applied sciences

MDPI

Article

A Disturbance Rejection Control Method Based on Deep Reinforcement Learning for a Biped Robot

Chuzhao Liu [1,2], Junyao Gao [1,2,*], Dingkui Tian [1,2], Xuefeng Zhang [3], Huaxin Liu [1,2] and Libo Meng [1,2]

[1] Intelligent Robotics Institute, School of Mechatronical Engineering, Beijing Institute of Technology, 5 Nandajie, Zhongguancun, Haidian, Beijing 100081, China; 3120150091@bit.edu.cn (C.L.); 3120170099@bit.edu.cn (D.T.); 7420161093@bit.edu.cn (H.L.); menglibo@bit.edu.cn (L.M.)
[2] Beijing Advanced Innovation Center for Intelligent Robots and Systems, Beijing 100081, China
[3] School of Mechanical Engineering, Yanshan University, 438 Hebeidajie, Haigang, Qinhuangdao 066000, China; zhangxuefeng@stumail.ysu.edu.cn
* Correspondence: gaojunyao@bit.edu.cn; Tel.: +86-010-68917611

Abstract: The disturbance rejection performance of a biped robot when walking has long been a focus of roboticists in their attempts to improve robots. There are many traditional stabilizing control methods, such as modifying foot placements and the target zero moment point (ZMP), e.g., in model ZMP control. The disturbance rejection control method in the forward direction of the biped robot is an important technology, whether it comes from the inertia generated by walking or from external forces. The first step in solving the instability of the humanoid robot is to add the ability to dynamically adjust posture when the robot is standing still. The control method based on the model ZMP control is among the main methods of disturbance rejection for biped robots. We use the state-of-the-art deep-reinforcement-learning algorithm combined with model ZMP control in simulating the balance experiment of the cart–table model and the disturbance rejection experiment of the ASIMO humanoid robot standing still. Results show that our proposed method effectively reduces the probability of falling when the biped robot is subjected to an external force in the x-direction.

Keywords: biped robot; model zero-moment-point control; deep reinforcement learning

Citation: Liu, C.; Gao, J.; Tian, D.; Zhang, X.; Liu, H.; Meng, L. A Disturbance Rejection Control Method Based on Deep Reinforcement Learning for a Biped Robot. *Appl. Sci.* **2021**, *11*, 1587. https://doi.org/10.3390/app11041587

Academic Editor: Manuel Armada
Received: 3 January 2021
Accepted: 3 February 2021
Published: 10 February 2021

1. Introduction

Humanoid robots have become a hot spot of research in the field of robotics in recent years, because human appearances are more easily accepted by people. Researchers have developed many service robots using the humanoid robot platform. Most of the robots move using a wheeled chassis instead of using legs like humans. The advantage is a greatly reduced risk of robot falling. However, there are still many researchers working on the stability control of biped robots. In recent years, deep reinforcement learning (DRL) has become a research hotspot in the field of artificial intelligence and robotics. DRL combines the advantages of reinforcement learning (RL) and deep neural network (DNN) technology, i.e., the ability of the agent to search Markov decision process (MDP) problem's policy and the DNN's powerful nonlinear representation.

Recently, an increasing number of DRL algorithms have been developed and applied to robot control. However, it is difficult and risky to deploy state-of-the-art algorithms directly on robots in experiments. The DRL algorithm requires much data, which usually can only be generated from the interaction between the robot and environment. There are many random actions that need to be taken by the robot; some motor rotation angles exceed robot joint position limits, and some actions exceed the maximum torques and maximum speeds of motors. This is likely to cause accidents, whether for the robots or for the researchers. Unlike humans and animals, humanoid robots are prone to falling during the training of stability, which may result in damage to the robot. In addition, the control strategy of humanoid robots is complicated and sometimes requires multiple

stability control methods to operate simultaneously [1]. It is thus necessary to develop a method that can be quickly trained and verified effectively in a simulation environment.

The model zero-moment point (ZMP) control method is among the disturbance rejection control methods for biped robots. In general, biped robots require a mixed disturbance rejection control scheme of multiple methods. The model ZMP control method is a kind of disturbance rejection control method that quickly and slightly adjusts the acceleration of the center of mass of the humanoid robot above the waist in the forward direction. The simplified model of this method can be represented by a table with freely rotating legs and a cart placed on the desktop. When the cart moves to the side of the table with a reasonable acceleration, the table can maintain balance until the cart flies off the table. Too large or small acceleration will cause the desktop to be unable to maintain parallel to the ground for a long time. Regarding the acceleration of the cart at any time as the acceleration of the upper body of the biped robot, we transferred the cart–table model to the real biped robot.

In view of this situation, we deployed the agent using DRL algorithms for a robot and replace the real environment with a simulation environment. This can reduce the cost of development while improving safety. Algorithms can be easily adjusted and monitored at any time. Additionally, the final training effect can be better quantified than that in a real environment. However, a problem that cannot be ignored is the gap between the simulation environment and the real environment. In a simulation, the robot is in an ideal state, without the effects of natural factors, and the irregular topography is relatively predictable. In addition, there are differences between the real robot and robot model in the simulation, even if the model is established accurately.

The main contributions of this paper, against the background of the above research dilemma, are as follows.

(1) We used six DRL algorithms to train an agent to generate policies that maintain the balance of the table in the cart–table model as long as possible. Comparison results for two tasks of a fixed initial angle of the table and a random initial angle of the table show that the deep deterministic policy gradient (DDPG) and model-based model predictive control (MPC) algorithms performed best among the six algorithms.

(2) We proposed a method that combines DRL with a biped robot standing still for disturbance rejection and then used DDPG and model-based MPC algorithms that perform well for the cart–table model for the biped robot model in the x-direction in the simulation. The experimental results show that the biped robot standing still in the simulation effectively resists an impact in the x-direction.

(3) A method of using DRL to train the disturbance rejection ability of a humanoid robot was proposed. We combined a stability control method of a humanoid robot, namely, model ZMP control and DRL in simulation and consider application to a robot in the y-direction and deployment in real robots.

