2025 IEEE-RAS 24th International Conference on Humanoid Robots (Humanoids)
September 30 – October 2, 2025 | COEX, Seoul, Korea

Learning Point-to-Point Bipedal Walking Without Global Navigation

Xueyan Ma, Boxing Wang[†], Chong Tian, Boyang Xing, and Yufei Liu

Abstract— **This paper presents a reinforcement learning framework for navigation-free point-to-point walking in bipedal robots. Unlike traditional velocity-command-based approaches, we introduce displacement-based commands that align more naturally with the discrete stepping nature of legged locomotion. Our method enables smooth transitions between standing and walking, accurate control over the number of steps, and support for both discrete and continuous walking trajectories. We design a phase-encoded command format and train the policy using a history of proprioceptive states and well-crafted reward functions. The policy is trained with PPO in Isaac Lab and deployed on a full-size humanoid robot (1.8 meters tall, 80 kg, 6 DoF per leg). In simulation, the controller achieves an average tracking error of less than 0.11 meters in position and 0.05 radians in heading. On hardware, the maximum errors reach to 0.3 meters and 0.2 radians due to sim-to-real discrepancies. The proposed framework demonstrates robust and efficient point-to-point locomotion without the need for high-level navigation modules.**

I. Introduction

Bipedal or humanoid robots are regarded as general-purpose machines intended for a wide range of applications, including fieldwork, industrial tasks, and household assistancemuch like human beings. Recent advances in deep reinforcement learning (RL) have enabled the development of controllers that significantly enhance bipedal robots' agility and robustness, particularly for navigating uneven terrain or executing highly dynamic motions. However, in industrial or domestic settings, precision of movement often takes precedence over agility.

Traditional approaches typically use base velocity commands as inputs and rely on a high-level navigation module to perform closed-loop point-to-point navigation. While this method is well-suited for wheeled robots, it is less appropriate for bipedal locomotion. In bipedal systems, discrete legged steps serve as the fundamental control primitives, in contrast to the continuous velocity control of wheeled platforms. Furthermore, base position control is generally more manageable in indoor environments, where foot slippage is minimal and turning does not produce counteracting forces between the feet, unlike the dynamics seen in differential drive systems.

Another drawback of typical velocity-based bipedal control methods is that continuous stepping motions are required even for minor base position adjustments. For instance, if a robot needs to move a short distance in just two or three steps, velocity-based methods first require switching to stepping mode, issuing velocity commands, and waiting for the robot to walk to the target location before stopping. This process often results in taking more steps than necessary, making the movement less efficient. The comparison between position control and velocity control is illustrated in Figure 1.

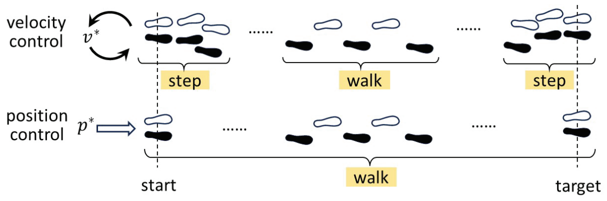


Fig. 1: Comparison between position and velocity control.

In this work, we explore reinforcement learning for bipedal robots using base displacement commands as inputs. Our approach builds upon previous RL methods by incorporating these commands into the reward function and training the policy on randomized point-to-point targets in simulation. This enables the robot to perform point-to-point base movements directly, without relying on a high-level navigation module. Moreover, the approach supports seamless transitions between standing and walking, allowing for precise control over the number of steps taken to ensure efficient movement. The proposed framework also accommodates continuous walking, which can be viewed as point-to-point movement toward an infinitely distant target.

We validate the proposed approach on a bipedal robot in both simulation and real-world hardware experiments (Figure 2). The learned controller successfully executes point-to-point or continuous walk in response to different displacement command sequences.

