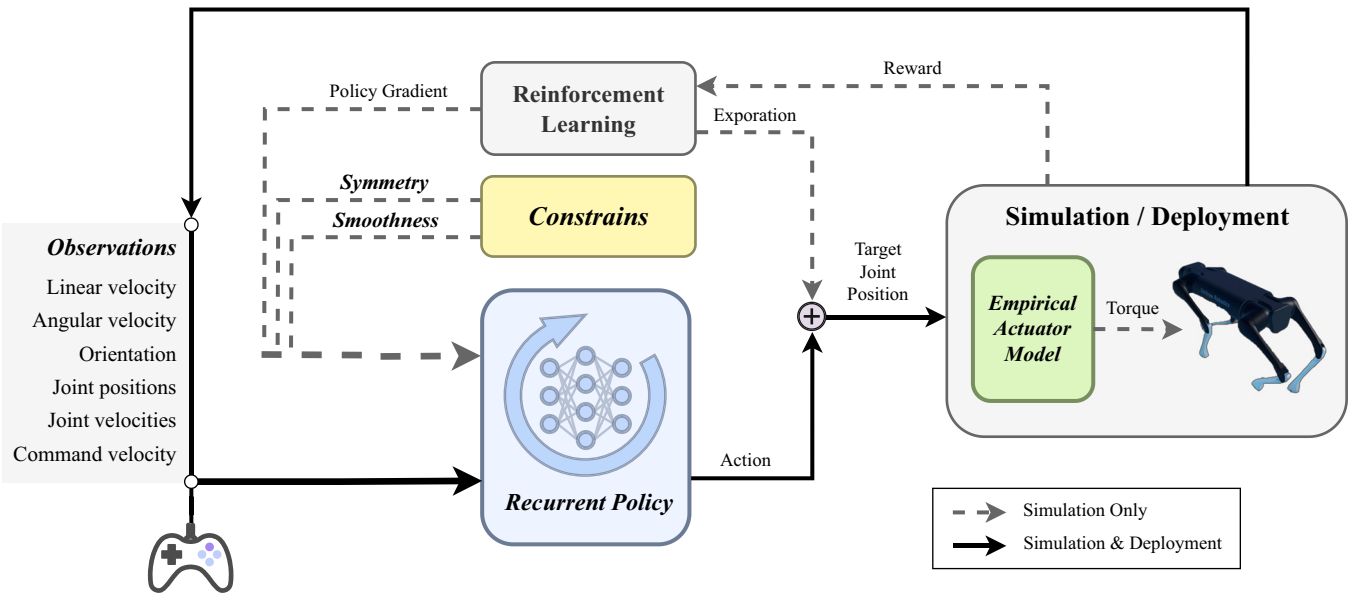

FIGURE 2 | An overview diagram of our proposed framework is shown. First, an empirical actuator model is identified from data collected from the robot and converts the actuator commands into joint torques in simulation. Then, a policy is trained using reinforcement learning with symmetry and smoothness constraints. It accepts a velocity command, orientation, joint states, and velocities and outputs target joint positions. At deployment, the observations are collected from onboard sensors, and the target joint positions are directly sent to the actuators.

To finely model an actuator, we propose an analytical and parameterized actuator model called the EAM. EAM considers the physical processes of how the actuator works, thus improving simulation accuracy. It is expressed as,

$$\hat{\tau}_{t+t_o} \sim \mathcal{N}\left(\hat{K}_p\left(\hat{q}_{t-t_i} - q_t\right) - \hat{K}_d \dot{q}_t, \delta^2\right), \tag{1}$$

$$\hat{K}_p = K_p + \Delta K_p, \hat{K}_d = K_d + \Delta K_d, \tag{2}$$

where q, $\dot{q}$, and τ are the position, velocity, and torque of the actuator, $\hat{q}$ and $\hat{\tau}_t$ are the target position and torque, t_i and t_o are the input and output delays, K_p and K_d are the command proportional and derivative gains, ΔK_p and ΔK_d are the deviation of the gains, and δ is the torque execution error, respectively. The input delay t_i is the transmission time of commands through several hardware and software layers. The target torque follows the form of PD control, with its gains varying with the actuator state. Since the torque execution typically has a short transient process, it is simplified to a normal distribution with a variance δ^2 and an output delay t_o.

The parameters of EAM can be easily identified from just seconds of locomotion segments collected from the robot. For $t_i \in [0, t_{i,max}]$, $t_o \in [0, t_{o,max}]$, we simulate each $<t_i, t_o>$ pair and compute ΔK_p and ΔK_d by minimizing the mean squared error (MSE) between the measured torque and the predicted torque using linear regression. The optimal $<t_i, t_o>$ pair is the one with the minimum MSE $\underline{e}$, and δ is set to $\sqrt{\underline{e}}$. To augment the diversity of the simulation, all pairs with MSE less than $1.5\underline{e}$ are considered probable and sampled in the simulation, and the PD gains are randomized to $[K_p + 2\Delta K_p, K_p]$ and $[K_d + 2\Delta K_d, K_d]$ (assuming ΔK_p and ΔK_d are negative).

3.2 | Problem Definition

Quadrupedal locomotion can be formalized as a partially observable Markov decision process (POMDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, \Omega, \mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{S}, \mathcal{A}, \Omega$ are the state, action, and observation spaces respectively, $\mathcal{O}: \mathcal{S} \times \mathcal{A} \times \Omega \to [0, 1]$ is the emission function, $\mathcal{T}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function, and $\mathcal{R}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function. At time t, the policy emits an action $a_t \in \mathcal{A}$ to the environment according to the historical observations $o_{0:t}$, the environment updates the state by the transition function $s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t)$, the reward is computed as $r_{t+1} = \mathcal{R}(s_t, a_t, s_{t+1})$, and the next observation is sampled as $o_{t+1} \sim \mathcal{O}(\cdot|s_{t+1}, a_t)$. Given a discount factor $\gamma \in [0, 1)$, the objective of POMDP is to find a parameterized policy $\pi_\theta(a_t|o_{0:t})$ that maximizes the expected long-term reward.

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1}\right]. \tag{3}$$

Real quadrupedal systems have many unobservable states, such as accurate robot dynamics and comprehensive terrain dynamics. Latencies in real quadrupedal systems also introduce historical states and actions to the state space. Therefore, the policy should consider past interactions in addition to current observations.

The action is defined as,

$$a_t \in \mathcal{A} = (\hat{q}) \in \mathbb{R}^{12}, \tag{4}$$

where $\hat{q}$ is the target joint positions. It is converted into joint torques by EAM in the simulation or sent to the robot at deployment. The observation is defined as

$$o_t \in \mathcal{O} = (c, \psi, q, \dot{q}, v, \omega, a_{t-1}) \in \mathbb{R}^{48}, \tag{5}$$

where c is the command, $\psi \in \mathbb{R}^2$ is the body orientation including roll ψ_x and pitch ψ_y, $q, \dot{q} \in \mathbb{R}^{12}$ are the joint positions and velocities, $v, \omega \in \mathbb{R}^3$ are the body linear and angular velocities, and $a_{t-1} \in \mathbb{R}^{12}$ is the previous action. The command $c = (c_x, c_y, c_r, c_s) \in \mathbb{R}^3 \times \{0, 1\}$, consists of a longitudinal velocity c_x, a lateral velocity c_y, a rotation velocity c_r, and a stance command c_s. The stance command determines whether the robot should locomote at the given velocities ($c_s = 0$) or stand still without cyclic leg movements ($c_s = 1$).

