applied sciences

MDPI

Article

Dual-Layer Reinforcement Learning for Quadruped Robot Locomotion and Speed Control in Complex Environments

Yilin Zhang, Jiayu Zeng, Huimin Sun, Honglin Sun and Kenji Hashimoto *

Graduate School of Information, Production and Systems, Waseda University, Kitakyushu 808-0135, Japan; zhangyilin@moegi.waseda.jp (Y.Z.); jiayuzeng@asagi.waseda.jp (J.Z.); hannah_sun@akane.waseda.jp (H.S.); hsun@akane.waseda.jp (H.S.)

* Correspondence: kenji.hashimoto@waseda.jp

Abstract: Walking robots have been widely applied in complex terrains due to their good terrain adaptability and trafficability. However, in some environments (such as disaster relief, field navigation, etc.), although a single strategy can adapt to various environments, it is unable to strike a balance between speed and stability. Existing control schemes like model predictive control (MPC) and traditional incremental control can manage certain environments. However, they often cannot balance speed and stability well. These methods usually rely on a single strategy and lack adaptability for dynamic adjustment to different terrains. To address this limitation, this paper proposes an innovative double-layer reinforcement learning algorithm. This algorithm combines Deep Double Q-Network (DDQN) and Proximal Policy Optimization (PPO), leveraging their complementary strengths to achieve both fast adaptation and high stability in complex terrains. This algorithm utilizes terrain information and the robot's state as observations, determines the walking speed command of the quadruped robot Unitree Go1 through DDQN, and dynamically adjusts the current walking speed in complex terrains based on the robot action control system of PPO. The speed command serves as a crucial link between the robot's perception and movement, guiding how fast the robot should walk depending on the environment and its internal state. By using DDQN, the algorithm ensures that the robot can set an appropriate speed based on what it observes, such as changes in terrain or obstacles. PPO then executes this speed, allowing the robot to navigate in real time over difficult or uneven surfaces, ensuring smooth and stable movement. Then, the proposed model is verified in detail in Isaac Gym. Wecompare the distances walked by the robot using six different control methods within 10 s. The experimental results indicate that the method proposed in this paper demonstrates excellent speed adjustment ability in complex terrains. On the designed test route, the quadruped robot Unitree Go1 can not only maintain a high walking speed but also maintain a high degree of stability when switching between different terrains. Ouralgorithm helps the robot walk 25.5 m in 10 s, outperforming other methods.

Keywords: walking robots; dual-layer reinforcement learning; proximal policy optimization; deep double Q-network; adaptive control; dynamic speed adjustment

Citation: Zhang, Y.; Zeng, J.; Sun, H.; Sun, H.; Hashimoto, K. Dual-Layer Reinforcement Learning for Quadruped Robot Locomotion and Speed Control in Complex Environments. *Appl. Sci.* **2024**, *14*, 8697. https://doi.org/10.3390/app14198697

Academic Editor: Gianluigi Ferrari

Received: 20 August 2024
Revised: 17 September 2024
Accepted: 25 September 2024
Published: 26 September 2024

1. Research Background

1.1. Background of Quadruped Robots

Quadruped animals have demonstrated extremely strong adaptability and coordinated movement ability in various natural environments [1,2]. This natural advantage has inspired the research of bionics, thereby giving birth to quadruped robots [3]. Walking robots have made significant progress in recent years, especially showing unique advantages in adaptability and trafficability in complex terrains. Compared with wheeled and tracked robots, walking robots can better cope with uneven terrains, which makes them particularly outstanding in application scenarios such as disaster relief, field navigation, and military reconnaissance [4]. Especially in the disaster relief environment, walking

robots can traverse rubble, crushed stones, and other irregular terrains and quickly reach the disaster-stricken areas, providing great convenience for rescue work [5]. In addition, in field navigation tasks, walking robots can autonomously avoid obstacles, cross gullies, and climb steep slopes to complete complex tasks [6,7].

1.2. Existing Control Methods and Limitations

To achieve efficient motion control of quadruped robots in complex terrains, many researchers have proposed various methods and strategies. These studies mainly focus on traditional control methods, bio-inspired methods, and the emerging deep reinforcement learning methods in recent years:

Model Predictive Control:

MPC is an advanced control method that achieves precise control of the system by predicting a series of future states and optimizing the current control input. MPC has been widely used in robot control, especially for the motion control of quadruped robots in complex terrains.

MPC works by predicting the system's state changes over the next few steps. It creates a dynamic model and adjusts the control input at each step to match the desired state [8]. The advantage of MPC is that it can achieve optimal control under multiple constraints and multiple objectives. However, MPC relies heavily on the accuracy of the model, and inaccurate models can lead to poor control effects [9]. Ding and others proposed a cascaded MPC framework based on a disturbance observer for the walking of humanoid robots, which integrates ankle, step, hip, and height change strategies to achieve high robustness [10]. Daneshmand and others developed a variable horizon MPC for bipedal walking control. This allows the robot to adjust its prediction horizon in real time. This improves flexibility and adaptability during walking [11].

MPC has shown significant advantages in the control of quadruped robots, especially in known and static terrains. However, when dealing with highly dynamic and unknown complex terrains, the performance of MPC is still limited by the accuracy of the model and the real-time computing [12].

Zero Moment Point (ZMP) Control:

ZMP control is a classic method for robot stability control, which ensures the balance of the robot by maintaining the position of the zero moment point. The ZMP control method is widely applied in biped and quadruped robots, especially playing a crucial role in ensuring stable walking in robots [13].

The concept of ZMP refers to a point on the robot's supporting surface where the resultant moment of all supporting forces is zero. Controlling the position of the ZMP can ensure that the robot does not tip over during walking and thus achieves stable walking. The advantage of the ZMP control method lies in its simplicity and effectiveness. The precise ZMP position can be obtained through calculation, and the robot's posture can be adjusted in real time to maintain balance [14]. Kajita and others presented a method for generating stable walking patterns for biped robots using ZMP preview control, allowing real-time adjustment and stability during locomotion [15]. Herdt and others discussed a method for online walking motion generation that uses ZMP to automatically adjust foot placement, improving the robot's ability to navigate unpredictable terrain [16].

However, when dealing with dynamic changes and complex terrains, ZMP control shows certain limitations. The ZMP control method assumes that the contact between the robot and the ground is completely known and controllable. When the terrain is complex and irregular, this assumption is often difficult to meet [17]. For example, in rubble or crushed stone terrains, the contact points between the robot's foot and the ground keep changing, making it difficult to accurately calculate the ZMP position, resulting in poor control effects. In addition, the ZMP control method mainly focuses on balance and ignores the issues of walking speed and efficiency. In some application scenarios, such as disaster relief and emergency response, robots are required to walk stably and quickly [18].

The ZMP control method has significant advantages in ensuring stable walking in robots, especially in known and static terrains. But, on dynamic and complex terrains, ZMP performance is limited. It needs improvement with other methods.

1.3. Applications of Deep Reinforcement Learning in Robot Control

Deep reinforcement learning (DRL), with its strong adaptive ability and the ability to handle high-dimensional state spaces, provides a new path for robot control [19]. Traditional control methods like model predictive control (MPC) and zero moment point (ZMP) rely on pre-defined models and struggle to adapt to unpredictable conditions, limiting their ability to balance speed and stability in real time. In contrast, a dual-layer reinforcement learning system leverages the strengths of both value-based methods, such as DDQN, and policy-based methods, such as PPO, to dynamically adjust robot behavior based on real-time feedback.