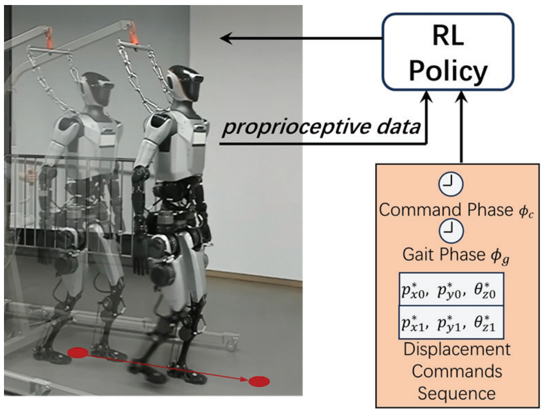
In summary, contributions of this work are as follows:

1) **We propose a reward-specification RL framework for navigation-free point-to-point walking.** We present reward functions that can train the RL policy to achieve point-to-point walk merely with proprioceptive feedback. Mode transitions between standing, point-to-point walking, and continuous walking are implicitly handled by the policy through the design of the input command sequence.
2) **We demonstrate the proposed framework in both simulation and hardware experiments.** The training is conducted in IsaacLab [1], [2] and deployed on the bipedal robot, QingLoong. Performance evaluations are presented for both simulation and real-world

All authors are with Humanoid Robot (Shanghai) Co., Ltd., Shanghai, China.
[†] boxing.wang@outlook.com

This work was funded by the National Natural Science Foundation of China, grant number 52205035

2025 IEEE-RAS 24th International Conference on Humanoid Robots (Humanoids) | 979-8-3315-9869-3/25/$31.00 ©2025 IEEE | DOI: 10.1109/HUMANOIDS65713.2025.11203054



RL Policy
proprioceptive data
Command Phase ϕ_c
Gait Phase ϕ_g
$p^*_{x0}, p^*_{y0}, \theta^*_{z0}$
$p^*_{x1}, p^*_{y1}, \theta^*_{z1}$
Displacement Commands Sequence

Fig. 2: Point-to-point bipedal walking.

hardware experiments.

3) **Open Source.** To facilitate future research, we release our training code with Isaac Lab. Code release: `https://github.com/BoxingWing/Bipedal-p2p-walk`

The remainder of this paper is organized as follows. Section II provides a brief overview of related work. Section III and IV discusses our proposed method thoroughly. Section V presents the experimental setting, results and performance evaluation. Finally, Section VI concludes this work and briefly discusses directions for future work.

II. Related Work

Model-based planning and control methods offer fundamental insights into bipedal locomotion. Early methods using simplified models like the inverted pendulum [3] and zero-moment-point (ZMP) [3] focused on static gait planning by relying heavily on support polygons for accurate foot placement and base control. However, this often limited robustness and agility. To overcome these, research advanced toward dynamic gait generation using more complex models such as centroidal [4], [5] and full-body dynamics [6], combined with optimal control methods including trajectory optimization [7], whole-body control [8], and model predictive control (MPC) [9]. While these methods enable agile behaviors like jumping, the gap between simulation and real deployment remains, due to assumptions like perfect state estimation and no foot slippage.

Reinforcement learning (RL) has become a powerful tool for robust legged locomotion. End-to-end policies mapping user commands directly to joint targets have been deployed on quadrupeds such as ANYmal [10], [11] and bipeds like Cassie [12], [13]. These policies, usually neural networks like MLPs, take inputs including user commands and proprioceptive or historical states [13]. With accurate joint modeling and dynamics randomization during training, several works have demonstrated zero-shot sim-to-real transfer [14]–[16].

Footstep Constrained Learning for bipedal Robots. When environments constrain feasible footsteps, some studies address footstep-constrained learning. Some approaches use known models or heuristics to pre-assign touchdown locations [17], [18], while others train gait policies to track pre-defined footsteps [19]–[21]. These relate closely to our work. However, in indoor point-to-point navigation without explicit footstep constraints, we let the policy learn footstep placements, focusing instead on tracking base displacement commands and managing transitions between standing and walking, as well as responding to sequential commands.

III. Design of Base Displacement Commands

To achieve the aforementioned navigation-free point-to-point walking, the input commands must be designed based on the desired position, desired heading, the arrival time, and other task-related parameters. Thus, we propose a new type of base displacement commands $\mathbf{u}$:

$$\begin{aligned}
\mathbf{u} &= [\boldsymbol{\phi}_c, \boldsymbol{\phi}_g, \boldsymbol{\xi}^*_0, \boldsymbol{\xi}^*_1] \in \mathbb{R}^{10}, \qquad (1)\\
\boldsymbol{\xi}^*_0 &= [p^*_{x0}, p^*_{y0}, \theta^*_{z0}],\\
\boldsymbol{\xi}^*_1 &= [p^*_{x1}, p^*_{y1}, \theta^*_{z1}],\\
\boldsymbol{\phi}_c &= [\sin(2\pi c_c(t)), \cos(2\pi c_c(t))],\\
\boldsymbol{\phi}_g &= [\sin(2\pi c_g(t)), \cos(2\pi c_g(t))],\\
\dot{c}_c &= 1/T_c,\\
\dot{c}_g &= 1/T_g,
\end{aligned}$$