3.3 | Constrained Reinforcement Learning (CRL)

Our training algorithm is based on the recurrent version of proximal policy optimization (PPO) (Schulman et al. 2017), an actor-critic-based RL algorithm. The actor is designed as a long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) to preserve the encodings of historical interactions. It can infer some physical properties from the encodings that cannot be directly observed, such as the shape and friction coefficients of the terrain. The critic is similarly designed to learn the return of reward function.

Despite the success that RL has achieved, its limitations are also frequently reported. The vast exploration space and inappropriate reward function can cause many local optima, thus producing nonideal results. The policy suffers from learning plateaus because the data for policy learning is directly collected by the policy itself (Schaul et al. 2019). If some constraints based on prior knowledge are introduced into the learning process, the policy can get out of learning plateaus and converge to a better solution.

In this study, we incorporate two constraints into RL. First, as RL independently explores each action dimension, it does not guarantee consistency or coordination across limbs. Also, biologically similar behaviors, such as walking left and right or turning left and right, do not form associations in RL. As a result, RL policies often exhibit uncoordinated locomotion and inconsistent velocity tracking. As Figure 3 illustrates, since the morphology of a typical quadruped robot is approximately symmetric, an ideal policy should generate symmetric actions given symmetric observation sequences. Inspired by Yu et al. (2018), we introduce a ***symmetry constraint*** to encourage coordinated and consistent locomotion, which is defined as

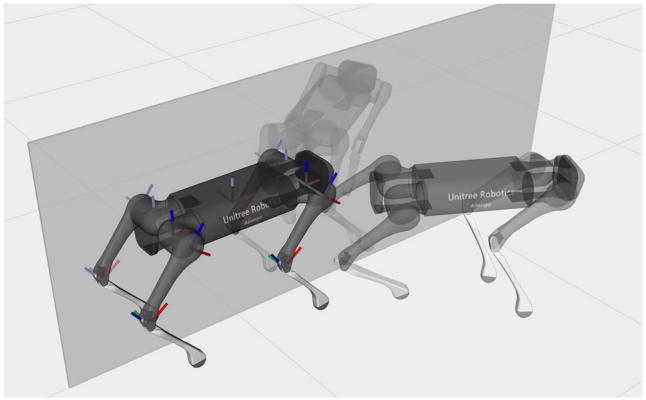

FIGURE 3 | Symmetric locomotion. An ideal policy should produce symmetric action sequences given symmetric observation sequences. The robot is symmetrically constructed along its mid-sagittal plane, and the orientation of all joint coordinate systems at zero position is aligned with the robot's base.

TABLE 1 | Symmetry mapping of observation and action.

	Expression	Symmetry mapping
Observation		
Orientation	ψ_x, ψ_y	$-\psi_x, \psi_y$
Linear velocity	v_x, v_y, v_z	$v_x, -v_y, v_z$
Angular velocity	$\omega_x, \omega_y, \omega_z$	$-\omega_x, \omega_y, -\omega_z$
Joint positions of leg i	q_1^i, q_2^i, q_3^i	$-q_1^j, q_2^j, q_3^j$
Joint velocities of leg i	$\dot{q}_1^i, \dot{q}_2^i, \dot{q}_3^i$	$-\dot{q}_1^j, \dot{q}_2^j, \dot{q}_3^j$
Action		
Target joint positions of leg i	$q_{t1}^i, q_{t2}^i, q_{t3}^i$	$-q_{t1}^j, q_{t2}^j, q_{t3}^j$

$$L_{\text{sym}} = \sum_t \text{KL}(\pi_\theta(o_{0:t}), \mathcal{M}_a(\pi_\theta(\mathcal{M}_o(o_{0:t})))), \tag{6}$$

where KL is the Kullback–Leibler divergence, $\mathcal{M}_o$, $\mathcal{M}_a$ denote the symmetry mappings for observations, actions. $\mathcal{M}_o$ and $\mathcal{M}_a$ are defined in Table 1, where leg j is the opposite of leg i, and the order of leg-wise joints is shoulder, thigh, and calf. The symmetry constraint shares the experience of symmetric limbs and conditions to generate aesthetically pleasing behavior, balance the payload between legs, and improve the consistency of velocity tracking in symmetric directions.

Another drawback of RL policies is their preference for jerky actions, caused by the unstructured exploration of most RL algorithms (Raffin et al. 2022). A jerky policy may be acceptable in simulation but can lead to shaky behavior and mechanical failure once deployed on physical robots. Jerky actions also increase the sim-to-real gap, as physical actuators typically cannot respond to jerky targets ideally. Although an action smoothness term can be added to the reward function (Ji et al. 2022; Lee et al. 2020; Choi et al. 2023), it is less effective than expected for the RL algorithms that optimize a stochastic policy rather than a deterministic policy. To reduce jerky actions, a ***smoothness constraint*** is introduced to directly constrain the deterministic policy, which is defined as

$$L_{\text{smooth}} = \sum_t \left\| \frac{d^2}{dt^2} \pi_\theta(o_{0:t}) \right\|. \tag{7}$$

The smoothness constraint reduces the high-frequency component of the action and helps with stable and smooth velocity tracking. It behaves like a low-pass filter during training, and no additional action filtering is required at deployment time (Peng et al. 2020). This maintains consistency between simulation and deployment and makes the policy perform exactly as it is.

Incorporating PPO's surrogate loss L_{surr} and the two constraints, Constrained RL optimizes the policy with the objective defined as follows:

$$\pi_\theta = \arg\min_\theta L_{\text{surr}} + w_{\text{sym}} L_{\text{sym}} + w_{\text{smooth}} L_{\text{smooth}}, \tag{8}$$

where w_{sym} and w_{smooth} are the weight coefficients of the symmetry and smoothness constraints, respectively.

3.4 | Environment Setup

The policy is trained in a highly randomized simulation environment, which can significantly boost the generalization and robustness of the policy (Peng et al. 2018a). The environments randomize the robot dynamics to simulate variable payloads. Observation noise and transmission delays are also randomized in the simulation. The environment consists of two terrains: flat ground and rough terrain. The latter is generated using Perlin noise with a random slope angle. The friction coefficient between each foot and the ground is individually randomized to [0.3, 1.0] to simulate potential foot slippage. In addition, random force and torque disturbances are imposed on the robot. The objective of the policy is to adapt to all these situations and perform accurate velocity tracking.

The velocity command is uniformly sampled from an ellipsoid, where $c_x \in [-4.5, 4.5]$ m/s, $c_y \in [-2, 2]$ m/s, $c_r \in [-4, 4]$ rad/s. There is a 20% chance to set $c_x = c_y = c_z = 0$ and $c_s = 1$, which allows the policy to learn the capability of standing still. The velocity command is resampled every Uniform (2, 10) seconds to learn the transition of different commands. Random force and torque disturbances are imposed on the robot to improve the robustness of the policy. The maximum duration of an episode is 20 s. If an episode ends due to timeout, there is a 25% chance to initialize the next episode with the final state to simulate long-term locomotion scenarios.