The dual-layer approach allows the robot to optimize discrete decision making (via DDQN) for high-level speed commands while using PPO to refine continuous motion adjustments for stability. This layered structure makes it more adaptable to changing environments compared to traditional methods, ensuring that the robot can navigate a variety of terrains with both efficiency and precision. Furthermore, the dual-layer system provides better generalization across different terrain types, making it more robust and scalable than single-layer approaches or classical control techniques. This flexibility and adaptability make it a superior choice for complex robot control tasks.

The Deep Q-Network (DQN) effectively addresses the poor performance of traditional Q-learning in high-dimensional state spaces by using deep neural networks to approximate the Q-value function. However, DQN has the problem of overestimation of Q-values, which can lead to instability and inefficiency of the strategy. Double DQN effectively reduces the overestimation bias of Q-values and improves the stability and efficiency of learning by separating the two processes of action selection and Q-value update [20]. This method can provide more accurate and stable action strategies, thereby enhancing the maneuverability and stability of robots in complex terrains.

As a widely applied policy optimization algorithm, PPO balances the speed and stability of policy improvement by restricting the step size of the policy update [21]. PPO adopts a truncated probability ratio objective function, effectively controlling the problems of gradient explosion and vanishing in the policy update process. In legged robots, PPO can be used to optimize the control strategy of joint positions. Through continuous interaction with the environment, the positions of robot joints can be dynamically adjusted to achieve more flexible and precise motion control. The PPO algorithm performs well in various reinforcement learning tasks and is widely used in the field of robot control [22]. PPO avoids the instability in the policy update process, enabling robots to achieve stable joint control in complex environments.

The advantages of deep reinforcement learning lie in its high flexibility and strong generalization ability, enabling it to adapt to different environments and tasks. Firstly, DRL can autonomously learn the optimal strategy through continuous interaction with the environment without relying on detailed models and prior knowledge, which is important in complex and dynamically changing environments. Secondly, DRL can handle high-dimensional state and action spaces and utilize the powerful representational ability of deep neural networks to capture complex environmental features, thereby making more precise decisions [23]. In addition, DRL shows a strong exploration ability in the face of unknown environments and can search for the optimal solution among multiple possible strategies [24]. This exploration ability gives DRL a significant advantage when dealing with tasks with high uncertainty and variability. In the gait control field, Zhang and others developed an exploration framework using DDPG and an improved SAC algorithm for biped robots. This improves stability, training speed, and walking adaptability in complex terrains [25,26]. Therefore, the method proposed in this paper uses DRL as the top layer of the control system to output instructions to the bottom layer of the DRL controller.

1.4. Hierarchical Control Methods

Deep reinforcement learning has significant advantages in high-level action selection and strategy optimization [27,28]. However, in practical applications, low-level fine torque control mostly relies on classical control methods. The PD controller achieves precise output of torque by adjusting the errors of joint position and velocity. The PD controller is simple to implement and insensitive to parameter changes, and is widely used in industrial control and robot joint control. In legged robots, the combination of the PD controller and reinforcement learning methods enables precise torque adjustment after high-level action selection and medium-level joint control, thereby achieving stable torque output and smooth motion trajectories [29].

Although a large number of studies have made significant progress in the motion control of walking robots, there are still some problems that need to be urgently solved: Existing control strategies can often only achieve a single optimization between speed and stability. In complex terrains, high-speed walking strategies can easily cause the robot to lose balance, while it is difficult for high-stability strategies to achieve rapid movement. At the same time, the adaptability and robustness of existing control algorithms in different terrains and tasks still need to be improved [30].

Especially in the face of unknown and dynamically changing environments, how to ensure stable walking in robots is a major challenge. Researchers have made significant progress in the control algorithms of walking robots [30]. The discrete-time model predictive control method is used to optimize the gait of legged robots online, enabling them to walk smoothly on uneven terrains. DRL technology is also widely used in robot control, significantly improving the adaptability and dynamic motion generation ability of quadruped robots in complex terrains [31].

Researchers have made significant progress in the control algorithms of walking robots. The discrete-time model predictive control method is used to optimize the gait of legged robots online, enabling them to walk smoothly on uneven terrains. DRL is also widely used in robot control, significantly improving the adaptability and dynamic motion generation ability of quadruped robots in complex terrains [32].

To address this challenge, this paper proposes a double-layer reinforcement learning algorithm that combines the advantages of DDQN and PPO to realize the dynamic speed adjustment of quadruped robots using terrain information and robot states. By combining Double DQN, PPO, and the PD controller to form a hierarchical control structure, efficient motion control of legged robots in complex terrains is achieved. Detailed experimental verification on the Isaac Gym simulation platform shows that this algorithm exhibits excellent speed adjustment ability and stability in complex terrains. In general, the contributions of this paper can be summarized as follows:

1. This paper innovatively proposes a double-layer reinforcement learning algorithm that combines DDQN and PPO to address the motion control problem of quadruped robots in complex terrains.
2. Using terrain information and robot states to dynamically adjust the walking speed of quadruped robots, thereby maintaining high stability and speed in complex terrains and maintaining a high degree of stability when switching between different terrains.
3. Various complex terrains, including slopes, cracks, obstacles, etc., are constructed in Isaac Gym to simulate various complex situations in the real environment. Using the quadruped robot Unitree Go1 as the test platform, the effectiveness of the algorithm is verified.

1.5. Research Content and Structure of This Paper

To systematically elaborate on the above research content, the structure of this paper is arranged as follows:

Section 1 introduces the application background and importance of walking robots in complex terrains. Section 2 elaborates on the basic concepts of the reinforcement learning methods used in this study, as well as the principles and design of the two-layer

reinforcement learning algorithm. Section 3 describes in detail the experimental design and implementation process for validating the algorithm, including using the Unitree Go1 quadruped robot to build various complex terrain environments on the Isaac Gym simulation platform for testing. Section 4 summarizes the full text and conducts a result analysis.

2. Double-Layer Reinforcement Learning Algorithm Design

2.1. Basic Concepts of RL

Reinforcement learning (RL) is a machine learning method that learns the optimal policy through interaction with the environment to maximize cumulative rewards. The core concepts of RL include state, action, reward, policy, value function, and environment. The state represents the specific situation of the environment at a certain moment, the action is the behavior that the agent can take in a specific state, and the reward is the immediate score that the environment feeds back after the agent takes an action. The policy determines the rule of the action taken by the agent in each state, and the value function estimates the value of a certain state or state–action pair.

The RL problem is usually modeled as a Markov Decision Process (MDP), which includes a state space S, an action space A, a state transition probability P, a reward function R, and a discount factor γ. At each time step, the agent starts from state $s \in S$ and selects action $a \in A$; the environment transitions to a new state $s' \in S$ based on the state transition probability $P(s'|s,a)$ and provides a reward $r = R(s,a)$. The objective of the MDP is to find the optimal policy π^* that maximizes the expected cumulative reward, expressed as follows:

$$R_t = \sum_{t=0}^{\infty} \gamma^t r_t \tag{1}$$

where $\gamma \in [0,1]$ is the discount factor, and r_t is the reward at time step t.