where $\boldsymbol{\phi}_c$ and $\boldsymbol{\phi}_g$ are phase signals generated by timing clocks associated with the movement duration clock $c_c(t)$ and the gait clock $c_g(t)$, respectively. T_c denotes the current desired duration for moving from the start to the target position. T_g denotes the desired gait period. p^*_x, p^*_y and θ^*_z corresponds to desired base displacements for the *x*, *y* positions and heading. The subscript *0* or *1* represents the current and the next movement period. We define $\boldsymbol{\xi}$ as a pose vector comprising the position p_x, p_y and orientation θ_z.

Specifically, phase signal $\boldsymbol{\phi}_c(c_c)$ represents the execution progress of the current movement period. c_c increasing linearly from 0 to 1 at rate $1/T_c$ and reset to 0 when the current movement period is finished. Phase signal $\boldsymbol{\phi}_g(c_g)$ represents the progression of the gait cycle, with slope $1/T_g$. Providing explicit gait phase signals help the robot avoid unnecessary stepping.

The base displacement commands form a two-period sequence: the current commands $\boldsymbol{\xi}^*_0$ and the next commands $\boldsymbol{\xi}^*_1$, both in the body frame. They are updated only upon the completion of the current instruction. During the update, $\boldsymbol{\xi}^*_1$ is assigned the new desired value and $\boldsymbol{\xi}^*_0$ inherits the previous value of $\boldsymbol{\xi}^*_1$. This enables better coordination of velocity and motion between adjacent commands, and it also supports both single-point-to-point and continuous walking modes.

The single point-to-point mode requires the robot to walk from start to target and stop in a standard standing posture with feet aligned. In this mode, $\boldsymbol{\xi}^*_0$ is set to the target, while $\boldsymbol{\xi}^*_1$ is set to zero. In the continuous walking mode, both sets of parameters $\boldsymbol{\xi}^*_0$ and $\boldsymbol{\xi}^*_1$ are assigned consecutive target positions. In this mode, the robot does not stop at the end of each movement period but instead continues walking across command transitions. Smooth transitions arise implicitly from reward design without explicit handling.

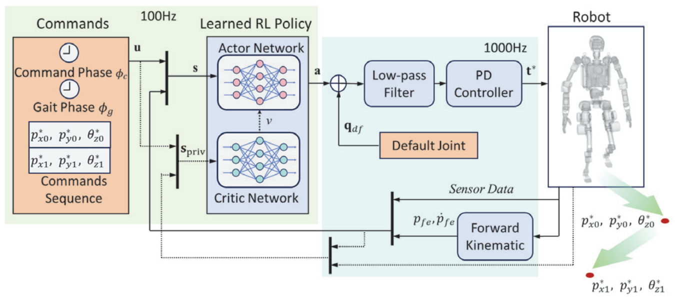


Fig. 3: The control structure: the solid line represents the data stream during deployment, and the dashed line during training.

IV. METHODS

A. *Control structure*

The control structure consists of an RL policy that generates desired joint positions at 100 Hz, followed by a low-pass filter and a PD controller operating at 1000 Hz. The PD controller is responsible for converting the desired joint positions into desired joint torques. The RL policy receives joint positions, base orientation, angular velocity, and foot-end postures (from simulation during training and from Pinocchio [22] in deployment). An overview of the control structure is shown in Figure 3.

The state space of the policy consists of proposed base displacement commands and observed robot states, including 15 frames of historical data, resulting in a dimension of $\mathbf{s} \in \mathbb{R}^{63\times 15}$ (reshaped to $\mathbb{R}^{945}$ for input to the network). For each frame, $\mathbf{s}_i$ is defined as follows:

$$\mathbf{s}_i = [\mathbf{u}, \mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\omega}, \boldsymbol{\theta}_{xy}, \mathbf{a}_{\text{pre}}, \mathbf{x}_{\text{f}}] \in \mathbb{R}^{63}, \tag{2}$$

where $\mathbf{u} \in \mathbb{R}^{10}$ are the phase and command inputs. $\mathbf{q} \in \mathbb{R}^{12}$ and $\dot{\mathbf{q}} \in \mathbb{R}^{12}$ represent the joint positions and velocities respectively. $\mathbf{q}$ represents the joint positions relative to the default joint positions $\mathbf{q}_{\text{df}}$. $\boldsymbol{\omega} \in \mathbb{R}^3$ and $\boldsymbol{\theta}_{xy} \in \mathbb{R}^2$ represent the base angular velocity and the rollpitch orientation, respectively. $\mathbf{a}_{\text{pre}} \in \mathbb{R}^{12}$ is the previous policy output and $\mathbf{x}_{\text{f}} \in \mathbb{R}^{12}$ is the foot-end postures in the body frame. $\mathbf{x}_{\text{f}}$ directly gives foot end postures and supports the policy to infer the robots kinematic model more easily.