The policy learns by maximizing the weighted sum of the reward functions listed in Table 2. Among them, the linear and angular tracking items dominate, driving the robot to move at given velocities. Reward items from *orientation* to *slippage*

TABLE 2 | Reward functions.

Reward	Expression	Coefficient
Linear tracking	$\exp(-(c_x - v_x)^2 + (c_y - v_y)^2)$	0.5
Angular tracking	$\exp(-\lvert c_r - \omega_z \rvert^2)$	0.2
Orientation	$-\lvert \psi_x \rvert - \lvert \psi_y \rvert$	0.1
Torso stability	$-\sqrt{(\omega_x^2 + \omega_y^2)} - \lvert v_z \rvert$	0.1
Joint velocity	$-\sum_i \lVert \dot{q}_i \rVert^2$	1.0e−4
Joint acceleration	$-\sum_i \lVert \ddot{q}_i \rVert^2$	1.5e−8
Slippage	$-\sum_i \lVert v_{\text{slip},i} \rVert^2$	0.1
Energy	$-\sum_i \max(k_r \tau_i^2 + \tau_i \dot{q}_i, 0)$	3e−4
Velocity compensation	$\sqrt{\min(v_x, c_x)^2 + \min(v_y, c_y)^2}$	0.3
Alive	1	0.8
Below rewards are valid when $c_s = 0$		
Foot lift time	$\sum_i \min(T_{\text{lift},i} - 0.2, 0.5)^* c_{\text{decay}}$	1.0
Foot clearance	$\sum_i \min(h_{cl,i}, 0.05)^* c_{\text{decay}}$	3.0
$c_{\text{decay}} = \max(1 - \lVert [c_x, c_y] \rVert^2, 0)$		

encourage the policy to improve locomotion stability, reduce energy consumption, and suppress slippage, where $v_{\text{slip},i}$ is the slippage velocity of foot i, and $k_r = 0.2$ is a constant related to motor heat dissipation. The *velocity compensation* item is designed to compensate for the inevitable increase in energy consumption when moving at high velocities. The *foot lift time* and *foot clearance* rewards encourage the policy to form cyclic leg movements at low velocities, where $T_{\text{lift},i}$ and $h_{\text{cl},i}$ are the lift time and the clearance of foot i.

4 | Experiments

In this section, we experimentally demonstrate the superiority of our framework in terms of accurate and efficient velocity tracking.

4.1 | Experiment Setups

1. *Training Setup*: The policy is defined as a sequence of one layer of LSTM with 256 hidden units and two layers of multi-layer perception with 192×128 neurons. The critic has the same structure as the policy with the shape of $[128 \times 128 \times 128]$. Both are activated with sigmoid linear unit (Elfwing et al. 2018). The environment is implemented with RaiSim (Hwangbo et al. 2018) with a simulation timestep of 1 ms. The policy is trained in 1024 parallel environments for approximately 3 h on an AMD Ryzen 9 7950X and an NVIDIA RTX 3090. The hyperparameters are listed in Table 3.

TABLE 3 | Hyperparameters.

Hyperparameter	Value
Learning rate	2e−4
Discount factor	0.99
Entropy coefficient	0.01
PPO clip ratio	0.2
Number of mini-batches	8
Number of rollouts	4
Step per update	24
Iterations	20,000

TABLE 4 | The EAM parameters of AlienGo actuators.

ΔK_p	ΔK_d	t_i	t_o	δ^2
−0.706	−0.021	[6, 8] ms	[0, 2] ms	$0.066N^2 \cdot m^2$

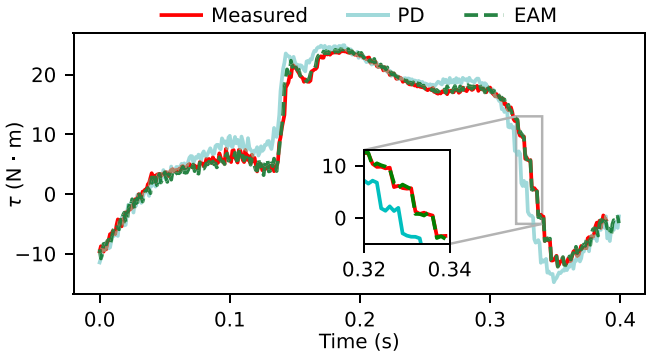

FIGURE 4 | The measured torque and the predicted torque of different models are shown. It is collected from the calf joint of the forward right leg in a state of walking forwardly at 1.7 m/s. The torque prediction of the ideal PD control deviates significantly from the measured values, while EAM precisely captures the dynamics of the actuator.

2. *Real-World Deployment*: We validate our method on a Unitree AlienGo robot. The policy is inferred on a mini PC, with a total inference time of less than 0.2 ms. All observations except the linear velocity of the policy come from the onboard inertial measurement unit and joint encoders. Inspired by Shao et al. (2022), the linear velocity is predicted by a learning-based state estimator. The policy is queried at 200 Hz, and the PD control of the underlying AlienGo actuators runs at 20 kHz with gains $K_p = 30$, $K_d = 0.5$.

4.2 | Actuator Identification

We collected a 5-s trajectory from the AlienGo actuators for identification. It is generated with a foot-lifting trajectory generator defined as