The value iteration method is a common policy in RL. Q-learning is a model-free value iteration method that finds the optimal policy by iteratively updating the state-action value function $Q(s,a)$. DQN approximates the Q-value function by using a deep neural network to solve the problem of traditional Q-learning in high-dimensional state spaces, but DQN has the problem of overestimation of Q-values. Double Q-learning reduces the overestimation of Q-values. It improves learning stability and efficiency by separating action selection and Q-value updates.

2.2. Policy Gradient Methods

Policy gradient methods are a type of algorithm that solves reinforcement learning problems by directly optimizing the policy function π. Unlike value iteration methods, policy gradient methods do not estimate the value function directly but instead optimize the policy itself to maximize the expected cumulative reward. These methods are widely used for reinforcement learning tasks with continuous action spaces and high-dimensional state spaces.

The basic idea of policy gradients is to compute the gradient ascent to directly optimize the policy parameters, maximizing the cumulative reward. The core of policy gradient methods is the policy gradient theorem, which provides the method for computing the gradient of the expected cumulative reward with respect to the policy parameters.

Let $\pi_\theta(a|s)$ be a parameterized probability distribution representing the probability of taking action a in state s, with parameters θ. The policy gradient theorem gives the gradient of the expected cumulative reward with respect to the policy parameters θ:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a) \tag{2}$$

In Formula (2), the term $\nabla_\theta J(\theta)$ represents the gradient of the expected cumulative reward $J(\theta)$ with respect to the policy parameters θ, indicating how to adjust θ to increase

the reward. The expectation $\mathbb{E}_{\pi_\theta}$ takes the average over all possible state–action pairs (s, a) that could occur under the current policy, capturing the average behavior of the policy. The term $\nabla_\theta \log \pi_\theta(a|s)$ is the gradient of the log probability of taking action a in state s, which measures how sensitive the action probabilities are to changes in θ. This helps in simplifying the gradient computation for policy updates. Finally, $Q^{\pi_\theta}(s, a)$ is the state–action value function, representing the expected cumulative reward from state s after taking action a and following the policy π_θ. This quantifies how beneficial a certain action is in a given state. Together, this formula directs the policy update towards actions that both are probable and lead to high rewards.

The steps of a policy gradient method are illustrated as Algorithm 1:

Algorithm 1: Policy Gradient Method

1: **Initialize** policy parameters θ and other related parameters.
2: **Sampling:** Interact with the environment using the current policy π_θ to generate state–action–reward sequences $(s_0, a_0, r_0), (s_1, a_1, r_1), \ldots$.
3: **Compute Gradients:** Compute the gradient of the expected cumulative reward with respect to the policy parameters θ according to the policy gradient theorem:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)]$$

4: **Update Policy:** Update the policy parameters θ using gradient ascent:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

where α is the learning rate.

PPO is a widely used policy gradient algorithm. PPO balances the speed and stability of policy improvement by limiting the step size of policy updates, addressing the instability issues often encountered in policy gradient methods. The core idea of PPO is to introduce a clipped probability ratio objective function, which limits the extent of change between the new and old policies, ensuring stable updates.

PPO optimizes the following objective function:

$$L^{CLIP}(\theta) = \mathbb{E}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right] \tag{3}$$

In Formula (3), $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ represents the probability ratio of the new policy to the old policy, indicating how the new policy differs from the previous one. The term $\hat{A}_t$ is the estimated advantage function, which helps determine how much better a particular action is compared to the expected outcome. The clipping operation $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$ limits the probability ratio to a range between $1-\epsilon$ and $1+\epsilon$, where ϵ is a hyperparameter that controls how much the policy can change in each update step. This ensures that the policy updates do not become too large, helping stabilize learning by preventing excessive updates.

By optimizing the above objective function, PPO effectively controls gradient explosion and vanishing problems during the policy update process. It also balance the speed and stability of policy improvement, enabling the robot to achieve stable action control in complex environments.

2.3. The Design Structure of the Double-Layer Structure

The double-layer reinforcement learning algorithm framework proposed in this paper combines DDQN and PPO to improve the speed adjustment ability and stability of the quadruped robot Unitree Go1 in complex terrains. In the initial stage of the algorithm, it is necessary to initialize the parameters of the DDQN and PPO models, including setting the terrain information and the state of the robot as observation variables, and initializing the

robot environment and the simulation platform Isaac Gym. Then, the algorithm collects the current terrain information and the state of the robot, such as position, speed, and posture, and inputs this data into the DDQN network to generate the walking speed command. The speed command serves as a high-level instruction, determining how fast the robot should move based on its environment and internal status. It provides a target speed that ensures the robot maintains balance and avoids obstacles, while optimizing its overall movement efficiency. Subsequently, the PPO model takes this speed command, along with the current observation information, to generate specific robot actions, such as joint angles and gaits. These actions control the precise movement of the robot's legs and joints. The PPO model continuously refines and adjusts these actions through policy optimization, helping the robot adapt to uneven or complex terrain while maintaining stability and smooth locomotion. After the robot executes the actions generated by PPO, the system updates its state, and collects new observation information and feeds it back to the DDQN and PPO models for further optimization and adjustment. Through training and update, the DDQN algorithm updates the Q value to optimize the walking speed strategy, and the PPO algorithm updates the policy network to optimize the action control strategy. The above steps are executed continuously in a loop until the model converges or reaches the predetermined number of training times. The step can be illustrated as Algorithm 2.

Algorithm 2: Dual-Layer Reinforcement Learning Algorithm

Initialization:
Initialize parameters for DDQN and PPO models
Set terrain information and robot state as observations
Initialize the environment and simulation platform (Isaac Gym) for Unitree Go1
loop
    Terrain Information and State Observation:
    Collect current terrain information and robot state (position, speed, posture, etc.)
    Input observation information into the DDQN network
    DDQN Speed Command Generation:
    DDQN network outputs a walking speed command based on the input observations
    PPO Action Control:
    PPO model generates specific robot actions (joint angles, gait, etc.) based on the speed command and current observations
    PPO model continuously adjusts actions through policy optimization to adapt to complex terrains and maintain stability
    Execute Action and Update State:
    Execute actions generated by PPO and update the robot's state in the environment
    Collect new observations and feed back to DDQN and PPO models
    Training and Updating:
    Use DDQN algorithm to update Q-values and optimize the walking speed strategy
    Use PPO algorithm to update the policy network and optimize the action control strategy
    if convergence or maximum training iterations reached **then**
        Model Validation:
        Perform detailed validation of the trained model in Isaac Gym
        Evaluate the model's speed adjustment ability and stability in various terrains
        Exit loop
    end if
end loop
Performance Evaluation:
Assess the model's performance, ensuring high walking speed and stability across different terrains

The schematic diagram of the method proposed in this paper can be represented by Figure 1. The dual-layer structure offers several advantages by separating high-level decision making from low-level control, improving both learning efficiency and motion precision. In the upper layer, DDQN reduces the risk of overestimating action values, allowing for more accurate decisions in complex environments, while PPO ensures stable policy updates, leading to smoother transitions in control actions. This setup enhances the system's ability to learn optimal strategies without being overwhelmed by fine-grained control details. Meanwhile, the lower layer, governed by the PD controller, precisely converts these high-level commands into torque outputs, ensuring the robot's movements are accurate and responsive. By decoupling strategic decision making from mechanical control, the dual-layer structure allows for more effective learning and adaptability, especially in dynamic and unpredictable environments.