The 15-frame history sequence provides information about past states, which helps in inferring motion trends and unobservable states. Experimental results show that a short history length can lead to slower improvement in certain rewards, such as position tracking. In contrast, a long history input increases computational cost due to the increased input dimension.

The action space of the policy, $\mathbf{a} \in \mathbb{R}^{12}$, is defined as the desired joint positions of the 12 leg joints. The final desired joint positions are obtained by adding the network outputs $\mathbf{a}$ to $\mathbf{q}_{df}$.

B. *Network and training setup*

We use an asymmetrical Actor-Critic architecture with Proximal Policy Optimization (PPO), as implemented in the RSL-RL framework [11]. Both the Actor and Critic networks consist of three hidden layers in a multilayer perceptron (MLP) with [512, 256, 256] units and use ELU activations. Training is performed in Isaac Lab [1] on 4096 parallel environments. The policy is trained in two stages: The first stage learns basic behaviors under mild conditions. The second stage adds stronger disturbances and increases difficulty to enhance robustness and tracking.

Privileged observations $\mathbf{s}_{\text{priv}}$ are the inputs to the Critic network during training. $\mathbf{s}_{\text{priv}}$ includes 5 historical frames, resulting in a privileged observation vector of size $\mathbf{s} \in \mathbb{R}^{108\times 5}$ (reshaped to $\mathbb{R}^{540}$ for input to the network). For each frame, $\mathbf{s}_{\text{priv},i}$ is defined as follow:

$$\begin{aligned}\mathbf{s}_{\text{priv},i} = [&\mathbf{s}_0, \mathbf{v}_{\text{b}}, (\hat{\mathbf{p}}_{xy} - \mathbf{p}_{xy}), (\hat{\mathbf{v}}_{xy} - \mathbf{v}_{xy}), \theta_{z,\text{start}},\\ &\theta_{z,\text{muti}}, f_{r,s}, m, \mathbf{g}_{\text{stance}}, \mathbf{g}_c, \mathbf{f}_c, \mathbf{k}_p, \mathbf{k}_d] \in \mathbb{R}^{108},\end{aligned} \tag{3}$$

where $\mathbf{s}_0 \in \mathbb{R}^{63}$ is the current observation. $\mathbf{v}_{\text{b}} \in \mathbb{R}^3$ represents linear velocity in the body frame obtained from the simulation environment. $\mathbf{p}_{xy} \in \mathbb{R}^2$ and $\mathbf{v}_{xy} \in \mathbb{R}^2$ are the base position and linear velocity in the world frame. $\hat{\mathbf{p}}_{xy} - \mathbf{p}_{xy}$ are position and velocity errors in the world frame, where * denotes targets inputs, and $\hat{}$ denotes processed targets. $\theta_{z,\text{start}}$ denotes the heading at commands start. $\theta_{z,\text{muti}}$ is the accumulated yaw. $f_{r,s}$ denotes the static friction, and m is the mass of the robot. $\mathbf{g}_{\text{stance}} \in \mathbb{R}^2$ is the stance leg flags, and $\mathbf{g}_c \in \mathbb{R}^2$ is defined as actual contact flags. $\mathbf{f}_c \in \mathbb{R}^6$ represents the feet contact forces. Each element $\mathbf{g}_c^{(i)}$ is set to 1 if the norm of the contact force for the corresponding leg i exceeds 5 N, and 0 otherwise. $\mathbf{k}_p \in \mathbb{R}^{12}$ and $\mathbf{k}_d \in \mathbb{R}^{12}$ are the joints stiffness and damping. In addition, $f_{r,s}$, m, $\mathbf{k}_p$, and $\mathbf{k}_d$ are randomized within predefined ranges rather than fixed values.