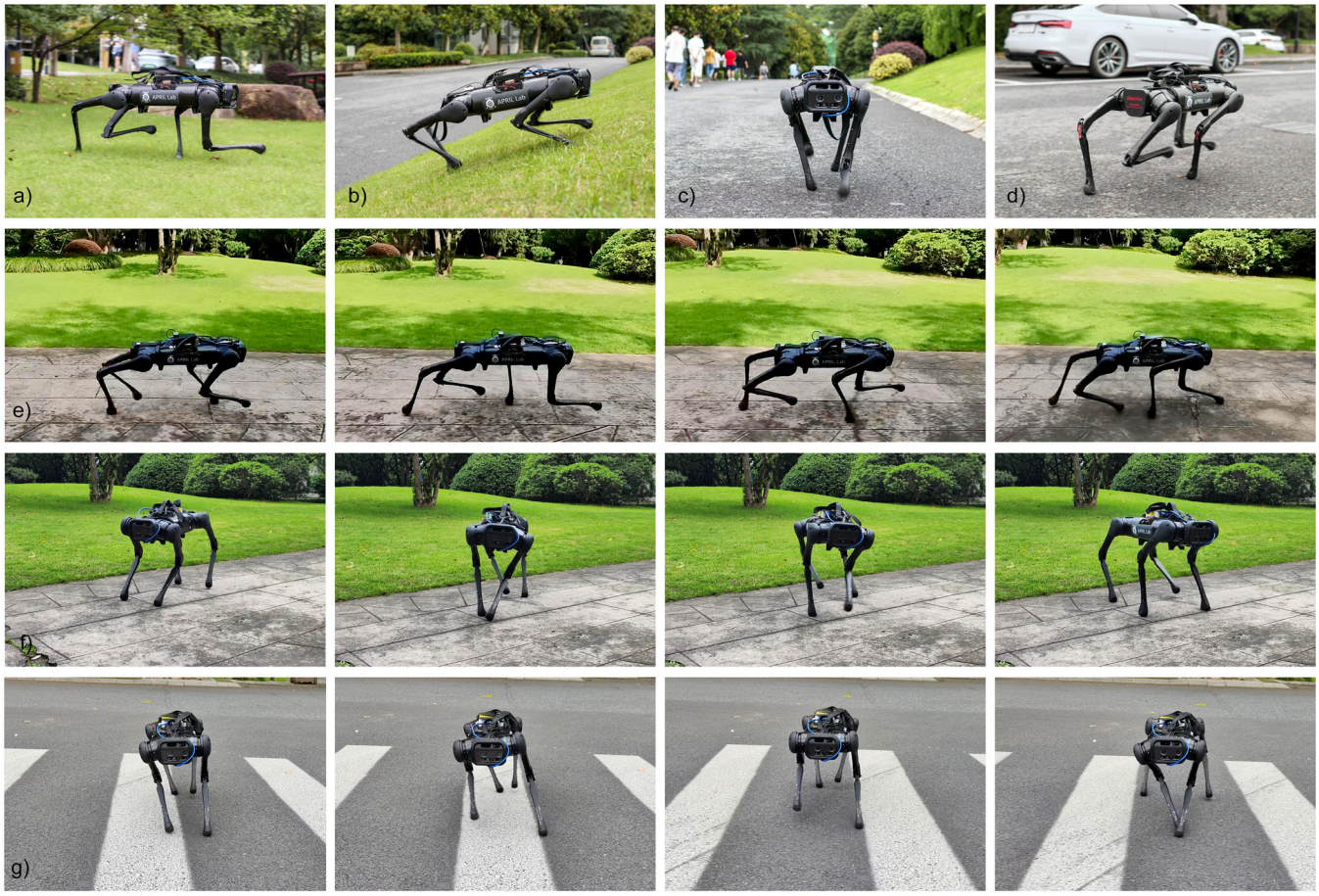

FIGURE 5 | Some captures of locomotion are shown. (a) The robot runs on lawns at a velocity of 3.5 m/s. (b) The robot walks up a 15° slope. (c) The policy tends to perform a catwalk with the feet positioned closer to the body center, which helps conserve energy by reducing gravity on the shoulder joint. (d) While the robot is running and rotating simultaneously, the legs on the inner side shift footholds towards the outer side to generate centripetal force. (e–g) Snapshots of the robot running rapidly, spinning rapidly, and walking laterally. In these scenarios, our controller demonstrates impressive robustness and agility.

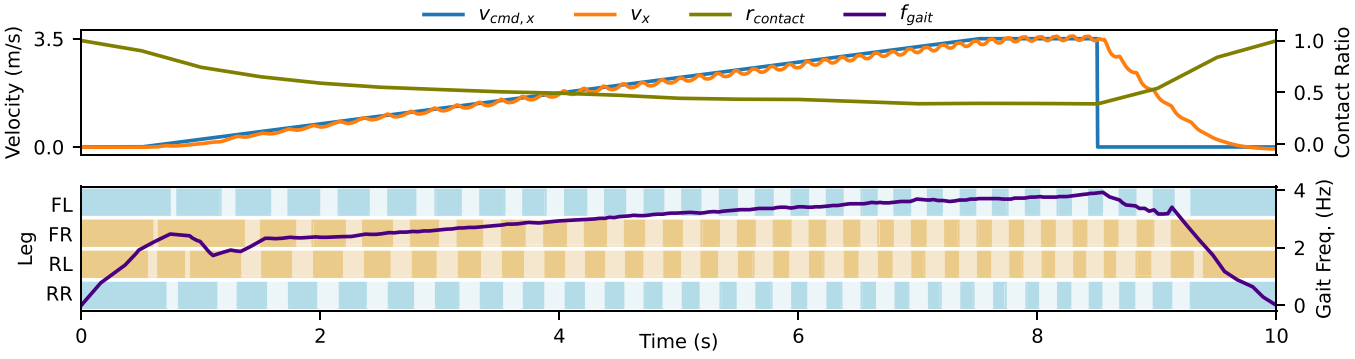

FIGURE 6 | Automatic gait transition during an acceleration process is illustrated, where v_x is the actual velocity along the x axis of the robot, r_{contact} is the contact ratio, the proportion of contact time throughout the gait cycle, and f_{gait} is the gait frequency. FL, FR, RL, and RR are the front/rear and left/right legs, respectively. The darker color represents the foot contact phase. Our controller exhibits a group of trot-like gaits but varies the gait frequency and contact ratio. As the velocity increases from 0 to 3.5 m/s, the gait frequency increases from 2.0 to 4.0 Hz, and the contact ratio decreases from 1.0 to 0.39.

$$h_i = \begin{cases} 0.1\left(-2k_i^3 + 3k_i^2\right) - 0.4 & k_i \in [0, 1) \\ 0.1\left(2k_i^3 - 9k_i^2 + 12k_i - 4\right) - 0.4 & k_i \in [1, 2) \\ -0.4 & \text{otherwise}, \end{cases} \tag{9}$$

where h_i is the desired height of foot i and $k_i = (8t + k_{i,0}) \bmod 4$ is a phase parameter. $k_{i,0} = 0$ for $i = 0, 3$ and $k_{i,0} = 2$ for $i = 1, 2$. The joint commands are calculated by inverse kinematics and sent to the actuators, and the joint positions, velocities, and torques are recorded at 1000 Hz. Then, the parameters of EAM are identified

using the method described in Section 3.1 and listed in Table 4. EAM achieves an identification error of 0.066 $N^2 \cdot m^2$, reduced by 86.9% compared to PD, whose error is 0.503 $N^2 \cdot m^2$.

To validate the generalization of EAM, a second data set is also collected, which includes various forms of locomotion, such as standing, walking, running, rotating, rapid acceleration, and deceleration. The actuator dynamics become more difficult to