The remainder of the paper is organized as follows. Section 2 reviews related work and highlights the contributions of the paper. Section 3 introduces the concepts of model ZMP control and the cart–table model and details the proposed method. Section 4 presents experiments and results. Section 5 analyzes the experimental results and presents conclusions.

2. Related Work

2.1. Calculation of the ZMP Position

In 1972, Vukobratović and Stepanenko defined the ZMP concept at the beginning of their paper on the control of humanoid robots [2]. The ZMP is the point on the ground at which the tipping moment equals zero. According to Vukobratović's theory, when a person stands upright on the ground, their ZMP coincides with the projection point of the center of gravity. The smallest polygonal area of all contact points between the foot and the ground is called the support polygon. If the center of gravity projection can be kept within the support polygon, then a person maintains balance no matter how they move.

Humanoid robots are usually equipped with six-axis force/torque sensors on the soles of their feet. One such sensor is installed on each foot at the joint between the sole of the foot and the ankle joint. Equations (1) and (2) are the ZMP formulas for the right foot based on measurements made by one six-axis force/torque sensor:

$$p_{R_x} = \frac{(-\tau_y - f_x d)}{f_z}, \tag{1}$$

$$p_{R_y} = \frac{(\tau_x - f_y d)}{f_z}, \tag{2}$$

where

$$p_R = \begin{bmatrix} p_{R_x} & p_{R_y} & p_{R_z} \end{bmatrix}^T$$

Here, p_R is the ZMP position of the right foot, and d is the installation height of the sensor from the ground.

After calculating the ZMP positions, p_R and p_L, of the two feet, the ground forces f_R and f_L are obtained from the force-torque sensor measurements. Using this information, we obtain the ZMP position when the two feet are considered at the same time:

$$p_x = \frac{p_{R_x} f_{R_z} + p_{L_x} f_{L_z}}{f_{R_z} + f_{L_z}}, \tag{3}$$

$$p_y = \frac{p_{R_y} f_{R_z} + p_{L_y} f_{L_z}}{f_{R_z} + f_{L_z}}, \tag{4}$$

2.2. Biped Robot Controlled by DRL

Humanoid robots are high-dimensional non-smooth systems with many physical constraints. Nowadays, an increasing number of humanoid robot control algorithms are being developed adopting DRL in research on, for example, robot balance [3–5], the dynamic performance after the training of a legged robot [6] and the passive dynamic walking robot [7]. Vuga et al. [8] used motion capture with a model-free reinforcement learning algorithm to reduce differences in humanoid robots learning human actions. Wu et al. [9] proposed a biped robot control method based on stability training and reinforcement learning by applying random disturbance to the robot. Tedrake et al. [10] designed a passive biped robot based on control with a policy gradient (PG) algorithm that can learn to walk while adapting to the terrain itself. Cristyan et al. [11] proposed a multilevel system and used the Q-learning method in this system to make the NAO robot walk quickly in simulation. Ao et al. [12] proposed a hybrid reinforcement learning method to keep the NAO robot balanced on an oscillation platform with different frequencies and amplitudes in the simulation. Takamitsu et al. [13] proposed a learning framework for central pattern generation (CPG)-based biped locomotion with a policy gradient method and applied it to real robots. Cai et al. [14] applied actor–critic (AC) algorithms to a humanoid robot learning to crawl and a puppy robot learning to gallop, and compared two types of AC algorithm (i.e., policy search and PG algorithms). Mohammadreza et al. [15] designed a learning framework using the proximal policy optimization (PPO) algorithm; the biped robot developed on this basis can recover from large external forces. Wu et al. [16] used an improved DDPG algorithm to study the walking stability of a passive biped robot. Mohammadreza et al. [17] proposed a model-based method for MPC, which is based on a framework that generates biped robot actions. Simulation results show that the framework can generate actions robustly.

In addition, many researchers are now conducting research on momentum-based robot control. Koolen et al. [18] proposed a control framework based on momentum and quadratic programming for the humanoid robot "Atlas", and demonstrated its ability to walk on rough terrain. Using a reformulation of existing algorithms, Herzog et al. [19]

proposed a simplification method based on the cascades of quadratic programs and hierarchical inverse dynamics. The experimental results prove the effectiveness of the method for feedback control under a real robot. Birjandi et al. [20] introduced a new regressor-based observer method for improving the collision detection sensitivity of the serial manipulators. Compared to the state of the art, this method improves collision detection accuracy and sensitivity.

All the above studies considered the improvement of robot stability control from the perspective of improving algorithms. To solve disturbance rejection from a practical point of view, the present paper develops a biped robot x-direction disturbance rejection training environment. Different algorithms can be used to test the disturbance rejection ability in the simulation, so that actual problems encountered in the research and development of a simulation robot can be solved quickly and conveniently.

3. Method

Since the simplified model of the model ZMP control method is the cart–table model, we took the model as the research object for analysis. The acceleration of the car represents the acceleration of the upper body of the robot. Only when moving at an appropriate acceleration, the table has the possibility to maintain balance for as long as possible, and the counterpart is the biped robot to maintain balance. The present study considers a force of rolling friction between the wheels of a cart and table and assumes that there is no relative sliding. Hirai et al. [21] stated that the table remains upright if the cart is moving with proper acceleration, while the table rises with excessive acceleration and the table sinks with insufficient acceleration, as shown in Figure 1.