C. *Command resampling*

During training, base displacement commands are generated from randomly sampled linear velocities along x- and y-axes (from [-0.4, 0.6] m/s and [-0.4, 0.4] m/s) and angular velocity around the z-axis (from [-0.8, 0.8] rad/s). The desired motion duration T_c is sampled within [0.45, 4.05] s, while the gait period T_g is fixed at 0.9 s. These samples produce the base displacement commands $\boldsymbol{\xi}_1^*$.

A random value determines if the robot stands or walks: if less than 0.8, $\boldsymbol{\xi}_1^*$ is zero (standing mode); otherwise, it follows the generated displacement commands.

D. *Reward design*

We explain several key rewards associated with base displacement control in detail below. To compactly express recurring structures in the reward terms, we define

$$E(x) = a \cdot \exp\{-b\|x\|\}, \quad L(x) = c\|x\|,$$

$$\Psi(x) = E(x) - L(x),$$

where a, b and c are coefficients that vary depending on the specific reward term. Here, $E(x)$, $L(x)$, and $\Psi(x)$ represent the exponential, linear, and combined forms, respectively.

Displacement Tracking. These rewards are designed to minimize the error between the robot's current pose $\boldsymbol{\xi}$ and the reference commands $\boldsymbol{\xi}^*$. The rewards are computed as follows:

$$\begin{aligned} r_p = &\, \Psi(\hat{\mathbf{p}}_{xy} - \mathbf{p}_{xy}) \\ &-1.3(\log\left\{\|\mathbf{p}^*_{xy} - \mathbf{p}_{xy}\|\right\} + 1\text{e-5}), \end{aligned} \tag{4}$$

$$r_v = \Psi(\hat{\mathbf{v}}_{xy,\text{his}} - \mathbf{v}_{xy,\text{his}}), \tag{5}$$

$$r_\theta = \Psi(\hat{\theta}_z - \theta_z) + E(\theta^*_z - \theta_z), \tag{6}$$

$$r_\omega = \Psi(\hat{\omega}_z - \omega_z), \tag{7}$$

For the position tracking reward r_p, the last two terms substantially improve tracking accuracy. Specifically, the negative log term is based on the target displacement $\mathbf{p}^*_{xy}$ and gives much higher rewards for small errors, which effectively promotes accurate tracking. For the velocity tracking reward r_v, we compute the averaged error over a horizon between the historical expected velocity $\hat{\mathbf{v}}_{xy,\text{his}}$ and the historical measured velocity $\mathbf{v}_{xy,\text{his}}$. This formulation not only enhances tracking performance over time but also smooths the velocity profile. The attitude and angular velocity tracking rewards r_θ and r_ω follow the same design principles as r_p and r_v, respectively. $(\hat{\mathbf{p}}_{xy}, \hat{\mathbf{v}}_{xy}, \hat{\theta}_z, \hat{\omega}_z)$ are final target processed from the command $\boldsymbol{\xi}^*$.

$\hat{\boldsymbol{\xi}}$ are calculated as follows:

$$\hat{\mathbf{p}}_{xy} = c_c \mathbf{p}^*_{xy}, \hat{\theta}_z = c_c \theta^*_z, \tag{8}$$

$\hat{\mathbf{p}}_{xy}$, $\hat{\theta}_z$ follows a linear trajectory, with the final value being $\mathbf{p}^*_{xy}$, θ^*_z, instead of maintaining a constant value throughout. $\hat{\mathbf{v}}_{xy}$ and $\hat{\omega}_z$ is calculated as follows:

$$\hat{\mathbf{v}}_{xy} = \mathcal{L}\left(\frac{\mathbf{p}^*_{xy}}{T_c} + (\mathbf{p}^*_{xy} - \mathbf{p}_{xy})\right), \tag{9}$$

$$\hat{\omega}_z = \mathcal{L}\left(\frac{\theta^*_z}{T_c} + (\theta^*_z - \theta_z)\right), \tag{10}$$

where $\mathcal{L}$ denotes a low-pass filtering operation. The position error gain term ensures that the target velocity increases to maintain base displacement tracking performance when the robot is far from the target. As the robot approaches the target, the gain decreases to prevent the velocity from becoming too large.

Single-point-to-point and continuous walking modes supports. r_{stop} is designed for the single-point-to-point mode and computed as follows:

$$\begin{aligned} r_{\text{stop}} = &\, (E(\mathbf{v}) + E(\omega_z) + E(\mathbf{q} - \mathbf{q}_{\text{df}}) \\ &+ E(\mathbf{G}_{xy})) \cdot g_{\text{stand}}, \end{aligned} \tag{11}$$

where g_{stand} is the standing flag and $\mathbf{G}_{x,y}$ denotes the projection of gravity onto the x- and y-axes.