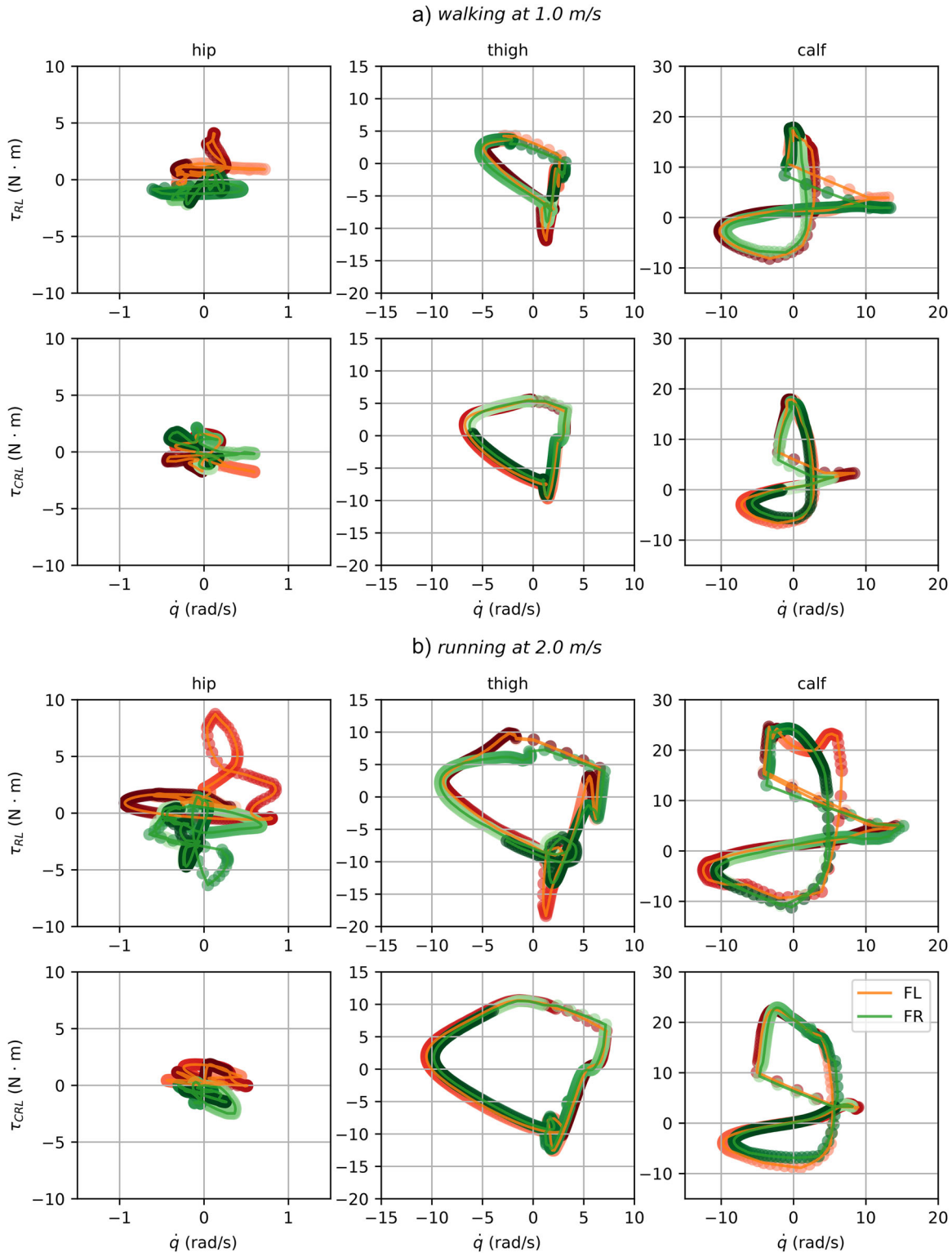

FIGURE 7 | The symmetry comparison between the RL and CRL policies is shown. The torque–velocity curves are collected from the actuators of the two front legs in a gait cycle at 1 and 2 m/s respectively. The policy trained with CRL generates consistent trajectories for the symmetric two legs, while the RL policy produces asymmetric trajectories. This contributes to the balancing of locomotion and also saves energy by encouraging a fair distribution of torque.

predict under such complex locomotion, resulting in an increased identification error to 0.387 $N^2 \cdot m^2$. However, EAM maintains a 92.5% lead over PD, whose error increases significantly to 5.149 $N^2 \cdot m^2$. The results suggest that EAM can simulate the behavior of the actuator in a variety of scenarios. Figure 4 shows the measured torque and the prediction of PD and EAM during walking.

4.3 | Locomotion Performance

Our controller demonstrates superior performance locomotion performance in field scenarios. Figure 5 shows some scenes of outdoor locomotion, where our controller demonstrates impressive robustness and agility on natural terrains such as grass and slopes. Our controller achieves a maximum sustained velocity of 4.2 m/s, while the model predictive control (MPC) controller provided by the manufacturer is limited to a maximum velocity of 1.6 m/s. This is because our controller dynamically adjusts the gait frequency, stride, and contact ratio based on the velocity, as illustrated in Figure 6, whereas the MPC employs a predefined fixed gait. This characteristic allows for accurate and efficient tracking at arbitrary velocity. Additionally, our controller achieves a maximum backward velocity of 3.2 m/s, lateral velocity of 2.0 m/s, and spinning velocity of 6.0 rad/s separately. To further evaluate sustained performance, we conduct a human-following test of 3.1 km, shown in Figure 1, traversing diverse terrain types: asphalt pavement, slopes exceeding 15° inclination, gravel paths, and uneven grassland. The robot finishes the tour within 37 min without

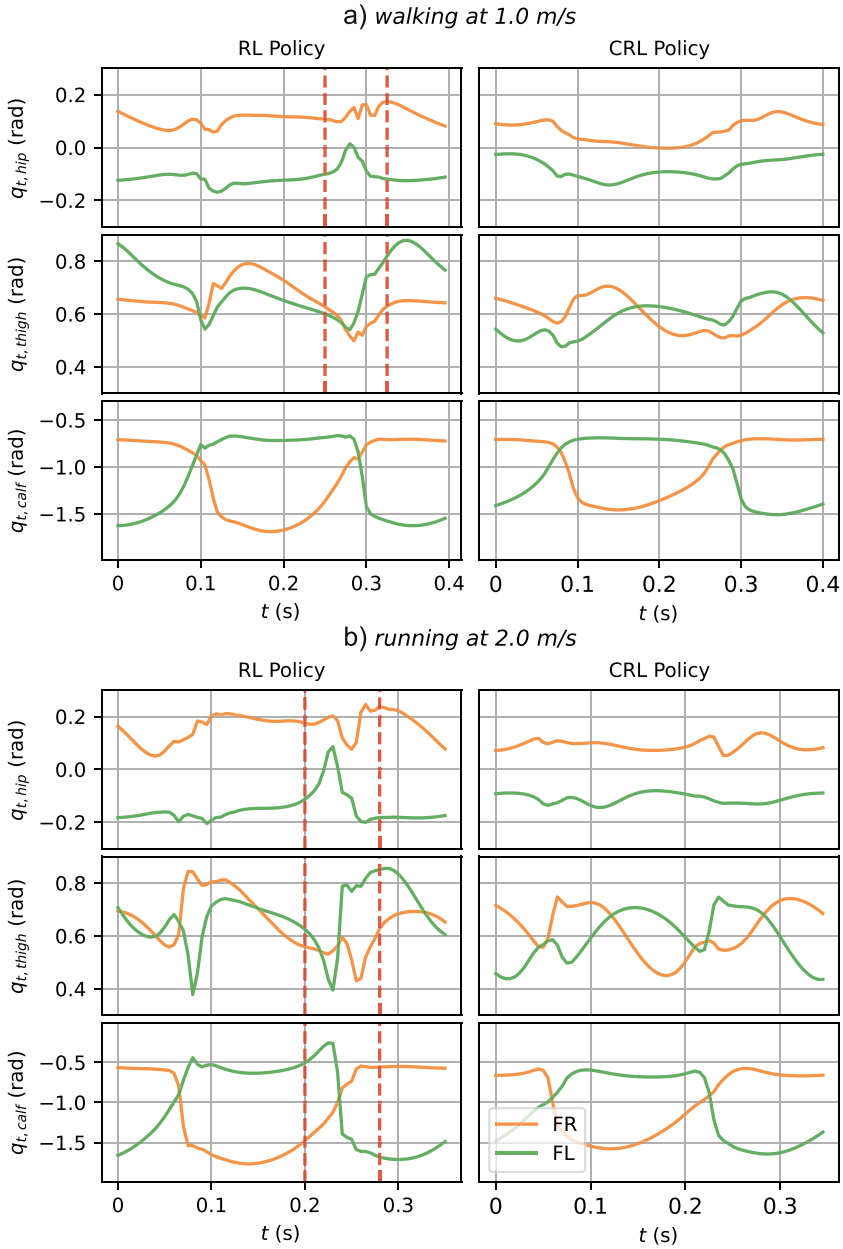

FIGURE 8 | The action curves of RL and CRL policies in a gait cycle are shown. Actions between the two red dotted lines are considered to be jerky. In contrast, CRL policy produces significantly smoother actions, which makes it easier to deploy to physical robots and produces more stable motions.

human intervention and achieves an average linear velocity error of 0.077 m/s and an angular velocity error of 0.059 rad/s.

