Reinforcement Learning based Locomotion and Gait Generation for Petoi Bittle

Group 6, Construction Robotics, WS 2024/25

Sucheth Shenoy (564379), Harshith Gowda Shakaladevanapura Maregowda (565699), Po-Jen Sun (602222)

***Abstract*—Quadruped robots face significant challenges in adapting their locomotion to varied and dynamic terrains. Current hardcoded gait generation approaches lack the flexibility to address high-dimensional dynamics, terrain variability, and physical limitations effectively. To overcome these limitations, this study explores reinforcement learning (RL) as a tool for enabling autonomous gait generation and step-climbing tasks. While RL has shown promise in simulation environments, its application to real-world quadrupedal locomotion remains underexplored.**

This research bridges the gap by focusing on sim-to-real transfer ideas and validating them through the implementation and evaluation of an RL-based control framework for the Petoi Bittle. The methodology begins with designing reward functions tailored for gait generation and step-climbing tasks, followed by implementing and benchmarking various RL algorithms within simulation environments using OpenAI Gym and PyBullet to identify the most effective strategies. The RL policies are validated on hardware, emphasizing performance metrics and identifying limitations for further refinement. Key results are expected to demonstrate the feasibility of RL for generating stable and adaptive gaits in simulation while providing insights into strategies for overcoming real-world transfer challenges. This work is anticipated to underscore the potential of RL in advancing quadruped locomotion, paving the way for more autonomous and versatile robotic platforms.

***Index Terms*—Reinforcement Learning (RL), Quadruped Locomotion, Sim-to-Real Transfer, Petoi Bittle Robot**

I. Introduction

A. Topic background:

Quadruped robots have become a focus of robotics research due to their superior mobility in navigating complex and dynamic terrains, which are challenging for wheeled robots or humans. Quadruped robots have potential applications in search and rescue, inspection, and exploration in hazardous environments [1]. Quadruped locomotion involves complex coordination of limbs to achieve stable and adaptable movement, making it a difficult problem for traditional control methods. Reinforcement Learning (RL) has emerged as a promising approach to autonomously learning locomotion behaviors by optimizing control policies through trial and error [2]. However, while RL has achieved notable success in structured tasks like robotic manipulation [3] [4], its application to real-world quadruped locomotion remains a significant challenge.

B. Research problem statement

Despite advancements in RL, quadruped locomotion tasks suffer from key limitations, which include a lack of high-quality simulation platforms, unstable training environments due to diverse terrains, and a lack of publicly available datasets for benchmarking locomotion tasks [5] [6]. Furthermore, traditional RL paradigms, such as multi-task learning, struggle to model locomotion effectively, as the variety of terrains and tasks exceeds their assumptions of shared objectives. The above mentioned limitations and challenges raise the question: How can we enable quadruped robots to achieve robust and adaptive locomotion in real-world environments without depending on idealized multi-task RL frameworks?

C. Literature Review

Recent advancements in RL have demonstrated its potential for quadrupedal robot locomotion, with particular focus on gait generation and complex locomotion tasks such as step climbing. For instance, Peng et al. introduced a framework for learning robust locomotion policies in simulation environments, demonstrating transferable skills like walking and climbing by applying dynamics randomization to bridge the sim-to-real gap [6]. Similarly, Hwangbo et al. developed a learning pipeline that combined policy optimization with real-world constraints, achieving stable gait generation on quadruped platforms under varying terrain conditions [7]. "DeepGait" utilizes deep RL to plan and control quadrupedal gaits with minimal manual intervention, focusing on achieving smooth transitions between different gait patterns [5]. Bellicoso et al. extended DeepGait by addressing locomotion adaptability through modular architectures, allowing for the dynamic generation of gaits in unstructured environments [8].

In the context of step climbing and challenging terrain navigation, Kumar et al. explored hierarchical RL frameworks, enabling robots to break down complex tasks into sub-skills such as climbing and balancing [9], showcasing the importance of combining high-level planning with low-level control for overcoming obstacles. Additionally, Lee et al. employed RL to develop policies for real-world quadruped locomotion over uneven surfaces, with an emphasis on energy efficiency and task robustness [10].

While the literature review highlights significant progress, limitations remain in achieving real-world generalization and scalability. Most studies rely on specific simulation environments, and the transfer of policies to hardware is still a challenging aspect of RL for quadruped robots. Addressing the issues of learning robust RL locomotion policies and the bridging the sim-to-real gap requires robust frameworks capable

of handling dynamic environments and ensuring adaptability across diverse locomotion tasks.