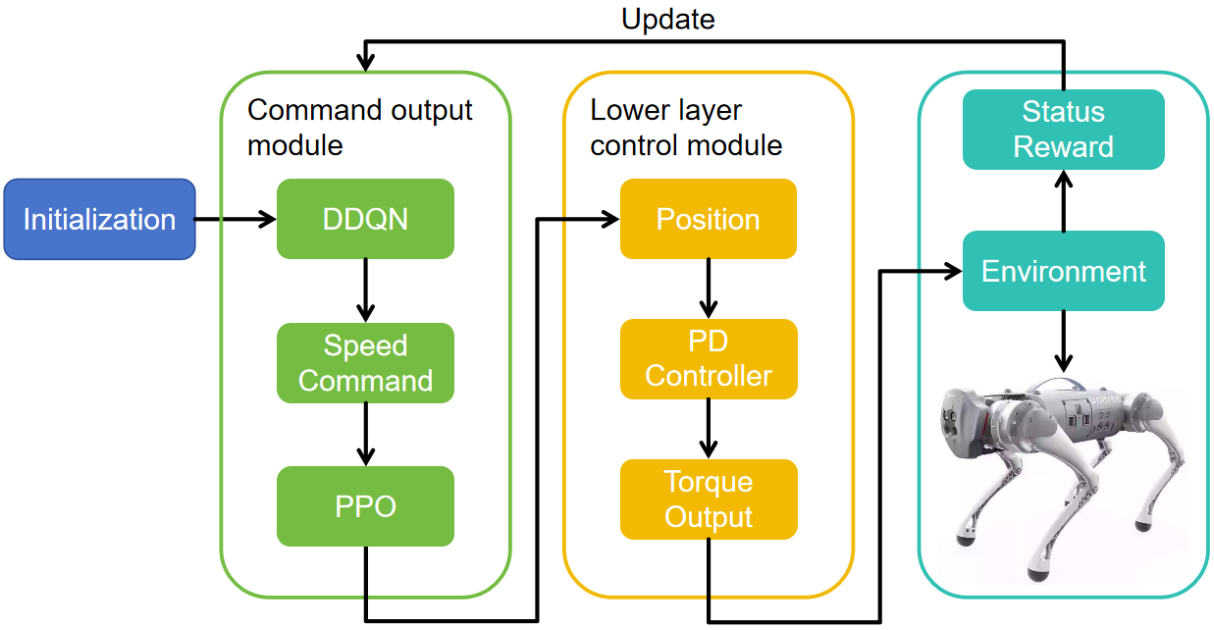

Figure 1. Architecture of the robot control system, where the command output module generates speed commands, the lower-layer control module controls the robot's movement based on these commands, and the control strategy is continuously optimized through interaction with the environment.

2.4. Terrain Information and Robot State Observation

In reinforcement learning tasks, choosing the right observation vectors is very important. These vectors help the robot understand both its environment and its internal state. They act as the robot's "sensors", giving the model real-time information about what is happening. For example, the robot needs to know its speed, position, and orientation, as well as details about the terrain or any obstacles it might face. This information allows the learning model to make better decisions. Without good observations, the robot would not know how to adjust its actions to achieve its goals. Therefore, clear and accurate observation data ensure the robot can learn quickly, adapt to new situations, and perform tasks successfully in complex environments.

In reinforcement learning tasks, the reasonable selection of robot state observation is very important. In this paper, the observation information of Unitree Go1 is divided into the following types: velocity v and acceleration a information obtained through the inertial measurement unit, torque T, position p, and joint angular velocity ω collected by the joints, and surrounding terrain height h information extracted through the depth camera. These pieces of information are integrated into a 235-dimensional vector at each time point. The observations specifically include the following: IMU data include linear acceleration a_{linear}, angular velocity $\omega_{angular}$, and the calculated attitude information (pitch angle θ_{pitch}, roll

angle θ_{roll}, and yaw angle θ_{yaw}). Linear acceleration a_{linear} is measured by the triaxial accelerometer, which can capture the acceleration changes of the robot in the x, y, and z directions. The angular velocity $\omega_{angular}$ is measured by the triaxial gyroscope, recording the rotational speed of the robot in space. Joint information comes from the output of multiple joint sensors, including joint position sensors, torque sensors, and angular velocity sensors. The joint position sensor records the absolute and relative angular positions of each joint. The torque sensor measures the torque T applied by the joint, helping to evaluate the load and resistance of each joint during movement. The joint angular velocity sensor provides real-time data of the rotational speed of the joint. In addition, the depth camera is responsible for capturing the three-dimensional environmental information around the robot, generating point cloud data and height maps. The depth camera measures the reflection time or deformation of light to construct the depth information of the environment. After processing these data, the height distribution of the terrain can be obtained, and features such as the ground, obstacles, and slopes can be identified. Through the real-time fusion of the above multi-source sensor data, the robot can accurately perceive its own state and the surrounding environment, thereby performing stable and efficient movement in complex terrains. In this study, in the process of determining the robot's speed command based on DDQN and the process of generating the robot's actions based on PPO, the robot's observations use all the observation information.

2.5. Speed Command Generation Based on DDQN

To achieve efficient motion control of the quadruped robot in complex terrains, this paper proposes a speed command generation method based on DDQN. In complex terrains, a single speed command cannot maintain good terrain adaptability. In rough terrain environments, an overly fast speed command will make the robot unable to maintain balance, while in flat terrain environments, an overly slow speed command will make the robot walk slowly. Therefore, it is necessary to dynamically adjust the speed, and DDQN can effectively judge the terrain and output corresponding commands. The Q-network structure in the structure is shown in Figure 2:

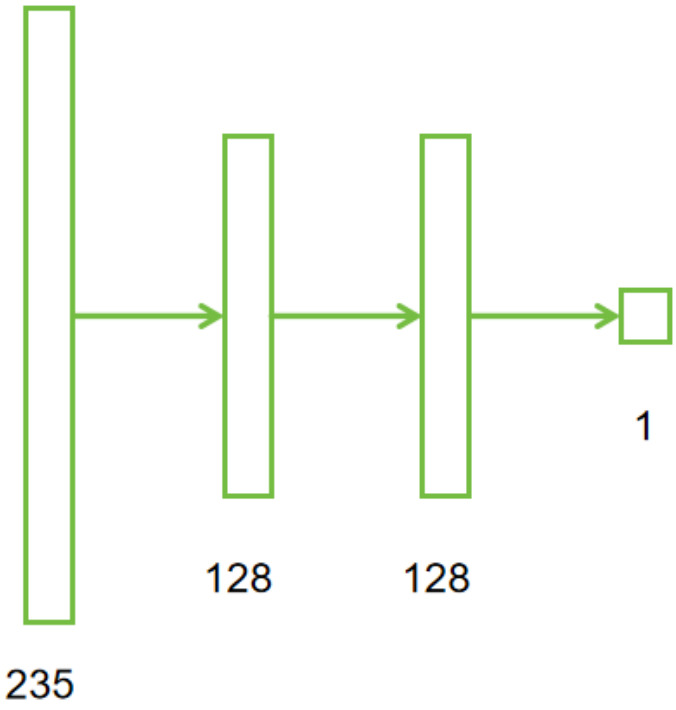

Figure 2. The structure of the speed command generation network, where the input is 235-dimensional observation information and the output is the speed command. The numbers in the figure are the quantities of neurons in each layer.