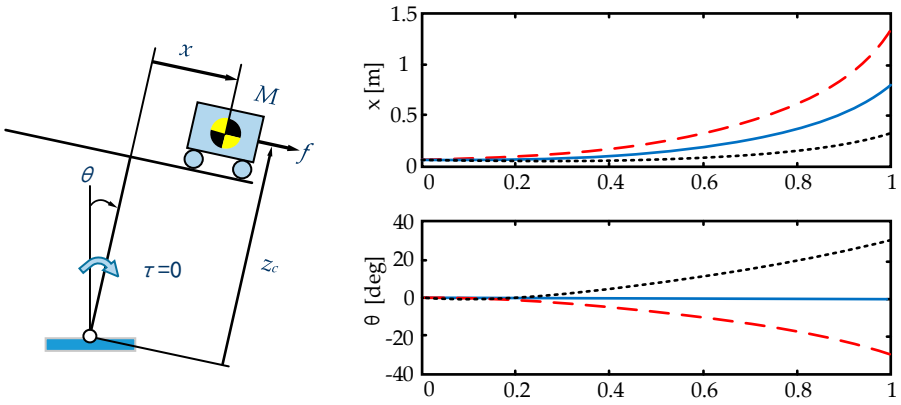


Figure 1. Relationship between the tilt angle θ and the acceleration $\ddot{x}$ of the cart.

The table tilts forward an angle θ relative to the vertical position. The cart acceleration $\ddot{x}$ corresponds to the body acceleration. Kajita et al. [22] considered the physical meaning of model ZMP control using a version of the cart–table model shown in Figure 2 and adopting Lagrange's method:

$$\begin{cases} \left(x^2 + z_c^2\right)\ddot{\theta} + \ddot{x}z_c - g(z_c \sin\theta + x\cos\theta) + 2x\dot{x}\dot{\theta} = \tau/M \\ \ddot{x} + \ddot{z}_c - \dot{\theta}^2 x + g\sin\theta = f/M \end{cases}, \tag{5}$$

where τ is the torque acting on the support point of the table, and f is the force that accelerates the cart on the table. Equation (6) is the result of linearizing Equation (5) around θ, $\dot{\theta} = 0$ and setting $\tau = 0$:

$$\left(x^2 + z_c^2\right)\ddot{\theta} = gx + gz_c\theta - z_c\ddot{x} \tag{6}$$

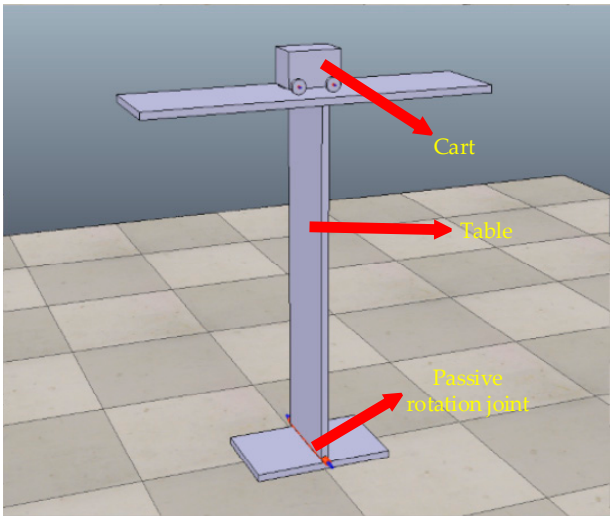


Figure 2. Cart–table model in the simulation.

Now, we assume that the target position of the cart x_d is generated by the dynamics of the linear inverted pendulum

$$\ddot{x}_d = \frac{g}{z_c}(x_d - p_d), \tag{7}$$

where p_d is the target ZMP. We, therefore, draw the following conclusions.

(a) The acceleration of the cart is increased when the target ZMP is set behind the current ZMP ($p_d < 0$).
(b) The acceleration of the cart is decreased when the target ZMP is set in front of the current ZMP ($p_d > 0$).

The main method of this research is to send the desired driving force to the cart's wheels within each control cycle, so as to continuously control the acceleration of the cart at any time (i.e., in any control cycle). Assuming that x in Equation (6) is the position of the cart x_d and substituting Equation (7) into Equation (6), it follows that

$$\ddot{\theta} = \frac{g}{x_d^2 + z_c^2} p_d + \frac{g z_c}{x_d^2 + z_c^2} \theta \tag{8}$$

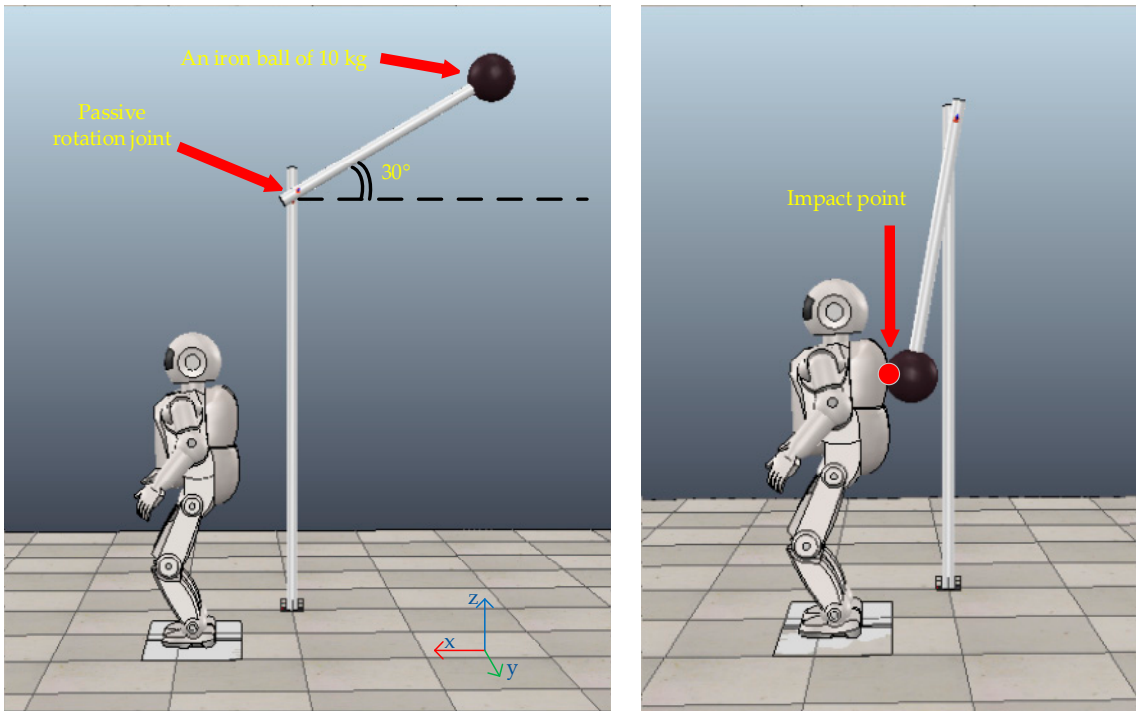
Equation (8) shows that the tilt angle θ of the table can be controlled by changing the position p_d of the target ZMP. In other words, the ZMP of the biped robot model can be controlled by the acceleration of the cart.