D. Own Approach Description

This paper implements and benchmarks several reinforcement learning (RL) algorithms for continuous action and observation spaces. A simulation setup is developed to train RL policies for various locomotion tasks, specifically focusing on the Petoi Bittle quadruped robot. The policies are trained in simulation environments designed to replicate real-world scenarios, including gait generation and step climbing tasks. After training, the learned policies are transferred to real hardware to evaluate their performance on the actual robot. The sim-to-real gap is thoroughly assessed to identify discrepancies between simulation and real-world execution, with the aim of improving the adaptability and robustness of the RL policies for dynamic locomotion tasks.

E. Outline (Roadmap)

This paper is organized as follows: A background on RL algorithms for continuous action spaces is provided in Section II. Section III describes the methodology for the locomotion tasks, while Section IV include a description of the simulation implementations. Section V provides an evaluation of the RL algorithms through simulated experiments. The implications of the findings are analyzed in Section VI, followed by a summary of the research contributions and final remarks in Section VII.

II. Reinforcement Learning Algorithms

The research focuses on RL algorithms that are suitable for continuous action and observation spaces, as quadruped locomotion tasks typically involve high-dimensional, continuous state and action spaces. The subsections describe three widely used algorithms for continuous spaces.

1) Proximal Policy Optimization: Proximal Policy Optimization (PPO) [11] is one of the most popular on-policy reinforcement learning algorithms known for its simplicity and robustness. PPO aims to balance the trade-off between exploration and exploitation by updating the policy with a constraint on the size of the policy update, preventing large, unstable updates, which is achieved through a clipped objective function that limits the deviation of the policy to ensure stable learning.

$$L^{CLIP}(\theta) = \mathbb{E}_t \Big[\min \Big(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \Big) \Big] \tag{1}$$

where, $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the new policy and the old policy, $\hat{A}_t$ is the estimated advantage at time step t and ϵ is a hyperparameter that controls the size of the update (the clipping range). The clipped term $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$ ensures that the policy update does not deviate too far from the old policy, which helps maintain stability during training. PPO is well-suited for tasks with continuous action spaces, such as locomotion, and has been successfully applied in various robotic control tasks, where stability and reliability are critical.

2) Deep Deterministic Policy Gradient: Deep Deterministic Policy Gradient (DDPG) [12] is an off-policy RL algorithm designed specifically for environments with continuous action spaces. It combines the actor-critic method with deep learning to efficiently handle high-dimensional continuous spaces. DDPG utilizes two main components: an actor, which outputs the action to take based on the current state, and a critic, which evaluates the action taken by the actor. The actor-critic architecture allows DDPG to optimize policies in environments that require precise continuous control, such as robotic locomotion tasks. DDPG enhances training stability through a replay buffer, which breaks temporal correlations by sampling random batches of past experiences, and target networks, which are updated less frequently to reduce update variance. Soft updates further stabilize training by gradually adjusting target network weights, avoiding abrupt changes.

3) Twin Delayed Deep Deterministic Policy Gradient: Twin Delayed Deep Deterministic Policy Gradient (TD3) [13] is an extension of DDPG designed to improve stability and performance. TD3 addresses some of the shortcomings of DDPG by incorporating three key modifications: (1) using two critics to reduce overestimation bias, (2) delayed updates to the target network to stabilize learning, and (3) noise addition to the target action to promote exploration. The modifications make TD3 more robust and effective in continuous action space tasks, reducing the risk of instability in environments with high-dimensional, continuous state spaces like quadrupedal locomotion. TD3 is highly effective for tasks that require stable learning.

III. Methodology

Quadruped locomotion tasks are essential for enabling robots like the Petoi Bittle to navigate complex terrains with stability and efficiency. RL offers a data-driven approach to autonomously optimize movement patterns by maximizing rewards for desired behaviors. Through RL, robots can learn to generate adaptive gaits and tackle obstacles while maintaining balance. Two key locomotion tasks, gait generation and step climbing, demonstrate how RL-driven learning can refine the movement strategies of the robot, improving performance across diverse environments. The tasks rely on carefully designed reward functions to shape behavior and enhance stability, efficiency, and adaptability.