In previous verifications, even on flat ground, the robot has difficulty walking at a speed exceeding 3.5 m/s. Therefore, the output of the network is four discrete speeds: 0.5, 1.5, 2.5, and 3.5, with a unit of m/s. Since the task of the DDQN model is to maintain the fastest speed as possible, the reward function of this task is set as Equation (4), where the reward is designed to encourage actions that maximize speed while penalizing unstable or inefficient movements. This structure helps the DDQN model prioritize behaviors that achieve high velocity while maintaining balance and control, ensuring the robot performs optimally in dynamic environments.

$$R = \exp\left(-\frac{\sum_{i=1}^{2}(3.5 - v_{i,\text{actual}})^2}{\sigma_{\text{tracking}}}\right) \tag{4}$$

where v_x is the horizontal speed of the robot walking along its own orientation, and σ_{tracking} is a parameter that determines the reward difference corresponding to the difference in different speeds.

Taking the speed command as the input, the corresponding action planning is carried out using the control module based on PPO. In each training step, the robot selects the appropriate speed command by perceiving the changes in the environment and updates the Q-value function according to the actual walking result. To avoid the model overfitting to a specific terrain during training, this paper uses a variety of different terrain types for mixed training during training, including flat ground, slopes, steps, and complex gravel roads, etc.

2.6. Reward Function

In reinforcement learning, the reward function is one of the core elements guiding the agent's learning process. It provides positive or negative feedback, helping the agent make optimal decisions in its environment. The reward function design is key in this algorithm. It directly affects the robot's stable walking in complex terrains. Since different factors contribute variably to the robot's overall performance, a series of weighted reward functions are used to guide the robot's behavior in both positive and negative ways [33].

To determine the most effective combination of weights for these reward functions, we employed the grid search method, a widely used hyperparameter tuning technique. Grid search involves defining a grid of possible values for each weight and systematically evaluating the performance of the model for every possible combination of these values. In our case, we defined a range of candidate values for each reward component, such as penalties for instability or incentives for speed, and tested the algorithm across these values.

Each combination was evaluated by running simulations in various terrains, and the overall performance was measured by metrics such as walking stability, terrain adaptability, and motion efficiency. By comparing the results for different weight combinations, we identified the optimal set of weights that maximized performance across all terrains. This approach ensured that the robot could balance speed and stability under diverse conditions, leading to more reliable control. The specific reward functions and their associated weights are shown in Table 1.

2.7. Motion Trajectory Generation Based on PPO

To achieve the efficient motion planning of quadruped robots in complex terrains, this study further adopts the motion trajectory generation method based on PPO. The input of the PPO model is the same as that of DDQN, which is 235-dimensional observation information. In order to train the robot to walk smoothly under different terrain complexities, this study selects four different terrain parameters for training in four different terrains, as shown in Table 2.

Table 1. Reward function and weight factors.

Reward Term	Weight	Function
tracking_lin_vel	1.0	$\exp\left(-\frac{(\mathbf{v}_{\text{cmd}}^{xy}-\mathbf{v}_{\text{base}}^{xy})^2}{\sigma_{\text{track}}}\right)$
tracking_ang_vel	0.5	$\exp\left(-\frac{(\omega_{\text{cmd}}^{z}-\omega_{\text{base}}^{z})^2}{\sigma_{\text{track}}}\right)$
lin_vel_z	−2.0	$v_{\text{base}}^{z} \cdot v_{\text{base}}^{z}$
ang_vel_xy	−0.05	$\sum\left(\omega_{\text{base}}^{x}\right)^2 + \left(\omega_{\text{base}}^{y}\right)^2$
torques	−0.00001	$\sum \tau^2$
dof_acc	-2.5×10^{-7}	$\sum\left(\frac{\dot{q}_{\text{prev}}-\dot{q}}{\Delta t}\right)^2$
feet_air_time	1.0	$\sum(t_{\text{air}} - 0.5) \times f_{\text{contact}} \times (\|\mathbf{v}_{\text{cmd}}^{xy}\| > 0.1)$
collision	−1.0	$\sum(1 \times (\|\mathbf{F}_{\text{contact}}\| > 0.1))$
action_rate	−0.01	$\sum\left(\mathbf{a}_{\text{prev}} - \mathbf{a}\right)^2$

Table 2. Model training parameters.

Speed (m/s)	Random Speed Command (m/s)	Terrain Slope	Platform Size (m)	Obstacles Quantity	Obstacles Size (m)	Step Size (m)	Step Gap (m)
0.5	[−1, 1]	0.39	1	20	0.5–1	0.30	0.2
1.5	[−1, 2]	0.30	1	20	0.3–0.8	0.56	0.1
2.5	[−1, 3]	0.15	1	20	0.2–0.5	1.00	0.1
3.5	[−1, 3.5]	0.02	1	10	0.1–0.2	2.00	0.1

Four different models are used, corresponding to four different speeds, respectively. When walking at a slow speed, the corresponding model mainly focuses on maintaining stability. Therefore, the reward function of the model is adjusted to make it pay more attention to the balance ability of the robot in rough terrains rather than pursuing speed. When walking at medium and fast speeds, the corresponding models will appropriately balance the requirements of stability and speed. When walking at high speed, the reward function of the model is more inclined to an increase in speed, while considering the balance requirements at high speed. Therefore, at different speeds, the corresponding PPO model will generate the optimal motion trajectory that meets the current speed requirements.

3. Experiment

Isaac Gym is a physics engine and development platform for robotics simulation and reinforcement learning developed by NVIDIA [34]. It combines high-performance physical simulation and large-scale reinforcement learning capabilities, enabling developers to train and simulate robots in a unified environment. Its adopted NVIDIA GPU acceleration technology enables large-scale parallel physical simulation. This allows developers to train tens of thousands of robots simultaneously. In this study, Pytorch (https://pytorch.org/) and Isaac Gym (https://developer.nvidia.com/isaac-gym) are combined, and Pytorch is used to build neural networks. Ten thousand robots are trained simultaneously during training. Training is conducted on an Intel Core i9-11900 CPU and an Nvidia GeForce RTX 3080 GPU for 3000 episodes, and each round of training takes approximately 150 min. In the speed command generation module, 500 episodes are conducted for each training, and the training takes approximately 40 min. During the training process, the GPU utilization rate remains at approximately 80%, and the total memory usage averages around 10 GB.

In this study, the specific parameters of the quadruped robot Unitree Go1 used are shown in Table 3:

Table 3. Joint limits and dimensions.