3.1. Model Generation

A cart–table model and biped robot walking model were established in V-REP [23]. Figure 2 shows the cart–table model of the model ZMP control method. V-REP is a robot simulator with an integrated development environment and distributed control architecture. We used a remote application programming interface for communication, with a Python integrated development environment as the client and V-REP as the server. A cart with mass M runs on a table whose mass is negligibly small. The foot of the table rotates freely around the fulcrum.

In this work, we trained the agent to control the cart to move quickly to the right (forward) or left (backward), as shown in Figure 2. Owing to the rolling friction between the wheels and desktop, the movement of the car tilts the table left or right. The link between the table leg and the lower board is a passive joint that rotates at will. We adopted six methods of DRL, namely, PG, Monte Carlo (MC)-based AC, temporal difference (TD)-based AC, PPO, DDPG and model-based MPC methods to train cart–table models with different starting conditions and thus demonstrate the feasibility of the method comprehensively.

According to the related work described in Section 2, the ZMP must always be located in the support polygon, and we thus established a biped robot model in the simulation environment as shown in Figure 3. At the beginning of simulation, the pendulum is in its reset state, as shown in Figure 3a. The ball accelerates and swings downward under the effect of gravity (where the pendulum is hard and the weight is negligible). When the iron ball collides with the back of the robot, the robot moves forward owing to the external force. At this time, the reinforcement learning algorithm begins to train the agent to interact with the environment (Figure 3b). The two experimental models are similar, and we thus used algorithms that perform well on the cart–table model to train the robot model and thus obtain a more reasonable and computationally efficient result.



(**a**) (**b**)

Figure 3. Biped robot in the simulation environment. (**a**) At the start of simulation, the pendulum is in its reset state, i.e., it is inclined upward at an angle with the ground of thirty degrees. (**b**) The pendulum with an iron ball accelerates and falls under the influence of gravity. When the back of the robot detects that the ball has struck the robot, the agent starts to interact with and learn the environment.

3.2. Policy Training

Six DRL algorithms were used in training the cart–table model, namely, the PG, MC-based AC, TD-based AC, PPO, DDPG and model-based MPC methods were used to train the agent to control the movement of the cart so that the table leg is upright.

In the PG method, the policy was first parameterized, and the parameters were then updated using gradients. Finally, optimal parameters, expressing the optimal policy, were found. The regression network was used as the policy network because the output of the cart–table model is a continuous variable. Gaussian policies were used for agent training, and the mean and standard deviation of the Gaussian distribution were parameterized. The network model of the policy is shown in Figure 4. In the cart–table model, the state of six DRL algorithms is the cosine of the table angle, sine of the table angle, table rotation speed, cart position and cart speed. The output of the network is the driving force of the cart wheels. This state–action pair was used in the cart–table model with six algorithms described in this section.

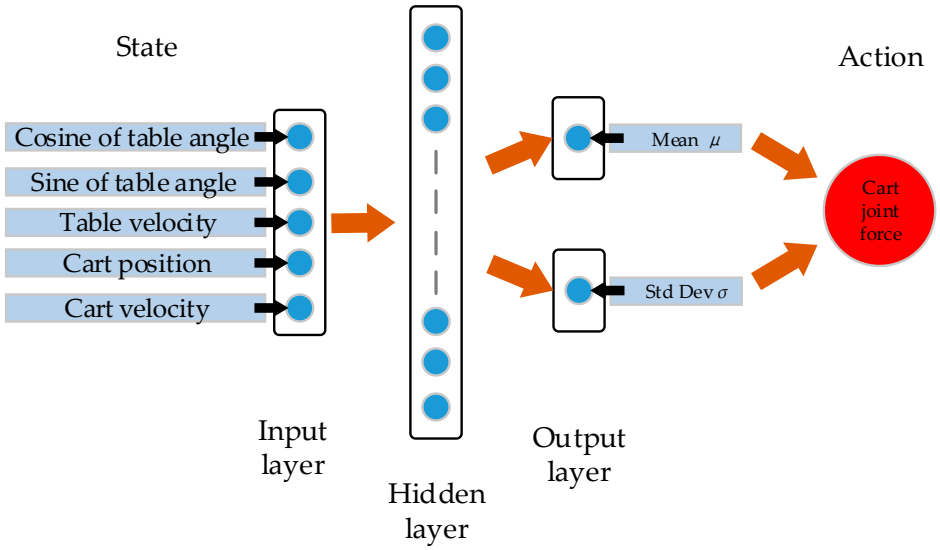


Figure 4. Policy network of the policy gradient (PG) method in cart–table model.

Compared with the classic REINFORCE [24] algorithm of reinforcement learning, we used a batch-based method instead of updating for each state s. When using the current policy to collect N trajectories, each trajectory contains T state–action pairs. These $N \times T$ data points are used to estimate the gradient. This helps improve the stability of the PG method. The final stochastic policy gradient formula is thus

$$\nabla_\theta U(\theta) \approx \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{T} \nabla_\theta \log \pi\left(a_t^i | s_t^i\right) \left[\sum_{k=t}^{T} R\left(s_k^i\right)\right] \tag{9}$$

where $R(s_k)$ is the cumulative discounted reward from this state, owing to the current action having no relationship with the past return.