A. Gait Generation

The gait generation task for the Petoi Bittle robot involves training the robot to autonomously produce stable and efficient movement patterns, which are essential for navigating various terrains. The robot must learn to control its four legs, coordinating joint movements to maintain balance during locomotion, which is achieved using RL, where the robot receives feedback in the form of rewards based on its performance. Reward shaping plays a critical role in guiding the learning

process, as it provides intermediate feedback to encourage desirable behaviors such as smooth leg movements, stability, and efficient energy use. The reward function, inspired by [14], is designed to reward the robot for maintaining balance, achieving smooth gait transitions, and moving forward in a stable manner, with penalties applied for actions that lead to instability, excessive movement, or collisions with the environment. The total reward r_{total} is defined as:

$$r_{\text{total}} = r_{\text{movement}} - \sum r_{\text{penalties}}$$

where r_{movement} encourages forward locomotion, and the penalties discourage unstable or inefficient actions.

a) Movement Reward: The robot is rewarded for forward movement, which is proportional to its velocity in the x-direction:

$$r_{\text{movement}} = \text{FAC\_MOVEMENT} \cdot v_x$$

where v_x is the velocity of the robot in the forward (x) direction.

b) Stability Penalty: To ensure stability, a penalty is applied based on the roll and pitch angular velocities of the robot:

$$r_{\text{stability}} = \text{FAC\_STABILITY} \cdot (|\omega_{\text{roll}}| + |\omega_{\text{pitch}}|)$$

where ω_{roll} and ω_{pitch} are the roll and pitch angular velocities of the robot body.

c) Vertical Movement Penalty: A penalty is applied to discourage vertical (z-direction) movement:

$$r_{\text{z\_velocity}} = \text{FAC\_Z\_VELOCITY} \cdot |v_z|$$

where v_z is the vertical velocity of the robot body.

d) Foot Slip Penalty: To avoid foot slipping during locomotion, a penalty is applied when the feet slip:

$$r_{\text{slip}} = \text{FAC\_SLIP} \cdot \text{slip\_indicator}$$

where slip_indicator is a binary variable indicating whether foot slippage occurs.

e) Arm Contact Penalty: Crawling or arm contact with the ground is penalized:

$$r_{\text{arm\_contact}} = \text{FAC\_ARM\_CONTACT} \cdot \text{contact\_indicator}$$

where contact_indicator is a binary variable representing arm or elbow contact with the ground.

f) Smoothness Penalty: Smoothness in motion is enforced using two penalties that punish jitter and vibrational movement:

$$r_{\text{smooth\_1}} = \text{FAC\_SMOOTH\_1} \cdot |\dot{q}_{\text{current}} - \dot{q}_{\text{previous}}|$$

$$r_{\text{smooth\_2}} = \text{FAC\_SMOOTH\_2} \cdot |\ddot{q}_{\text{current}} - 2\dot{q}_{\text{previous}} + \dot{q}_{\text{before\_previous}}|$$

where $\dot{q}$ represents the joint velocity, and $\ddot{q}$ represents the joint acceleration.

g) Foot Clearance Penalty: Foot clearance during the swing phase is encouraged by penalizing low foot heights:

$$r_{\text{clearance}} = \text{FAC\_CLEARANCE} \cdot \max(0, \text{PAW\_Z\_TARGET} - z_{\text{foot}})$$

where z_{foot} is the height of the foot during the swing phase, and PAW_Z_TARGET is the desired foot clearance height.

h) Step Penalty: To avoid excessive step counts, a step penalty is applied based on the number of steps taken:

$$r_{\text{step\_penalty}} = \frac{\text{step\_counter}}{\text{PENALTY\_STEPS}}$$

B. Step Climbing

The step climbing task for the Petoi Bittle robot involves teaching the robot to navigate obstacles such as steps or uneven terrain by learning how to lift its legs appropriately and step over obstacles while maintaining stability. The step climbing task requires coordination between the limbs of the robot to execute precise and adaptive movements, ensuring that each foot is placed securely to support the body weight during the climb. Reward shaping for the step climbing task involves designing a reward function that encourages the robot to successfully lift and place each leg over the step while maintaining balance and avoiding falls. Penalties are applied for undesirable behaviors, such as stumbling, excessive effort in lifting, or collisions with the environment. Along with a few modifications to the basic reward function terms used in gait generation, additional reward and penalty terms tailored to the step climbing scenario are introduced.