Joint	Dimensions (m)	Lower Limit (rad)	Upper Limit (rad)
Hip Joint (Front Right)	0.28 × 0.16 × 0.11	−1.047	1.047
Thigh Joint (Front Right)	0.213 × 0.0245 × 0.034	−0.663	2.967
Calf Joint (Front Right)	0.213 × 0.016 × 0.016	−2.723	−0.838
Hip Joint (Front Left)	Radius: 0.01	−1.047	1.047
Thigh Joint (Front Left)	0.213 × 0.0245 × 0.034	−0.663	2.967
Calf Joint (Front Left)	0.213 × 0.016 × 0.016	−2.723	−0.838
Hip Joint (Back Right)	Radius: 0.01	−1.047	1.047
Thigh Joint (Back Right)	0.213 × 0.0245 × 0.034	−0.663	2.967
Calf Joint (Back Right)	0.213 × 0.016 × 0.016	−2.723	−0.838
Hip Joint (Back Left)	Radius: 0.01	−1.047	1.047
Thigh Joint (Back Left)	0.213 × 0.0245 × 0.034	−0.663	2.967
Calf Joint (Back Left)	0.213 × 0.016 × 0.016	−2.723	−0.838

During the training process, the duration of each training episode is 10 s, while the frequency of the command output structure based on DDQN is 2.5 Hz. The output frequency of the position output structure based on PPO is 50 Hz, meaning that the speed command is re-determined every 20 position outputs. At the same time, between every two adjacent positions, the PD controller performs five torque control outputs. This study uses a 7 × 7 terrain grid, with each terrain having a size of 8 × 8 m. The entire terrain layout is shown in the upper part of Figure 3. The specific size parameters of the terrain are obtained through random sampling of the parameters used in training the four different models. These four types of terrains, as illustrated in the lower part of Figure 3, are concave slope, convex slope, cuboid obstacle, and stepping with gap, with proportions of 2:3:2:3. At the beginning of each training cycle, the robot's initial position is randomly set at the center of one of the terrains, and it moves forward in the direction it faces.

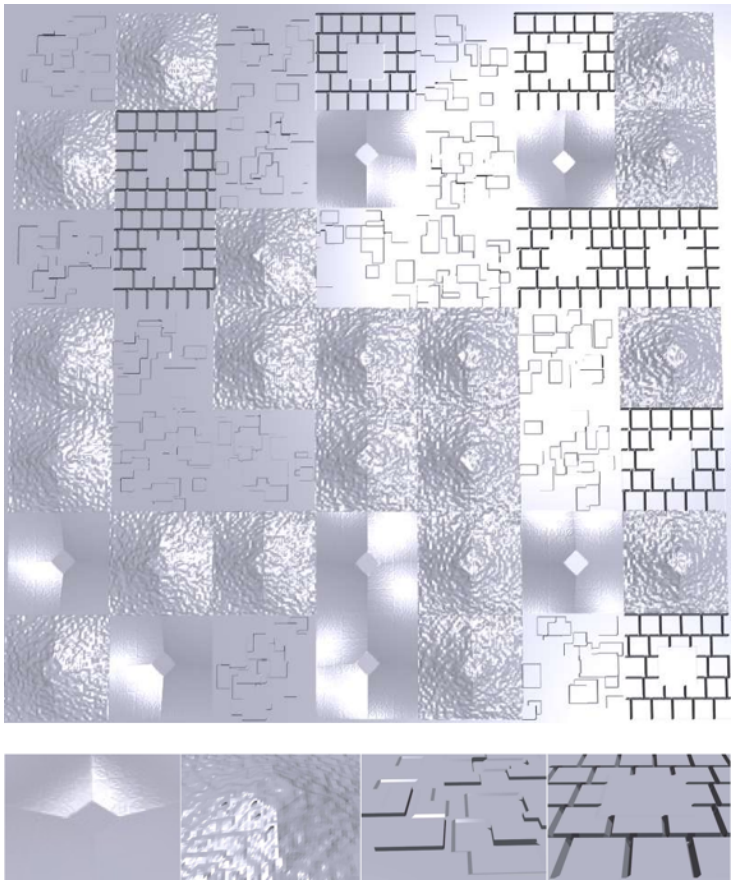

Figure 3. The entire terrain layout used for training (**upper**) and the specific terrain types (**lower**).

Experimental Results and Analysis

To verify the effectiveness of the method proposed in this study, this study chose to let the robot walk forward for 10 s and compared the effects of different models based on the distance the robot walked forward. During the training process, the different effects when using a single model and using four specially designed models for training were compared. The single model selected was designed for a speed of 2.5 m/s during training. No conditions for stopping training were set, and the training would end after 500 episodes. The characteristics of the reward changing with the episode during training are shown in the Figure 4. From these figures, it is observable that, as training progresses, the agents gradually receive increasing rewards, indicating that the agents are progressively optimizing their adaptation to the environment through learning.

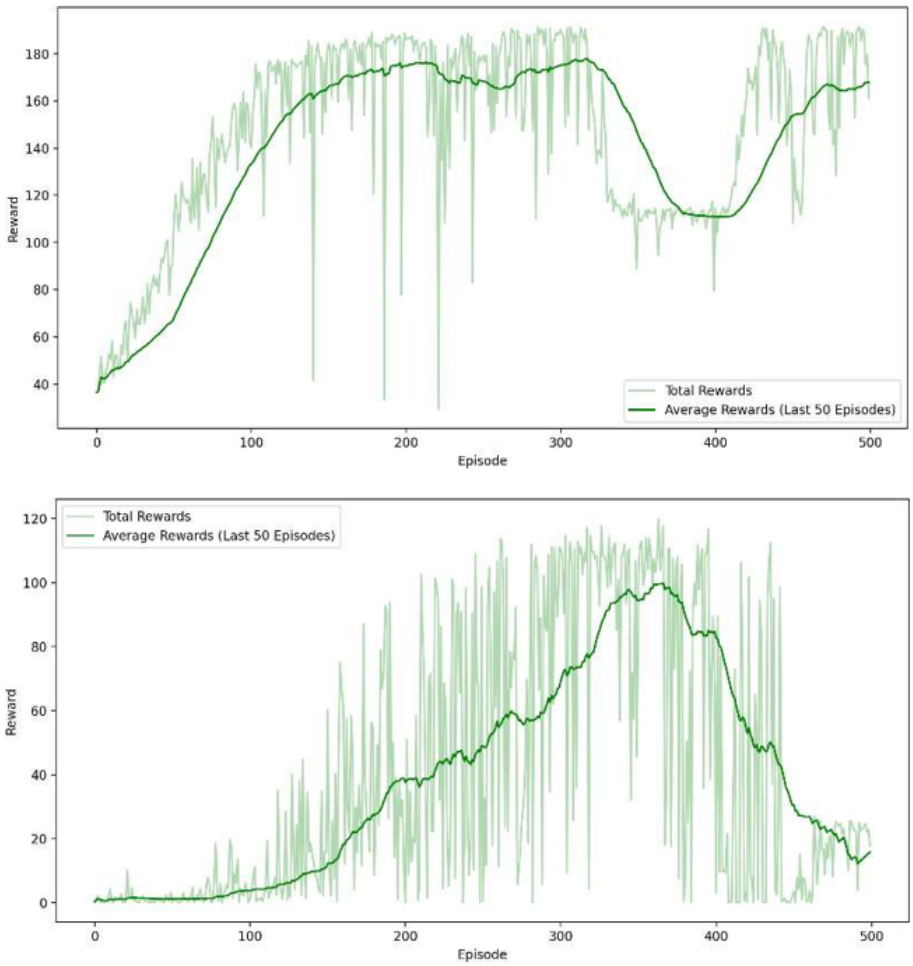

Figure 4. Experimental result comparison. The above figure shows the results of the method proposed in this paper, and the lower figure shows the training results of the single model at 2.5 m/s. The horizontal axis in these figures represents the number of training rounds, while the vertical axis shows the average reward obtained by the agents in each episode. The darker part in the figures is the moving average reward in 50 episodes, and the lighter part is the reward of each episode. Higher rewards represent that the robot may walk at a faster speed.