In the single-point-to-point mode, the robot stops after reaching the target position. The stop reward r_{stop} is applied only in the standing state to maintain zero base velocity and default joint positions. It is disabled in continuous walking to to prevent resetting $\mathbf{q}$ to $\mathbf{q}_{\text{df}}$ during command transitions.

Walk Gait. These following rewards are designed for walking with the desired style.

TABLE I: Parameters for domain randomization

Parameter	Range / Distribution	
	Stage 1	Stage 2
Joint Stiffness	[0.9, 1.1]	[0.5, 1.5]
Joint Damping	[0.9, 1.1]	[0.5, 1.5]
Base Mass	[0.0, 5.0]	[0.0, 5.0]
Rigid Mass	[0.0, 0.0]	[0.9, 1.1]
Joint Friction	[0.1, 2.0]	[0.1, 2.0]

Note: The parameter of base mass is randomized additively, while the other parameters are randomized multiplicatively.

The reward $r_{p_{\text{sw}}}$ encourages tracking the desired swing-leg foot-end position p^*_{sw} along the z-axis, and is activated only during the swing phase. The swing leg is determined by the sine component of the gait phase signal ϕ_g: a positive sine value indicates a right-leg swing, while a negative value indicates a left-leg swing. To initiate appropriate lateral movement, the initial phase of ϕ_g is selected based on the sign of the desired lateral displacement p^*_y. Specifically, when $p^*_y > 0$, ϕ_g starts in the negative sine region for a left-leg swing; when $p^*_y < 0$, ϕ_g starts in the positive sine region for a right-leg swing. This phase planning strategy ensures the correct leg initiates the motion, reducing unnecessary steps and improving efficiency in lateral walking.

In addition, the reward r_{sym} designed to promote leg symmetry. This reward is activated at the moments of foot contact, lift-off, and the middle phase of the gait. It encourages identical left-right liftoff and touchdown positions and symmetry along the x-axis.

Action Smoothness. To encourage smooth actions, we design the reward r_{af}, computed as follows:

$$r_{\text{af}} = \exp\left\{-\sum|\mathbf{a} - \mathbf{a}_{\text{filter}}|\right\} \tag{12}$$

where $\mathbf{a}$ denotes the current action, and $\mathbf{a}_{\text{filter}}$ is the action $\mathbf{a}$ processed by a low-pass filter. A smaller difference between $\mathbf{a}$ and $\mathbf{a}_{\text{filter}}$ indicates smoother policy outputs.

More detailed rewards and the constants in reward functions is shown in the open code repository because of the page limits.

E. Domain Randomization

We apply domain randomization to various physical and control parameters during training, including contact material properties, actuator gains, base mass, and joint friction. Table I summarizes these parameters and their randomized ranges, which are applied at the start of training. Additionally, a push force is applied to the robot at random intervals, every 6.0 seconds. This force is applied by setting the robot's velocity, constrained to a range of -0.2 to 0.2 in both the x- and y- axes.

V. RESULTS AND ANALYSIS

A. simulation results

In simulation, we assess the tracking accuracy of base displacement commands under two control modes: single-point-to-point and continuous walking.

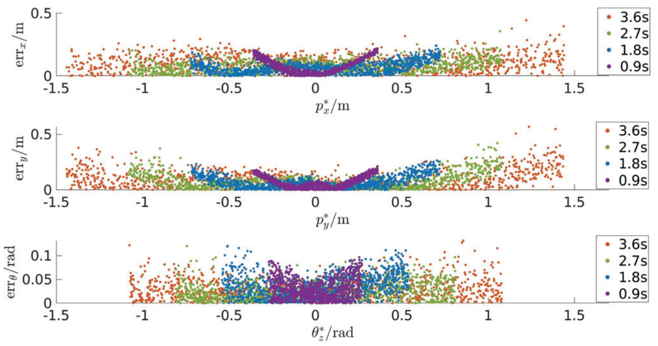


Fig. 4: Distribution of tracking errors in the single-point-to-point mode

In the single-point-to-point scenario, the robot is given isolated target base displacement commands. The evaluation is divided into 4 groups based on the desired execution durations: 0.9 s, 1.8 s, 2.7 s, and 3.6 s. In each group, 1000 robots are simulated, each assigned a random displacement target within the defined range before being commanded to stand.