In Equation (9), $\sum_{k=t}^{T} R\left(s_k^i\right)$ estimated using the MC method in the PG algorithm is adopted to judge the effect of the action; it is usually called the critic. However, online estimation methods are usually inaccurate, and they cannot solve the problem according to long-term interaction with the environment. $\nabla_\theta \log \pi_\theta(a_t|s_t)$ is related to policy and usually called the actor. In the AC algorithm, we used the function approximation method to estimate the critic, and make the critic as close to the true value as possible through continuous learning. The AC algorithm adopted in this research is thus written as

$$\mathrm{g} = \mathrm{E}[\sum_{t=0}^{\infty} \delta_t \nabla_\theta \log \pi_\theta(a_t|s_t)], \tag{10}$$

where $\delta_t = \hat{Q} - V(s_t; w)$ is the critic, and the state value function $V(s)$ is parameterized as $V(s_t; w)$. $\hat{Q}$ is the estimated value of the current action–value function. If a single-step update is adopted, the parameter update rule is

$$\theta \leftarrow \theta + \alpha^\theta \delta_t \nabla_\theta \log \pi_\theta(a_t|s_t) \tag{11}$$

The purpose of this review is to improve the estimation of the value function. The loss function is constructed as

$$\text{loss} = \left(\hat{Q} - V(s_t; w)\right)^2 \tag{12}$$

and the single-step update w is

$$w \leftarrow \mathrm{w} + \alpha^w \delta_t \nabla_w V(s_t; w) \tag{13}$$

In this paper, we used two methods to estimate $\hat{Q}$ in Equation (12) for comparison; one is based on the MC method, while the other is based on the TD method. We used Equation (14) to estimate the critic using the MC method, with the batch update method adopted to estimate $\hat{Q}$:

$$\hat{Q}_t = \sum_{t'=t}^{T'-1} \gamma^{t'-t} r_{t'} + \gamma^{T'-t} V(s_T; w) \tag{14}$$

Using the TD method to estimate $\hat{Q}$, Equation (14) is modified as

$$\hat{Q}_t = r + V(s'; w) \tag{15}$$

We also used the PPO algorithm to improve the training stability and efficiency, because the PPO uses a variable step size of the update instead of a fixed step size while also ensuring monotonicity. The theoretical basis of the PPO algorithm is to use the state distribution of the old policy π to approximate the state distribution of the new policy $\widetilde{\pi}$, as expressed by

$$\eta(\widetilde{\pi}) = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \widetilde{\pi}(a|s) A^\pi(s,a) \tag{16}$$

The advantage function $A^\pi(s,a)$ is calculated as $Q_\pi(s_t,a_t) - V_\pi(s_t)$, where $V_\pi(s_t)$ is calculated using the output of the critic network of the PPO. The objective function to be optimized for the PPO algorithm in this paper is

$$\min L(\theta) = \min \hat{E}_t \left[\min\left(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t \right) \right], \tag{17}$$

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{18}$$

Although the on-policy PPO algorithm based on the stochastic policy has higher efficiency and better convergence, the biggest disadvantage of an on-policy algorithm, including the on-policy PPO algorithm, is low data efficiency. Deterministic off-policy algorithms make full use of data generated by other policies and try to avoid sampling through a large number of random actions to explore the environment. These algorithms are thus more data efficient than on-policy algorithms. We used the DDPG algorithm as a representative deterministic policy algorithm. The DDPG algorithm combines elements of value-function-based and PG-based algorithms. With a DNN, adopting the actor–critic architecture, $\mu(s \mid \theta^\mu)$ is a parameterized actor function maintained by the deterministic PG, and the critic $Q(s,a)$ is learned through Q-learning to use the Bellman equation. The DDPG method is shown in Figure 5.

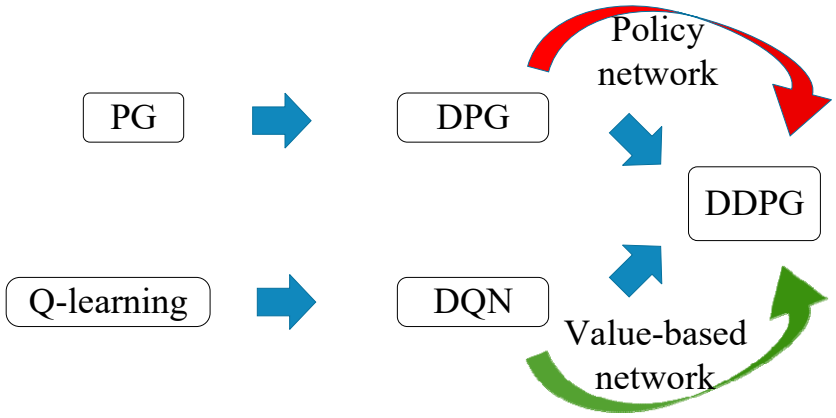


Figure 5. Vanilla deep deterministic policy gradient (DDPG) method.

On the basis of the actor–critic framework with a chain rule, the actor is updated using Equation (19). The DDPG method uses $Q(s,a)$ as a critic to evaluate the policy