At the same time, we conducted 10 experiments in random terrains and took the average, and compared the distances walked by the robot using six different methods within 10 s. The results are shown in the table. It can be seen from the table that the proposed method takes into account the fast moving speed while maintaining stability by adjusting the walking speed in different terrains. The specific results are shown in Table 4.

1. Using a fixed speed command of 0.5 m/s.

2. Using a fixed speed command of 1.5 m/s.
3. Using a fixed speed command of 2.5 m/s.
4. Using a fixed speed command of 3.5 m/s.
5. Using a 2.5 m/s motion model and dynamically adjusting the speed using the speed generation module.
6. The double-layer reinforcement learning algorithm proposed in this paper.

Table 4. Comparison of different methods for controlling robot's 10-second movement distance.

Method	Control Method Description	Distance (m)
1	Using 0.5 m/s constant speed command and corresponding model	4.8
2	Using 1.5 m/s constant speed command and corresponding model	13.8
3	Using 2.5 m/s constant speed command and corresponding model	21.1
4	Using 3.5 m/s constant speed command and corresponding model	20.5
5	Using 2.5 m/s motion model with speed generation module to dynamically adjust speed command	24.2
6	Proposed dual-layer reinforcement learning algorithm	25.5

To more intuitively reflect the advantages of the dynamic adjustment method proposed in this paper, this paper compares the gait differences between using the fixed speed method of 3.5 m/s and the method proposed in this paper when climbing an obstacle with a height of 0.5 m. The result is shown in Figure 5. It can be clearly seen that the adaptability of the method proposed in this paper to complex terrains is much better than the method of choosing a fixed speed, and it has the longest walking distance among all the methods.

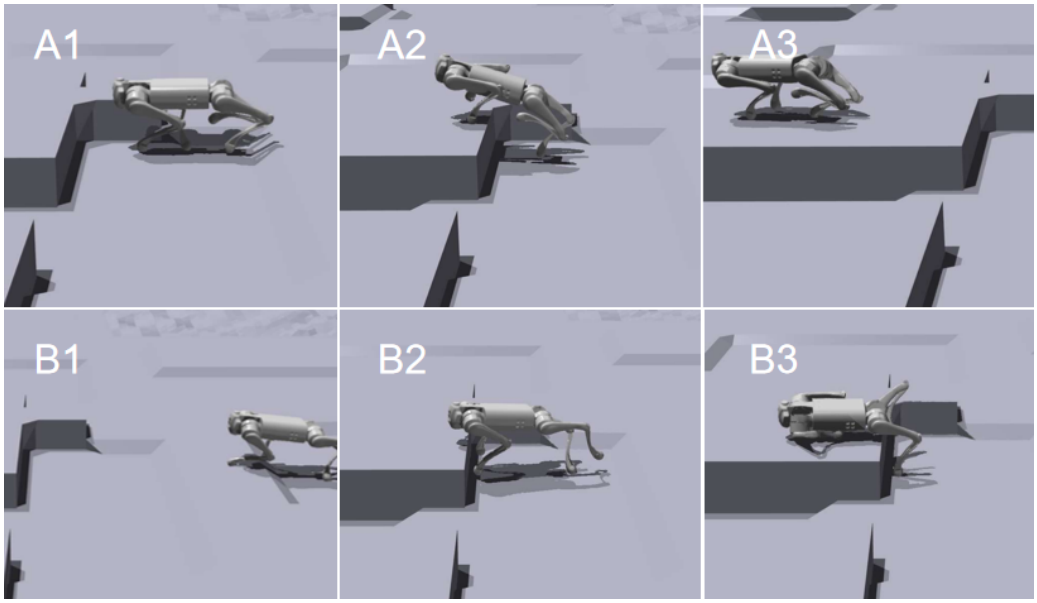

Figure 5. Gait differences between using the fixed speed method of 3.5 m/s and the method proposed in this paper. Both methods are tested on the same terrain and the same step obstacle. The two rows of pictures show the robot walking methods based on the two models, respectively, and the process of each model is from number 1 to 3. **A1**–**A3** is the adaptive speed adjustment method proposed in this paper, and **B**1–**B3** is the fixed speed method of 3.5 m/s. In the method proposed in this paper, the robot walks at a speed of 3.5 m/s before and after climbing the obstacle, and the speed is adjusted to 1.5 m/s when climbing the obstacle in order to maintain stability and pass the obstacle at a high speed. In the fixed speed method, the robot is tripped by the step and loses balance due to the excessive moving speed.

4. Discussion

This paper proposes a double-layer control algorithm based on deep reinforcement learning to address the motion control problem of quadruped robots in complex terrains,

especially the balance between speed and stability. The key findings and contributions of this study are summarized as follows:

1. The algorithm combines DDQN and PPO to dynamically adjust the walking speed based on real-time terrain information and robot states, ensuring efficient and stable movement across different terrains.
2. A detailed design of the double-layer reinforcement learning algorithm is provided, including the speed command generation and motion trajectory control modules.
3. Experimental results on the NVIDIA Isaac Gym simulation platform demonstrate that the algorithm significantly improves both the walking distance and stability of the robot, particularly on dynamic and irregular terrains.
4. The innovative combination of DDQN and PPO presents a widely adaptable control method, validated through comprehensive experiments.
5. Future research can focus on optimizing the real-time performance of the algorithm, expanding its application to multi-task scenarios, and testing its effectiveness in real-world environments.

Author Contributions: Conceptualization, Y.Z.; methodology, Y.Z.; software, Y.Z. and J.Z.; validation, Y.Z. and H.S. (Huimin Sun); formal analysis, J.Z.; investigation, Y.Z.; resources, K.H.; data curation, Y.Z.; writing—original draft preparation, Y.Z.; writing—review and editing, H.S. (Honglin Sun); visualization, Y.Z.; supervision, K.H.; project administration, K.H.; funding acquisition, K.H. All authors have read and agreed to the published version of the manuscript.