The distribution of tracking errors is visualized in Figure 4. The results suggest that greater commanded displacements within the same desired duration result in larger tracking errors. This is because the commanded displacement is larger, leading to greater absolute errors. In addition, the results indicate that given the same displacement commands, longer desired durations generally reduce tracking errors, as the robot has more time to adjust its motion and improve accuracy.

The tracking error statistics indicate that, across all target durations, the mean position tracking error remains consistently below 0.11 meters, while the mean heading tracking error stays below 0.036 radians. In terms of variance, the position tracking standard deviation is generally less than 0.065 meters, and the heading tracking standard deviation is below 0.025 radians. These results suggest stable and reliable performance across different target durations, with relatively low variability in both position and heading tracking.

In the continuous walking mode, the robot is commanded to follow a sequence of five consecutive base displacements. The resulting tracking performance for position and heading is illustrated in Figure 5, and the associated errors are quantified in Table II. The recorded tracking deviations exhibit similar patterns to those observed in the single-point-to-point evaluation, further supporting the robustness of the learned policy under sequential motion planning.

B. experimental results

The learned policy is implemented on a physical robotic platform and validated through a series of systematic experiments. Its performance is evaluated in two separate sets of experiments with different desired durations, each consisting of ten groups of base displacement commands. The robots base position is defined as the midpoint between its feet, with actual displacements recorded accordingly. As shown

TABLE II: Tracking errors in the continuous walking mode

Command \ Error Type	p_x (m)	p_y (m)	θ_z (rad)
1st	0.0799	0.0312	-0.0336
2nd	0.0293	0.0009	-0.0385
3nd	0.0685	0.0465	-0.0141
4nd	-0.0031	0.0092	-0.0199
5nd	0.0255	0.0715	-0.0445

TABLE III: Tracking errors in hardware experiments

Desired Duration \ Error Type	p_x (m)	p_y (m)	θ_z (rad)
1.8s	0.064 (0.060)	0.059 (0.047)	0.103 (0.054)
3.6s	0.141 (0.090)	0.080 (0.065)	0.100 (0.062)

in Table III, the mean position tracking error is within 0.15 meters, and the mean heading error is within 0.11 radians. Their standard deviations remain below 0.1 meters and 0.1 radians. Figure 6 illustrates the tracking performance.

It shows that extending the desired duration doesn't always reduce tracking errors. Specifically, some experiments showed larger errors at 3.6 seconds than at 1.8 seconds in the x-axis position and heading angle. This may be caused by significant external disturbances and sensor noise in the real system, leading to greater error accumulation over extended durations. Among the three directions, the y-axis position shows consistently smaller errors and minimal fluctuations under both durations settings. It is also less influenced by motions in other directions. This indicates the most stable control performance in the y-axis position.

The policy is also evaluated in the continuous walking mode. In this test, the robot executes a sequence of three consecutive commands comprising translational motions along the x- and y-axes, as well as rotational motion. The results are presented in the accompanying video.

VI. CONCLUSIONS

In this work, we propose a reward-specification reinforcement learning (RL) framework for navigation-free point-to-point walking with legged robots. The proposed method is

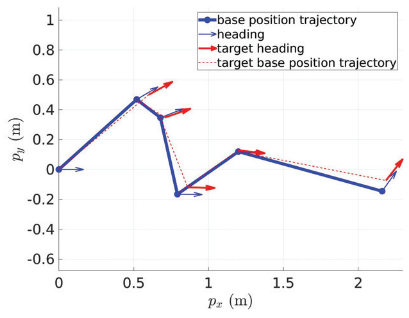


Fig. 5: Base position and heading tracking in the continuous walking mode

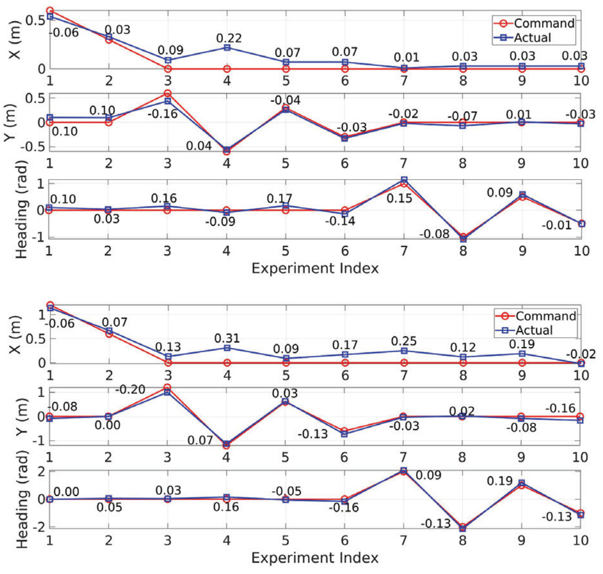


Fig. 6: Base position and heading tracking in the single-point-to-point mode. (Top) 1.8s desired duration; (Bottom) 3.6s desired duration.