4.4 | Ablation Study

To evaluate the effectiveness of the proposed framework, we fully test the following experimental setups:

- *MPC* is the MPC controller provided by the robot manufacturer.
- *RL-PD* is the typical setup, training the controller with RL and simulating the actuator with PD control.
- *RL-EAM* trains the controller with RL and simulates the actuator with EAM instead.
- *CRL-EAM* is our proposed framework, which trains the controller with Constrained RL and simulates the actuator with EAM.

The reward of RL policies includes an additional action smoothness reward $r_t = -|a_t - 2a_{t-1} + a_{t-2}|$ with coefficient 0.01, as an alternative to the smooth constraint. The w_{sym} and w_{smooth} of CRL-EAM are set to 1.0 and 0.05, respectively.

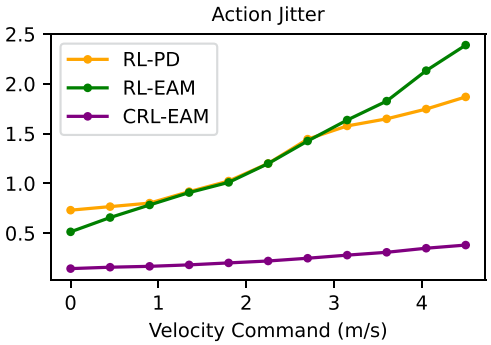

FIGURE 9 | The action jitter of different policies is shown. The jitter is calculated as the average absolute 2-order difference of three consecutive actions. The jitter of RL policies is significantly higher than that of CRL policies, which brings difficulties for sim-to-real transfer.

Figures 7 and 8 show the direct effect of the symmetry and smoothness constraints on the policy. CRL increases the symmetry of the policy within a gait cycle. For example, it encourages two symmetrical legs to alternate in front when walking forward, one for support and the other for striding. It also encourages symmetrical legs to have analogous gait cycles and leg lift times. In contrast, RL policies tend to support more body weight on a single leg, resulting in lameness, increased energy consumption, and instability in velocity tracking. RL policies also suffer from their inherent jerky actions, while the CRL policy produces significantly smoother action sequences, as illustrated by Figures 8 and 9. As a result, the CRL policy can be easily transferred to reality and produce predictable performance on the robot.

The policies are transferred to the physical robot for validation. For policies trained with RL only, a low-pass filter must be applied to the action to reduce jitter; otherwise, the robot will tremble violently and make shrill noises. The CRL-EAM policy is directly deployed to the robot without any additional filtering. We conduct comprehensive evaluations of omnidirectional velocity tracking performance for the four controllers. The robot is commanded with velocity values spanning v_x ranging from -2.5 to 2.5 m/s in increments of 0.5 m/s and v_y ranging from -1.0 to 1.0 m/s in increments of 0.2 m/s. Tracking errors are quantified as the deviation between the commanded velocities and the measured average velocities, with results visualized in Figure 10. The CRL-EAM policy exhibits consistently lower tracking errors compared to baseline methods, highlighting its superior performance in achieving precise velocity control. The accuracy of velocity tracking is also evident in the performance of open-loop trajectory tracking, shown in Figure 11, where our proposed policy shows the lowest trajectory deviation.

While these policies perform similarly in simulation, their behavior on the physical robot is quite different. The RL-PD policy struggles to achieve velocities greater than 2.5 m/s, and it often staggers with noticeable body vibrations during running. RL-PD also produces exaggerated limb movements during acceleration and deceleration. In contrast, both EAM policies passed the test successfully. RL-EAM successfully tracks each velocity command with a lower average error than RL-PD, maintains stable orientation, and produces less vibration. The

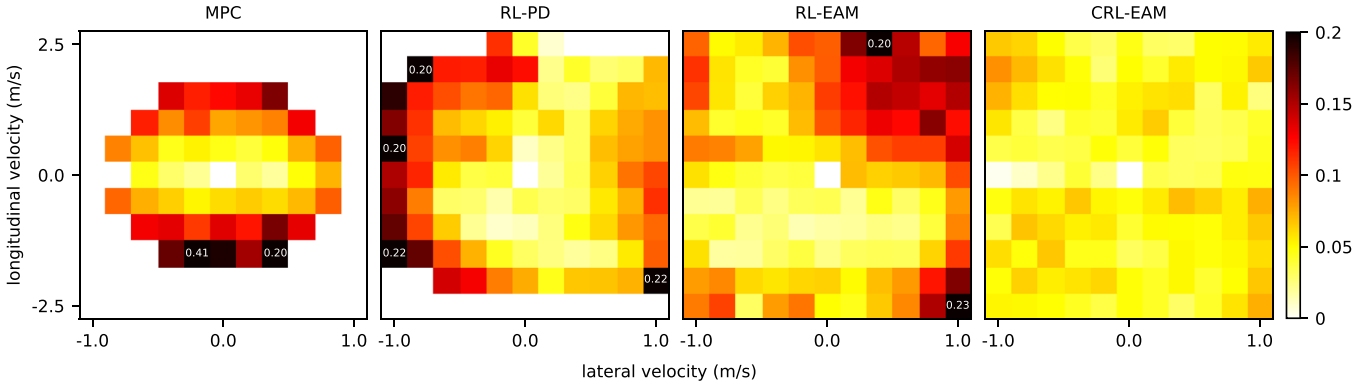

FIGURE 10 | The velocity tracking performance of MPC and learning-based controllers is shown. CRL-EAM achieves higher locomotion capability and tracking accuracy than ablation controllers. The superiority is attributed to the empirical actuator model, which helps reduce the sim-to-real gap, and constrained RL, which promotes symmetry and smoothness.

EAM policies never fail throughout the test, while RL-PD fails three times when given the velocity command (v_x, v_y) of $(-2.5, 0), (-2.0, -0.8)$, and $(2.0, -1.0)$, and MPC fails once when commanded at $(-0.8, 0)$. These results suggest that EAM can effectively reduce the sim-to-real gap and optimize the transfer of the policy's performance to real robots.

CRL-EAM exhibits superior velocity tracking accuracy throughout the velocity command space, whereas the RL policies show inconsistent performance and lower accuracy. RL-PD performs acceptably when commanded at $(2.0, 1.0)$, but fails when commanded at $(2.0, -1.0)$. RL-EAM has relatively high tracking accuracy while commanded to move forward to the right, but much lower accuracy when commanded to move to the left. These phenomena indicate that they learn each velocity command individually and do not generalize across commands, resulting in a maximum tracking error of over 0.2 m/s. CRL-EAM, on the other hand, shows better symmetry and omnidirectional consistency, achieving a maximum tracking error of 0.084 m/s, which is 61.8% lower than the RL policies. The reason for this disparity is that CRL synthesizes the experience of symmetric situations, allowing it to perform consistently in all directions. This makes our controller particularly suited to track the velocity commands given by the upper-level planner with a small pose and velocity error. It also ensures the safety of the robot in case of obstacles around.