$\mu(s \mid \theta^{\mu})$. The cumulated reward is defined as the sum of the discounted future reward $R_t = \sum_{i=t}^{T} \gamma^{(1-t)} r(s_i, a_i)$ with discounting factor $\gamma \in [0,1]$:

$$\nabla_{\theta^{\mu}} J \approx E_{s_t \sim \rho^{\beta}} \left[\nabla_{\theta^{\mu}} Q\left(s, a|\theta^{Q}\right)|_{s=s_t, a=\mu(s_t|\theta^{\mu})} \right] = E_{s_t \sim \rho^{\beta}} \left[\nabla_a Q\left(s, a|\theta^{Q}\right)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s=s_t} \right], \quad (19)$$

where β is a stochastic behavior policy different from μ. A deterministic policy is used in the DDPG method, and the value function thus does not depend on any policy. The DDPG method does not need to perform importance sampling like in the case of a stochastic policy. The exploratory strategy of the off-policy method is adopted by the agent to generate data, and the gradient of policy is calculated using the data. It is, therefore, assumed that the samples come from the policy ρ^{β}. θ^{μ} is a parameter that generates the actor network and θ^{Q} is that for a critic network. The critic network is trained by minimizing the loss:

$$\mathrm{L}(\theta^{Q}) = E_{s_t \sim \rho^{\beta}, a_t \sim \beta, r_t \sim E}[(Q(s_t, a_t|\theta^{Q}) - y_t)^2], \quad (20)$$

where

$$y_t = r(s_t, a_t) + \gamma Q'\left(s_{t+1}, \mu'\left(s_{t+1}|\theta^{\mu'}\right)|\theta^{Q'}\right) \quad (21)$$

The above algorithms are all reinforcement learnings of model-free methods. A model-free method only uses previous data to optimize and improve the current behavior, and we, therefore, used the model-based method to train the model and thus further improve the learning speed and generalization of the agent. We use a method described in the literature [25] to train the agent, i.e., we used a neural network to fit the dynamic model and thus avoid the design of complex MPC controllers. In the model of an agent training of the biped robot, the state of DDPG and model-based MPC algorithms are the x and y positions of robot ZMP, as well as the waist joint angle and angular velocity. The neural network structure of the model-based MPC on the biped robot model is shown in Figure 6. The loss function of the neural network is written as

$$\varepsilon(\theta) = \frac{1}{|D|} \sum_{(s_t, a_t, s_{t+1})} \frac{1}{2} \| (s_{t+1} - s_t) - \hat{f}_{\theta}(s_t, a_t) \|, \quad (22)$$

where D is a data set with state action pairs obtained from sampling, while $s_{t+1} - s_t$ is the output of the neural network used to fit the dynamic model. $\hat{f}_{\theta}(s_t, a_t)$ is the state at the next moment after the agent interacts with the real environment or the simulated environment in the reinforcement learning problem. Minimizing this cost function can bring the output of the dynamic network closer to the transition output of the environment, providing richer knowledge for the reinforcement learning problem and greatly improving the learning speed.

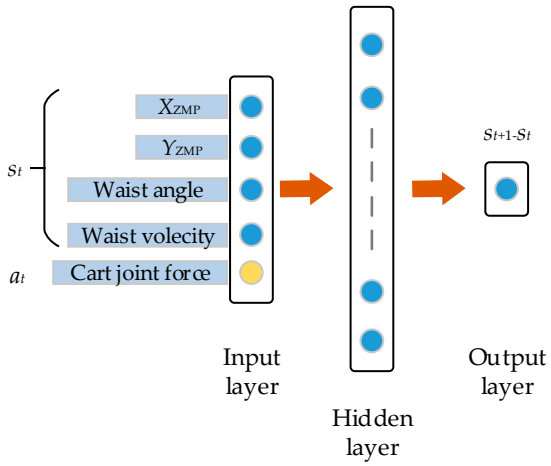
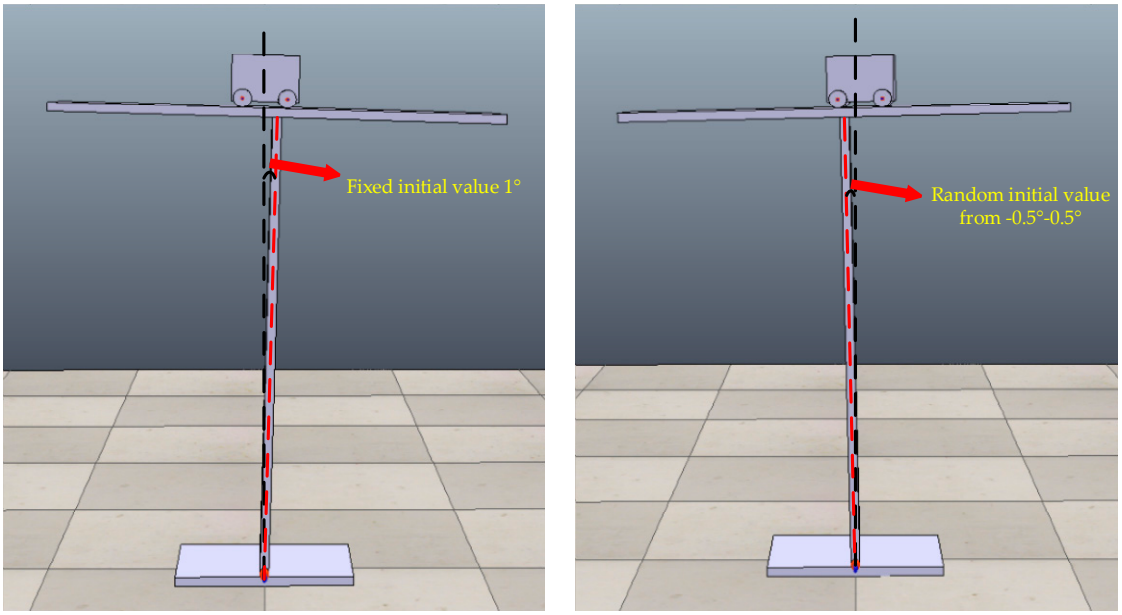


Figure 6. Neural network structure.

4. Experiments and Results

The learning process takes place for a fixed number of time steps. To evaluate the performance of the algorithm, several simulations were performed in the first experiment. The average accumulated reward and average number of evolution steps of several trials better reflect the efficiency and reliability of the algorithm. The experiments were performed using a personal computer with an Intel Core i5 2520 2.5-GHz CPU and 8 GB of RAM running Windows 7. Python 3.7 was used as the programming language, TensorFlow 1.14 was used as the deep learning framework and V-REP 3.5 EDU was used as the simulation environment for these two experiments.