a) Movement Reward: In the case of step climbing, the movement reward r_{movement} should focus not only on forward velocity but also on the ability of the robot to overcome vertical obstacles. Therefore, it should account for both horizontal (v_x) and vertical (v_z) progress:

$$r_{\text{movement}} = \text{FAC\_MOVEMENT} \cdot (v_x + \alpha \cdot v_z)$$

where α is a scaling factor that balances the reward for vertical progress with forward motion.

b) Stability Penalty: Maintaining balance is critical in step climbing, especially as the robot lifts its legs and adjusts to uneven terrain. The stability penalty can be made more sensitive to angular velocities:

$$r_{\text{stability}} = \text{FAC\_STABILITY} \cdot (\gamma_1 |\omega_{\text{roll}}| + \gamma_2 |\omega_{\text{pitch}}|)$$

where γ_1 and γ_2 are scaling factors that can be tuned to emphasize the importance of reducing roll or pitch instabilities during climbing.

c) Foot Placement Reward: To ensure that the robot places its feet securely on top of the step, encouraging accurate foot placement during the climbing phase:

$$r_{\text{foot\_placement}} = \text{FAC\_PLACEMENT} \cdot \sum_{i=1}^{4} \mathbb{I}(\text{foot\_on\_step}(i))$$

where foot_on_step(i) is a binary indicator that checks if the i-th foot is correctly placed on the step surface, and $\mathbb{I}(\cdot)$ is the indicator function that returns 1 if the condition is true and 0 otherwise.

d) Leg Lifting Reward: To encourage proper leg lifting during step climbing, we can reward the robot for lifting its feet sufficiently high to avoid stumbling:

$$r_{\text{leg\_lift}} = \text{FAC\_LEG\_LIFT} \cdot \sum_{i=1}^{4} \mathbb{I}(z_{\text{foot}(i)} > z_{\text{step}})$$

where $z_{\text{foot}(i)}$ is the height of the i-th foot during the swing phase, and z_{step} is the height of the step.

e) Body Clearance Penalty: To avoid scraping the body or getting stuck while climbing, a penalty is introduced for excessive body lowering:

$$r_{\text{body\_clearance}} = \text{FAC\_BODY\_CLEARANCE} \cdot \mathbb{I}(z_{\text{body}} < z_{\text{step}})$$

where z_{body} is the vertical position of the robot's body. The penalty is applied if the body drops below the step height during climbing.

f) Step Climbing Progress Reward: A reward can be added to encourage the robot to make forward progress while climbing the step:

$$r_{\text{climb\_progress}} = \text{FAC\_CLIMB\_PROGRESS} \cdot \Delta x$$

where Δx is the change in the x-coordinate of the robot, indicating forward progress toward completing the step climb.

IV. Implementation

This section outlines the key tools and frameworks used to develop and train RL policies for quadrupedal locomotion tasks. The frameworks provide the necessary simulation environments, physics modeling, and algorithm implementations required to design, train, and evaluate RL agents for the Petoi Bittle robot.

A. OpenAI Gym

OpenAI Gym [15] is a widely-used toolkit that provides a consistent framework for developing and evaluating RL algorithms. Gym environments expose an observation space, action space, and reward function, enabling RL agents to interact with the environment in a standardized way. OpenCat Gym environment, a custom extension of OpenAI Gym designed specifically for the Petoi Bittle quadruped robot, is used to train the RL policies by providing a controlled interface that models the dynamics and behavior of the robot. While Gym environments are typically used for training, their compatibility with various RL algorithms facilitates experimentation and comparison across different approaches.

B. PyBullet

PyBullet [16] is an open-source physics engine used to simulate real-time dynamics for robotic systems in various environments. In this work, PyBullet is utilized to implement the physics simulation within the OpenCat Gym environment by loading the URDF (Unified Robot Description Format) files of the Petoi Bittle robot. PyBullet ensures that the physical properties, such as mass, friction, and joint constraints of the robot, are accurately modeled during the training of RL policies. PyBullet handles interactions like collisions and joint movements, enabling the RL agents to experience realistic physics as they learn to control the robot. The setup helps bridge the gap between simulation and real-world implementation, ensuring that the policies are well-suited for practical applications. Figure 1 shows the Bittle robot in the simulation environment.