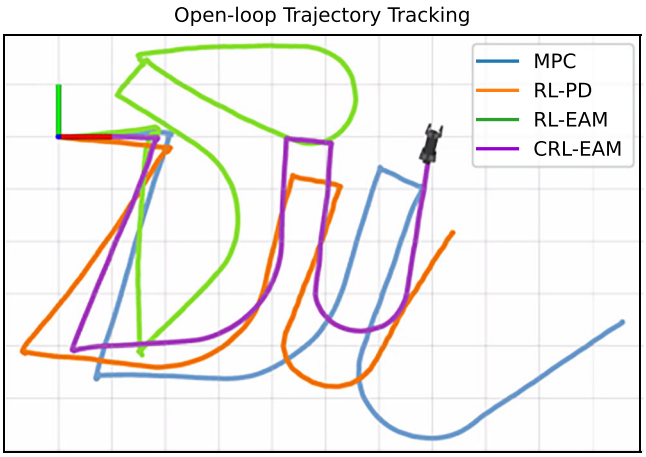

FIGURE 11 | The results of open-loop tracking of predefined trajectories with different controllers are shown. The trajectory is shaped like the letters "ZJU" and contains both translation and rotation. Among these controllers, CRL-EAM achieves the lowest trajectory deviation, demonstrating the superiority of our method in terms of velocity tracking accuracy.

Figure 12 illustrates the energy efficiency of different controllers. It can be evaluated using the cost of transport (CoT) and the cost of turning (CoTu) defined as

$$\text{CoT} = \int P dt / mg\bar{v}, \text{CoTu} = \int P dt / mgl_c r_z, \tag{10}$$

where $P = \sum_i \max(k_r \tau_i^2 + \tau_i \dot{q}_i, 0)$ is the total power of the 12 actuators, $\bar{v}$ is the average linear velocity in the commanded direction, and $l_c = 0.555$ is the characteristic length, the default distance between two diagonal feet. CRL-EAM achieves the highest energy efficiency by promoting a balanced distribution of payloads among the limbs with the symmetry constraint. It also tends to perform a catwalk, which reduces the gravity on the shoulder joint and saves energy.

5 | Conclusion

This paper presents a learning-based framework to enable accurate and stable velocity tracking for quadrupedal robots. We first introduce an EAM for actuator modeling and simulation, which significantly reduces the sim-to-real gap. We also propose a CRL framework for training a symmetric and smooth control policy for quadrupedal robots. The proposed framework is extensively validated through real-world experiments, demonstrating superior velocity tracking accuracy, and robustness compared to ablation controllers. For future work, we aim to incorporate exteroceptive perception during policy training to enable adaptive navigation in unstructured environments. Additionally, we plan to extend the symmetry and smoothness

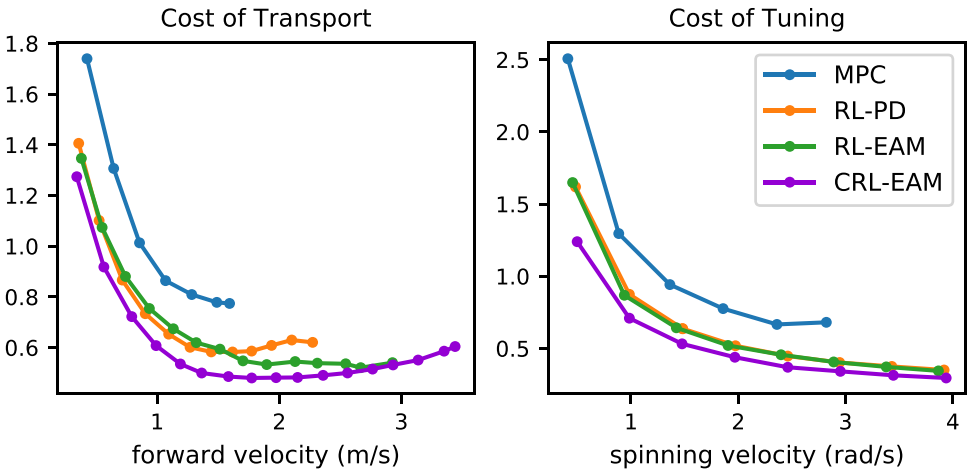

FIGURE 12 | The energy efficiency of MPC and learning-based controllers is shown. CRL-EAM shows clear advantages over other controllers in terms of energy efficiency in both translation and rotation.

constraints by integrating terrain-aware gait transitions and energy-optimal motion patterns.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Agarwal, A., A. Kumar, J. Malik, and D. Pathak. 2023. "Legged Locomotion in Challenging Terrains Using Egocentric Vision." In *Proceedings of the 6th Conference on Robot Learning and Machine Learning Research*, edited by K. Liu, D. Kulic and J. Ichnowski, vol. 205, 403–415. PMLR.

Aswin Nahrendra, I. M., B. Yu, and H. Myung. 2023. "DreamWaQ: Learning Robust Quadrupedal Locomotion With Implicit Terrain Imagination via Deep Reinforcement Learning." In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 5078–5084.

Choi, S., G. Ji, J. Park, et al. 2023. "Learning Quadrupedal Locomotion on Deformable Terrain." *Science Robotics* 8, no. 74: eade2256.

Di Carlo, J., P. M. Wensing, B. Katz, G. Bledt, and S. Kim. 2018. "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control." In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1–9.

Elfwing, S., E. Uchibe, and K. Doya. 2018. "Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning." *Neural Networks* 107: 3–11.

Feng, G., H. Zhang, and Z. Li, et al. 2023. "Genloco: Generalized Locomotion Controllers for Quadrupedal Robots." In *Proceedings of The 6th Conference on Robot Learning and Machine Learning Research*, edited by K. Liu, D. Kulic and J. Ichnowski, vol. 205, 1893–1903. PMLR.

Ha, S., P. Xu, Z. Tan, S. Levine, and J. Tan. 2020. "Learning to Walk in the Real World With Minimal Human Effort." Preprint, arXiv, arXiv:2002.08550.

Haarnoja, T., S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine. 2018. "Learning to Walk via Deep Reinforcement Learning." Preprint, arXiv, arXiv:1812.11103.

Hochreiter, S., and J. Schmidhuber. 1997. "Long Short-Term Memory." *Neural Computation* 9, no. 8: 1735–1780.

Hoyt, D. F., and C. R. Taylor. 1981. "Gait and the Energetics of Locomotion in Horses." *Nature* 292, no. 5820: 239–240.

Hwangbo, J., J. Lee, A. Dosovitskiy, et al. 2019. "Learning Agile and Dynamic Motor Skills for Legged Robots." *Science Robotics* 4, no. 26: eaau5872.

Hwangbo, J., J. Lee, and M. Hutter. 2018. "Per-Contact Iteration Method for Solving Contact Dynamics." *IEEE Robotics and Automation Letters* 3, no. 2: 895–902.

Iqbal, A., Y. Gao, and Y. Gu. 2020. "Provably Stabilizing Controllers for Quadrupedal Robot Locomotion on Dynamic Rigid Platforms." *IEEE/ASME Transactions on Mechatronics* 25, no. 4: 2035–2044.