Funding: This study was conducted with the support of the Information, Production and Systems Research Center, Waseda University; Future Robotics Organization, Waseda University, and as part of the humanoid project at the Humanoid Robotics Institute, Waseda University. This work was supported by JSPS KAKENHI Grant Number JP21H05055 and a Waseda University Grant for Special Research Projects (Project number: 2024C-188). This work was also supported by JST SPRING, Grant Number JPMJSP2128.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DDQN	Deep Double Q-Network
PPO	Proximal Policy Optimization
MPC	Model Predictive Control
ZMP	Zero Moment Point
DRL	Deep Reinforcement Learning
DQN	Deep Q-Network
MDP	Markov Decision Process
RL	Reinforcement Learning

References

1. Fukuhara, A.; Gunji, M.; Masuda, Y. Comparative anatomy of quadruped robots and animals: A review. *Adv. Robot.* **2022**, *36*, 612–630. [CrossRef]
2. Raibert, M.; Blankespoor, K.; Nelson, G.; Playter, R. Bigdog, the rough-terrain quadruped robot. *Ifac Proc. Vol.* **2008**, *41*, 10822–10825. [CrossRef]
3. Semini, C.; Barasuol, V.; Goldsmith, J.; Frigerio, M.; Focchi, M.; Gao, Y.; Caldwell, D.G. Design of the hydraulically actuated, torque-controlled quadruped robot HyQ2Max. *Ieee/Asme Trans. Mechatron.* **2016**, *22*, 635–646. [CrossRef]

4. Zhu, J.; Rong, C.; Rosendo, A. Probabilistic inferences on quadruped robots: An experimental comparison. In Proceedings of the 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), Dali, China, 6–8 December 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1892–1898.
5. Chan, C.Y.; Liu, Y.C. Towards a walking, turning, and jumping quadruped robot with compliant mechanisms. In Proceedings of the 2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), Banff, Canada, 12–15 July 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 614–620.
6. Ma, Z.l.; Ma, Q.Y.; Lyu, R.J.; Wang, J.M. Running analysis of quadruped robot with flexible spine. *J. Northeast. Univ. (Natural Sci.)* **2020**, *41*, 113.
7. Hutter, M.; Gehring, C.; Jud, D.; Lauber, A.; Bellicoso, C.D.; Tsounis, V.; Hwangbo, J.; Bodie, K.; Fankhauser, P.; Bloesch, M.; et al. Anymal-a highly mobile and dynamic quadrupedal robot. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Republic of Korea, 9–14 October 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 38–44.
8. Neunert, M.; Stäuble, M.; Giftthaler, M.; Bellicoso, C.D.; Carius, J.; Gehring, C.; Hutter, M.; Buchli, J. Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robot. Autom. Lett.* **2018**, *3*, 1458–1465. [CrossRef]
9. Garcia, C.E.; Prett, D.M.; Morari, M. Model predictive control: Theory and practice—A survey. *Automatica* **1989**, *25*, 335–348. [CrossRef]
10. Ding, J.; Han, L.; Ge, L.; Liu, Y.; Pang, J. Robust locomotion exploiting multiple balance strategies: An observer-based cascaded model predictive control approach. *Ieee/Asme Trans. Mechatron.* **2022**, *27*, 2089–2097. [CrossRef]
11. Daneshmand, E.; Khadiv, M.; Grimminger, F.; Righetti, L. Variable horizon mpc with swing foot dynamics for bipedal walking control. *IEEE Robot. Autom. Lett.* **2021**, *6*, 2349–2356. [CrossRef]
12. Kouvaritakis, B.; Cannon, M. Model Predictive Control. *Switz. Springer Int. Publ.* **2016**, *38*, 13–56.
13. Hirai, K.; Hirose, M.; Haikawa, Y.; Takenaka, T. The development of Honda humanoid robot. In *1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*; IEEE: Piscataway, NJ, USA, 1998; Volume 2, pp. 1321–1326.
14. Kajita, S.; Tani, K. Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode. In Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, CA, USA, 9–11 April 1991; IEEE Computer Society: Piscataway, NJ, USA, 1991; pp. 1405–1406.
15. Kajita, S.; Kanehiro, F.; Kaneko, K.; Fujiwara, K.; Harada, K.; Yokoi, K.; Hirukawa, H. Biped walking pattern generation by using preview control of zero-moment point. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*; IEEE: Piscataway, NJ, USA, 2003; Volume 2, pp. 1620–1626.
16. Herdt, A.; Diedam, H.; Wieber, P.B.; Dimitrov, D.; Mombaur, K.; Diehl, M. Online walking motion generation with automatic footstep placement. *Adv. Robot.* **2010**, *24*, 719–737. [CrossRef]
17. Takenaka, T.; Matsumoto, T.; Yoshiike, T. Real time motion generation and control for biped robot-1 st report: Walking gait pattern generation. In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St Louis, MI, USA, 11–15 October 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 1084–1091.
18. Feng, S.; Whitman, E.; Xinjilefu, X.; Atkeson, C.G. Optimization-based full body control for the darpa robotics challenge. *J. Field Robot.* **2015**, *32*, 293–312. [CrossRef]
19. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
20. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
21. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
22. Yu, W.; Turk, G.; Liu, C.K. Learning symmetric and low-energy locomotion. *Acm Trans. Graph. (Tog)* **2018**, *37*, 1–12. [CrossRef]
23. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
24. Andrychowicz, O.M.; Baker, B.; Chociej, M.; Jozefowicz, R.; McGrew, B.; Pachocki, J.; Petron, A.; Plappert, M.; Powell, G.; Ray, A.; et al. Learning dexterous in-hand manipulation. *Int. J. Robot. Res.* **2020**, *39*, 3–20. [CrossRef]
25. Zhang, Y.; Sun, H.; Sun, H.; Huang, Y.; Hashimoto, K. Biped Robots Control in Gusty Environments with Adaptive Exploration Based DDPG. *Biomimetics* **2024**, *9*, 346. [CrossRef] [PubMed]
26. Zhang, Y.; Xie, J.; Du, X.; Sun, H.; Wang, S.; Hashimoto, K. Biped Robot Terrain Adaptability Based on Improved SAC Algorithm. In Proceedings of the Combined IFToMM Symposium of RoManSy and the USCToMM Symposium on Mechanical Systems and Robotics, Petersburg, FL, USA, 22–25 May 2024; Springer: Cham, Switzerland, 2024; pp. 93–104.
27. Sutton, R.S. Reinforcement learning: An introduction. In *A Bradford Book*; Cambridge University Press: Cambridge, UK, 2018.
28. Morales, E.F.; Murrieta-Cid, R.; Becerra, I.; Esquivel-Basaldua, M.A. A survey on deep learning and deep reinforcement learning in robotics with a tutorial on deep reinforcement learning. *Intell. Serv. Robot.* **2021**, *14*, 773–805. [CrossRef]
29. Yu, W.; Kumar, V.C.; Turk, G.; Liu, C.K. Sim-to-real transfer for biped locomotion. In Proceedings of the 2019 Ieee/rsj International Conference on Intelligent Robots and Systems (Iros), Macau, China, 3–8 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 3503–3510.

30. Bellicoso, C.D.; Jenelten, F.; Gehring, C.; Hutter, M. Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots. *IEEE Robot. Autom. Lett.* **2018**, *3*, 2261–2268. [CrossRef]
31. Lee, J.; Hwangbo, J.; Wellhausen, L.; Koltun, V.; Hutter, M. Learning quadrupedal locomotion over challenging terrain. *Sci. Robot.* **2020**, *5*, eabc5986. [CrossRef]
32. Peng, X.B.; Berseth, G.; Yin, K.; Van De Panne, M. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *Acm Trans. Graph. (Tog)* **2017**, *36*, 1–13. [CrossRef]
33. Cheng, X.; Shi, K.; Agarwal, A.; Pathak, D. Extreme parkour with legged robots. In Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 13–17 May 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 11443–11450.
34. Makoviychuk, V.; Wawrzyniak, L.; Guo, Y.; Lu, M.; Storey, K.; Macklin, M.; Hoeller, D.; Rudin, N.; Allshire, A.; Handa, A.; et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv* **2021**, arXiv:2108.10470.