evaluated in both simulation and real-world experiments. In simulation, the average tracking error is approximately 0.1 meters in translation and 0.03 radians in heading. In real-world experiments, the mean tracking error increases to around 0.2 meters and 0.1 radians due to the sim-to-real gap. The maximum errors reach 0.3 meter and 0.2 radians. Moreover, the framework supports continuous walking by accepting non-zero sequential displacement commands, enabling the robot to follow curved trajectories effectively.

Instead of conventional velocity commands, we adopt displacement-based command inputs to better align with the intrinsic nature of legged locomotion, where maintaining a constant velocity is often impractical. This design increases the flexibility of omnidirectional movement, allows for rapid acceleration and deceleration, and facilitates blind trajectory tracking, which is challenging for wheeled platforms. Our framework represents a step toward enabling legged robots to develop a spatial awareness of their own displacement. However, it still relies heavily on dense reward shaping through continuous position and velocity tracking. Future work will explore the integration of estimator networks to further enhance tracking accuracy and agility. Additionally, imitation learning from human demonstrations or offline optimized trajectories may contribute to more robust and natural point-to-point walking behaviors. Moreover, the adaptability of the proposed method to slippery or uneven terrain remains unverified. Future work will aim to extend the framework for accurate position tracking across varied terrain.

References

[1] NVIDIA, "Isaaclab: Unified framework for robot learning built on nvidia isaac sim," https://github.com/isaac-sim/IsaacLab, 2024.

[2] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.

[3] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, *Introduction to Humanoid Robotics*, 1st ed. Springer, 2014.

[4] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous robots*, vol. 35, pp. 161–176, 2013.

[5] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 295–302.

[6] K. Ishihara, T. D. Itoh, and J. Morimoto, "Full-body optimal control toward versatile and agile behaviors in a humanoid robot," *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 119–126, 2019.

[7] F. Jenelten, R. Grandia, F. Farshidian, and M. Hutter, "Tamols: Terrain-aware motion optimization for legged systems," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3395–3413, 2022.

[8] D. Kim, S. J. Jorgensen, J. Lee, J. Ahn, J. Luo, and L. Sentis, "Dynamic locomotion for passive-ankle biped robots and humanoids using whole-body locomotion control," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 936–956, 2020.

[9] E. Dantec, M. Naveau, P. Fernbach, N. Villa, G. Saurel, O. Stasse, M. Taix, and N. Mansard, "Whole-body model predictive control for biped locomotion on a torque-controlled humanoid robot," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*. IEEE, 2022, pp. 638–644.

[10] 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, 2019.

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

[12] J. Siekmann, Y. Godse, A. Fern, and J. Hurst, "Sim-to-real learning of all common bipedal gaits via periodic reward composition," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7309–7315.

[13] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control," *The International Journal of Robotics Research*, vol. 44, no. 5, pp. 840–888, 2024.

[14] Q. Liao, B. Zhang, X. Huang, X. Huang, Z. Li, and K. Sreenath, "Berkeley humanoid: A research platform for learning-based control," *arXiv preprint arXiv:2407.21781*, 2024.

[15] X. Gu, Y.-J. Wang, X. Zhu, C. Shi, Y. Guo, Y. Liu, and J. Chen, "Advancing humanoid locomotion: Mastering challenging terrains with denoising world model learning," in *Robotics: Science and Systems*, 2024.

[16] X. Gu, Y.-J. Wang, and J. Chen, "Humanoid-gym: Reinforcement learning for humanoid robot with zero-shot sim2real transfer," *arXiv preprint arXiv:2404.05695*, 2024.

[17] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.

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

[19] H. Duan, A. Malik, J. Dao, A. Saxena, K. Green, J. Siekmann, A. Fern, and J. Hurst, "Sim-to-real learning of footstep-constrained bipedal dynamic walking," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 10 428–10 434.

[20] 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.

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

[22] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.