Jakobi, N., P. Husbands, and I. Harvey. 1995. "Noise and the Reality Gap: The Use of Simulation in Evolutionary Robotics." In *Advances in Artificial Life: Proceedings of the Third European Conference on Artificial Life Granada, Spain, June 4–6, 1995*, 704–720. Springer.

Ji, G., J. Mun, H. Kim, and J. Hwangbo. 2022. "Concurrent Training of a Control Policy and a State Estimator for Dynamic and Robust Legged Locomotion." *IEEE Robotics and Automation Letters* 7, no. 2: 4630–4637.

Jin, Y., X. Liu, Y. Shao, H. Wang, and W. Yang. 2022. "High-Speed Quadrupedal Locomotion by Imitation-Relaxation Reinforcement Learning." *Nature Machine Intelligence* 4: 1198–1208.

Kim, D., J. Di Carlo, B. Katz, G. Bledt, and S. Kim. 2019. "Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control." Preprint, arXiv, arXiv:1909.06586.

Kumar, A., Z. Fu, D. Pathak, and J. Malik. 2021. "RMA: Rapid Motor Adaptation for Legged Robots." Preprint, arXiv, arXiv:2107.04034.

Lee, J., J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. 2020. "Learning Quadrupedal Locomotion Over Challenging Terrain." *Science Robotics* 5, no. 47: eabc5986.

Lee, T., and F. C. Park. 2018. "A Geometric Algorithm for Robust Multibody Inertial Parameter Identification." *IEEE Robotics and Automation Letters* 3, no. 3: 2455–2462.

Li, C., T. Zhang, and D. I. Goldman. 2013. "A Terradynamics of Legged Locomotion on Granular Media." *Science* 339, no. 6126: 1408–1412.

Lu, G., T. Chen, X. Rong, et al. 2023. "Whole-Body Motion Planning and Control of a Quadruped Robot for Challenging Terrain." *Journal of Field Robotics* 40, no. 6: 1657–1677.

Margolis, G., G. Yang, K. Paigwar, T. Chen, and P. Agrawal. 2022. "Rapid Locomotion via Reinforcement Learning." In *Robotics: Science and Systems*.

Miki, T., J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. 2022. "Learning Robust Perceptive Locomotion for Quadrupedal Robots in the Wild." *Science Robotics* 7, no. 62: eabk2822.

Moeckel, R., Y. N. Perov, and A. T. Nguyen, et al. 2013. "Gait Optimization for Roombots Modular Robots-Matching Simulation and Reality." In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3265–3272. IEEE.

Mordatch, I., K. Lowrey, and E. Todorov. 2015. "Ensemble-CIO: Full-Body Dynamic Motion Planning That Transfers to Physical Humanoids." In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5307–5314. IEEE.

Neunert, M., T. Boaventura, and J. Buchli. 2017. "Why Off-the-Shelf Physics Simulators Fail in Evaluating Feedback Controller Performance—A Case Study for Quadrupedal Robots." In *Advances in Cooperative Robotics*, 464–472. World Scientific.

Peng, X. B., M. Andrychowicz, W. Zaremba, and P. Abbeel. 2018a. "Sim-to-Real Transfer of Robotic Control With Dynamics Randomization." In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 3803–3810.

Peng, X. B., M. Andrychowicz, W. Zaremba, and P. Abbeel. 2018b. "Sim-to-Real Transfer of Robotic Control With Dynamics Randomization." In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 3803–3810. IEEE.

Peng, X. B., G. Berseth, and M. Van de Panne. 2015. "Dynamic Terrain Traversal Skills Using Reinforcement Learning." *ACM Transactions on Graphics (TOG)* 34, no. 4: 1–11.

Peng, X. B., G. Berseth, and M. Van de Panne. 2016. "Terrain-Adaptive Locomotion Skills Using Deep Reinforcement Learning." *ACM Transactions on Graphics (TOG)* 35, no. 4: 1–12.

Peng, X. B., E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine. 2020. "Learning Agile Robotic Locomotion Skills by Imitating Animals." Preprint, arXiv, arXiv:2004.00784.

Pinto, L., J. Davidson, R. Sukthankar, and A. Gupta. 2017. "Robust Adversarial Reinforcement Learning." In *International Conference on Machine Learning*, 2817–2826. PMLR.

Raffin, A., J. Kober, and F. Stulp. 2022. "Smooth Exploration for Robotic Reinforcement Learning." In *Proceedings of the 5th Conference on Robot Learning and Machine Learning Research*, edited by A. Faust, D. Hsu and G. Neumann, vol. 164, 1634–1644. PMLR.

Schaul, T., D. Borsa, J. Modayil, and R. Pascanu. 2019. "Ray Interference: A Source of Plateaus in Deep Reinforcement Learning." Preprint, arXiv, arXiv:1904.11455.

Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. 2017. "Proximal Policy Optimization Algorithms." Preprint, arXiv, arXiv:1707.06347.

Shao, Y., Y. Jin, X. Liu, W. He, H. Wang, and W. Yang. 2022. "Learning Free Gait Transition for Quadruped Robots via Phase-Guided Controller." *IEEE Robotics and Automation Letters* 7, no. 2: 1230–1237.

Shin, J., D. J. Kwak, and Y.-I. Lee. 2017. "Adaptive Path-Following Control for an Unmanned Surface Vessel Using an Identified Dynamic Model." *IEEE/ASME Transactions on Mechatronics* 22, no. 3: 1143–1153.

Tan, J., Z. Xie, B. Boots, and C. K. Liu. 2016. "Simulation-Based Design of Dynamic Controllers for Humanoid Balancing." In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2729–2736. IEEE.

Tan, J., T. Zhang, and E. Coumans, et al. 2018. "Sim-to-Real: Learning Agile Locomotion for Quadruped Robots." Preprint, arXiv, arXiv:1804.10332.

Tobin, J., R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. 2017. "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)*, 23–30.

Winkler, A. W., C. D. Bellicoso, M. Hutter, and J. Buchli. 2018. "Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization." *IEEE Robotics and Automation Letters* 3, no. 3: 1560–1567.

Wu, J., G. Xin, C. Qi, and Y. Xue. 2023. "Learning Robust and Agile Legged Locomotion Using Adversarial Motion Priors." *IEEE Robotics and Automation Letters* 8, no. 8: 4975–4982.

Yu, W., J. Tan, C. K. Liu, and G. Turk. 2017. "Preparing for the Unknown: Learning a Universal Policy With Online System Identification." Preprint, arXiv, arXiv:1702.02453.

Yu, W., G. Turk, and C. K. Liu. 2018. "Learning Symmetric and Low-Energy Locomotion." *ACM Transactions on Graphics* 37, no. 4: 1–12.

Zhang, Z., J. Yan, X. Kong, G. Zhai, and Y. Liu. 2021. "Efficient Motion Planning Based on Kinodynamic Model for Quadruped Robots Following Persons in Confined Spaces." *IEEE/ASME Transactions on Mechatronics* 26, no. 4: 1997–2006.

Zhu, S., A. Kimmel, K. E. Bekris, and A. Boularias. 2017. "Model Identification via Physics Engines for Improved Policy Search." CoRR, Preprint, arXiv, arXiv:1710.08893.

Supporting Information

Additional supporting information can be found online in the Supporting Information section.
Supplementary Information Video S1.