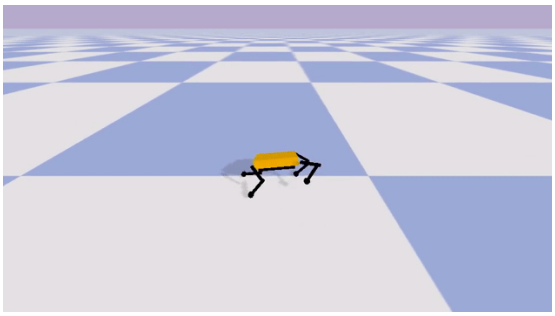

Fig. 1. Bittle robot rendered in PyBullet.

C. Stable Baselines3

Stable Baselines3 (SB3) [17] is an open-source library providing standardized implementations of RL algorithms in PyTorch. SB3 is used to implement PPO, DDPG, and TD3, each with multilayer perceptron (MLP) actor policies. SB3 enables efficient training and evaluation within the OpenCat Gym environment, leveraging its support for parallel environments and stable algorithm implementations. The use of SB3 ensures consistency and reliability in the training process, allowing for effective benchmarking of RL policies in the simulation of quadruped locomotion tasks.

V. Experimental Results

The RL algorithms PPO, DDPG and TD3 are evaluated based on episodic rewards r_{ep}, which represent the accumulated reward over 250-step episodes. For the gait generation task, the trained RL policies are further assessed based on the average gait velocity v_{x_avg}, and the accumulated stability penalty, $r_{ep\_stability}$, over the episode. For the step-climbing task, the policies are evaluated by the time taken to fully overcome the step of height 2cm (with all four legs on top), t_{step}, and the accumulated stability penalty, $r_{ep\_stability}$, over the episode.

Figures 2 and 3 show the variation of episodic rewards during training for the PPO, DDPG, and TD3 policies, corresponding to the gait generation and step-climbing tasks, respectively. TD3 exhibits the best convergence of episodic reward values, while DDPG shows poor convergence across both tasks. TD3 converges to the highest episodic rewards for the gait generation task, whereas PPO achieves the highest episodic rewards for the step-climbing task.

Table I presents the evaluation metrics for the RL policies in the gait generation and step-climbing tasks. For the gait generation task, the PPO policy achieves the fastest average gait velocity, while the TD3 policy performs best overall, with the highest episodic reward and the lowest stability penalty.

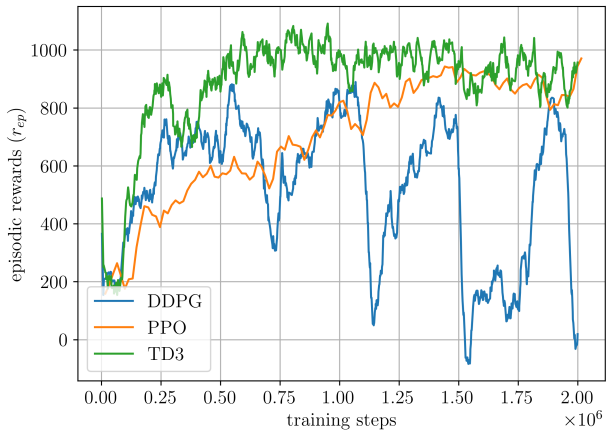

Fig. 2. Episodic rewards for DDPG, PPO and TD3 policies during training for the gait generation task.

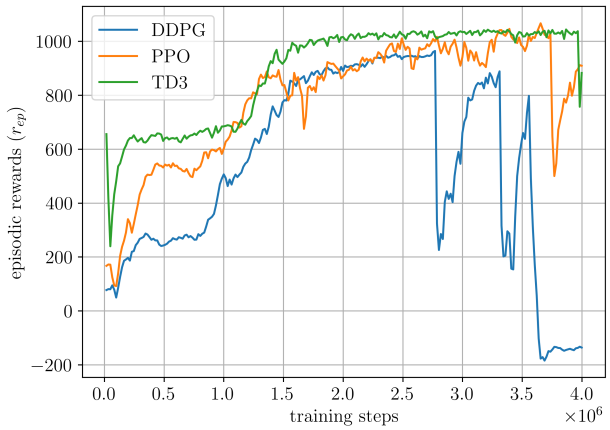

Fig. 3. Episodic rewards for DDPG, PPO and TD3 policies during training for the step-climbing task.

For the step-climbing task, the PPO policy completes the task the fastest, while the TD3 policy achieves the lowest stability penalty.

VI. Discussion

Implementing the learned RL policies on the Petoi Bittle hardware presents several challenges. A key difficulty is the unavailability of direct joint angle feedback from the hardware, which complicates real-time monitoring and control of the robot's limbs. Without precise joint angle states, it becomes challenging to ensure accurate position tracking and perform fine-grained adjustments during movement. Additionally, the hardware exhibits variable execution times when responding to target position commands, introducing delays and inconsistencies. The variability affects the synchronization of movements, making it difficult to deploy policies that rely on precise timing and coordination. The above mentioned factors underscore the gap between simulated environments, requiring further adaptation of the learned policies.

TABLE I
Evaluation of the trained RL Policies.

RL Policies		PPO	DDPG	TD3
Gait Generation	r_{ep}	978	892	**1089**
	v_{x_avg} (m/s)	**0.26**	0.14	0.24
	$r_{ep\_stability}$	4.75	2.10	**1.42**
Step-Climbing	r_{ep}	**1064**	969	1045
	t_{step} (s)	**1.46**	3.65	3.32
	$r_{ep\_stability}$	4.14	1.96	**1.66**

To implement the learned gait and step-climbing policies on the hardware, one approach is to extract the policies in terms of target joint positions, forming an array of commands for each joint of the robot. For the gait generation task, the policy can be encoded as a repeating sequence of joint positions, creating a cyclic gait pattern that continuously executes for stable movement. The Petoi Skill Composer tools can be used to integrate the gait pattern as a gait skill, which runs repeatedly to maintain locomotion. In contrast, the step-climbing task can be implemented as a discrete behavior, executed once to enable the robot to lift its legs and climb over obstacles. By leveraging the Skill Composer, the skills can be structured and controlled effectively, bridging the gap between the simulated policies and real-world hardware, ensuring the robot executes the learned tasks efficiently.

VII. Conclusion

This paper has explored RL as a tool for autonomous gait generation and step-climbing in quadruped robots. The primary objective has been to develop RL-based control strategies that can generate closed-loop locomotion patterns, addressing the limitations of existing hardcoded approaches. The study has focused on designing tailored reward functions for both gait generation and step-climbing tasks, and benchmarking various RL algorithms in simulation environments to identify the most effective strategies.

The RL policies have been evaluated based on episodic rewards and task-specific performance metrics, such as average gait velocity and stability penalties for gait generation, and time to overcome obstacles for step-climbing. Overall, the TD3 algorithm excelled in the gait generation task, while PPO performed best for the step-climbing task. Despite challenges such as variable hardware execution times and limited joint state feedback, the feasibility of implementing the learned RL policies on the Petoi Bittle hardware using target joint positions has been discussed, providing valuable insights into sim-to-real transfer challenges.

This study demonstrates the potential of RL to autonomously generate adaptive and stable quadrupedal locomotion. Future work should focus on improving hardware deployment strategies, exploring observation-based RL to address the lack of full observability, enhancing the robustness of RL policies to communication delays and execution variability, and expanding the adaptability of the policies to more complex terrains and multi-task locomotion scenarios, ultimately increasing the versatility of quadruped robots in real-world environments.

REFERENCES

[1] Marc Raibert. *Legged Robots That Balance*. MIT Press, 1986.

[2] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[3] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[4] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[5] Vasileios Tsounis, Markus Alge, Junha Lee, Farbod Farshidian, and Marco Hutter. Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):3699–3706, 2020.

[6] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810, 2018.

[7] Jemin Hwangbo, Junhyeok Lee, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 4(26):eaau5872, 2019.

[8] Dario C. Bellicoso, Vasileios Tsounis, Jemin Hwangbo, and Marco Hutter. Dynamic locomotion and whole-body control for quadrupedal robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, 2018.

[9] Animesh Garg Kumar, Priyanka Jangir, Anuj Tiwari, Viren Modh, and Mrinal Kalakrishnan. Hierarchical reinforcement learning for multi-skill locomotion. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10203–10210, 2021.

[10] Junhyeok Lee, Jemin Hwangbo, Lorenz Wellhausen, Hendrik Kolvenbach, and Marco Hutter. Learning energy-efficient quadruped locomotion over uneven terrain. *IEEE Transactions on Robotics*, 36(1):199–212, 2020.

[11] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[12] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[13] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[14] Michel Aractingi, Pierre-Alexandre Léziart, Thomas Flayols, Julien Perez, Tomi Silander, and Philippe Souères. Controlling the solo12 quadruped robot with deep reinforcement learning. *scientific Reports*, 13(1):11945, 2023.

[15] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Pieter Abbeel, Ilya Sutskever, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[16] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

[17] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.