We first designed two training tasks to verify the performance of the proposed scheme for the cart–table model that is used for the stability control of a biped robot when walking or in standby. The two tasks were co-simulated in Python 3.7 and V-REP. The difference between tasks was that the initial angle of the table was a fixed value of 1° in the first task, while it was a random value ranging from −0.5° to 0.5° in the second task, as shown in Figure 7.



(**a**) (**b**)

Figure 7. Two tasks in the evaluation of the proposed scheme. (**a**) Task 1: the initial angle between the leg of the table and the vertical direction is a fixed value of 1°. (The red dashed line is the table leg, and the black dashed line is the vertical direction.) (**b**) Task 2: the initial table angle is set as a random value between −0.5° and 0.5° for each episode.

We then applied the best-performing DDPG algorithm and model-based DRL algorithm based on MPC to the cart–table model to train the anti-disturbance ability of the ASIMO robot model in the x-direction when the biped robot is standing still in the simulation environment.

4.1. Agent Training for the Cart–Table Model

We used the six classic DRL algorithms introduced in Section 3 to train the model in the simulation. The condition for the end of each episode of simulation is that the angle between the desktop and the horizontal direction is greater than 15° or less than −15°. This design indicates that when the biped robot tilts forward or backward more than 15°, it is basically impossible to adjust the tilt of the upper body of the biped robot to rebalance the robot. Figure 8 shows the state when the table is about to lose its balance, when the absolute value of the table tilt angle is greater than or equal to 15°.

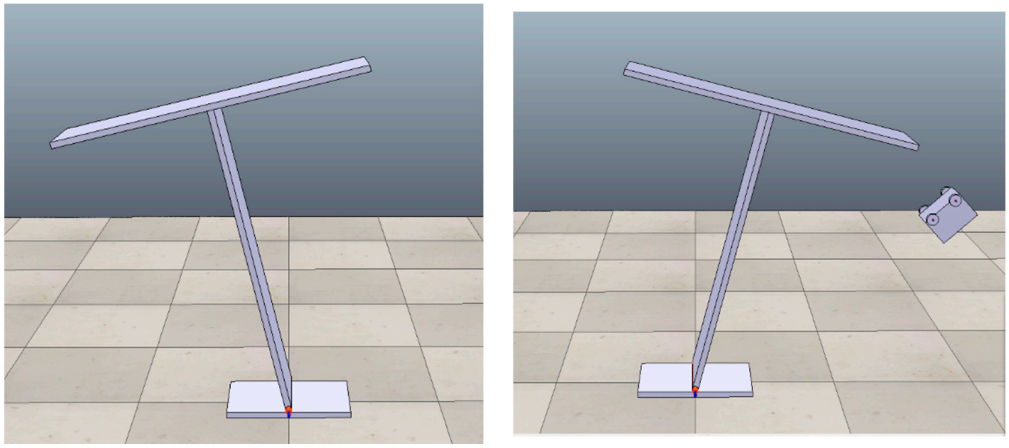


Figure 8. End of simulation of one episode when the table tilt angle is too large.

In this experiment, the reward function of reinforcement learning is

$$r_t = \frac{1}{(\theta_t^2 + 1)}, \tag{23}$$

where θ_t is the tilt angle of the table at each step. The simulation time step was 0.01 s.

The pseudo-code for training the cart–table model with reinforcement learning algorithms is presented in Algorithm 1. We executed each of the algorithms to train tasks 1 and 2. The use of the same program framework ensures that the results of different algorithms can be compared.

Algorithm 1. Cart–table model with DRL algorithms.

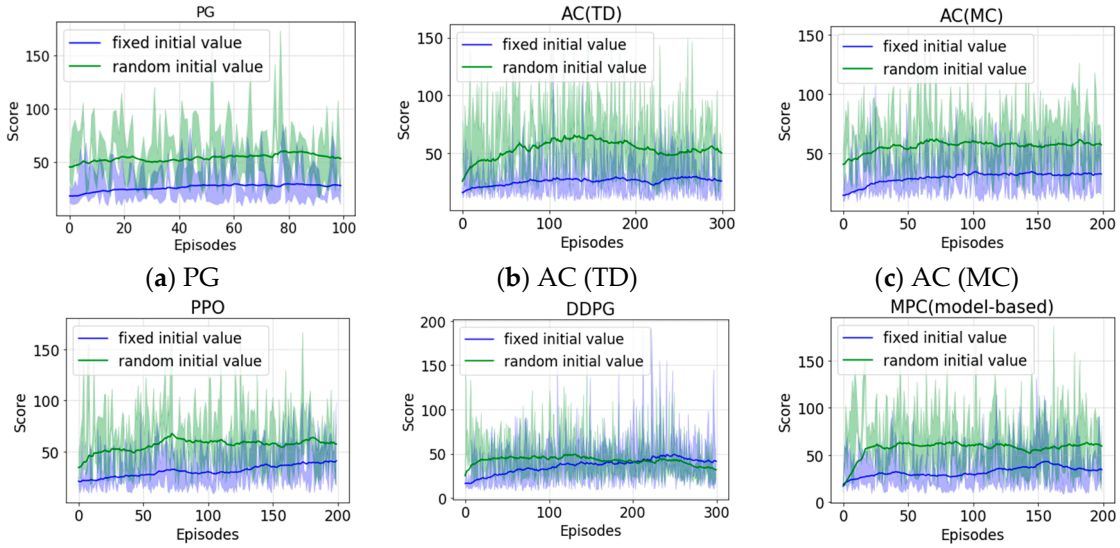
Build a cart–table model in V-REP
Choose a DRL algorithm
Randomly initialize a set of weights w and biases b of the deep neural network of this algorithm
for ep = 1, *EPISODE* **do**
Initialize the simulation environment: set the Cart to zero position to $x_0 = 0$ (the midpoint of the desktop), set the speed to $v_0 = 0$ and acceleration to $a_0 = 0$
Set the table tile angle θ_0 to a random value within the range from -0.5° to 0.5° (task 1) or a fixed value of 1° (task 2).
Get initial state s_0 by sorting (θ_0, x_0, v_0)
for t = 1, *STEP* **do**
Select action a_t according to current policy
Execute a_t to obtain reward r_t and observe new state $s_t(\theta_t, x_t, v_t)$
Add r_t to reward r
Obtain loss according to network output
Update network parameters using information such as the loss and gradient:

$$w_t \leftarrow w_t + \Delta_t^w$$
$$b_t \leftarrow b_t + \Delta_t^b$$

end for
end for

Figure 9 shows the training curves of the six algorithms for the two tasks. The blue line is the curve for task 1 and the green line the curve for task 2. Each task for each algorithm was trained three times. The shaded area of each graph is the range between the maximum and minimum scores in the three experimental runs. The thick line represents the mean of the average scores of the three experimental runs, which can also be called the average rising curve of scores obtained from the three experimental runs. Figure 10 is the result of extracting all the thick lines in Figure 9 and putting them into their respective tasks for comparison. Figure 9 shows the performance comparison of each algorithm on

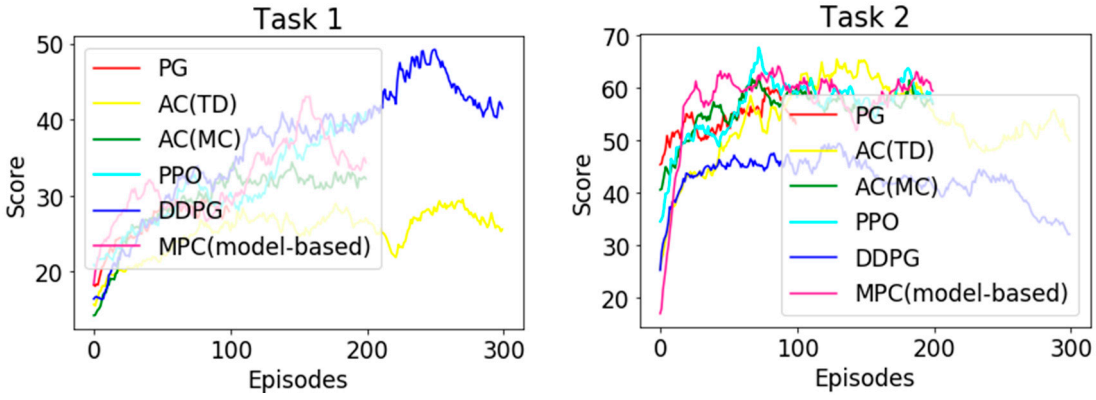
two tasks. Figure 10 shows the performance comparison of the six algorithms deployed in each task. The figures clearly show that deploying the DDPG- and model-based MPC algorithms for task 1 achieved better training results. In task 2, the policy trained adopting the model-based MPC method quickly maintained the table balance, and the balance was maintained for a relatively long time. The PPO method had similar performance. The reason for this may be that the initial angle of the desktop in task 2 was smaller than that in task 1, and it was thus easier to achieve long-term stability.



(**a**) PG (**b**) AC (TD) (**c**) AC (MC)

(**d**) PPO (**e**) DDPG (**f**) MPC (model-based)

Figure 9. Curves of the six reinforcement learning algorithms for two tasks.



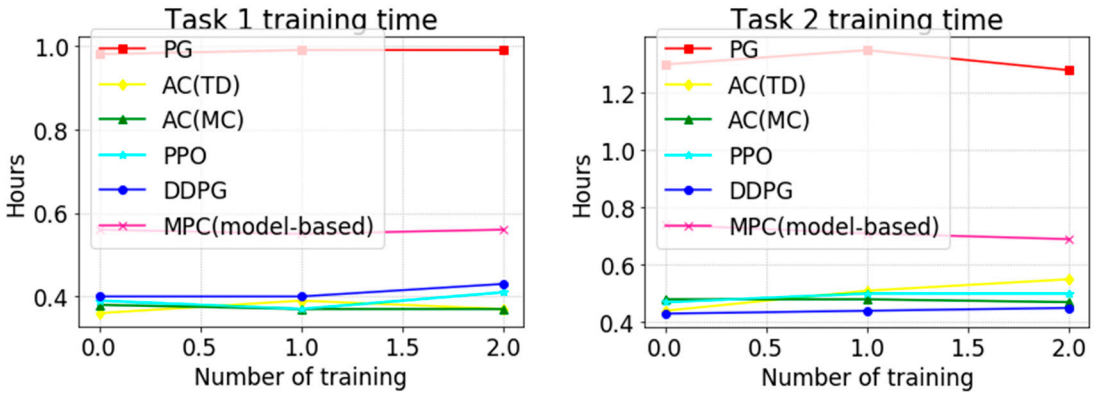
(**a**) Task 1 (**b**) Task 2

Figure 10. Performance comparison curves of the six reinforcement learning algorithms.

In task one, DDPG performed well, but this was not the case at all in task two. We think this is due to the inherent advantages and disadvantages of the deterministic policy gradient method. Because the deterministic policy is not conducive to the exploration of the action, it is easy to fall into the local minimum. To obtain an accurate policy, the estimated value of q also needs to be more accurate. However, when learning starts, the estimated value of q is not accurate enough to make the parameter transmission more accurate. Therefore, the setting of noise becomes particularly important. The high parameter sensitivity of DDPG is also among the reasons why the stochastic policy gradient method performs better in some tasks.

Figure 11 shows the time required to train the agent in the three runs using different algorithms for two tasks. Owing to equation 11, if the PG method is to be accurate, many

samples are required, which undoubtedly increases the training time. The training runs of other algorithms were at an acceptable level. The one-episode training of the model-based MPC method took a little longer. This is because the graphical processing unit of the personal computer not only displays the interaction scene between the model and environment but also performs neural network calculations, which slightly decreases the performance of the personal computer.



(**a**) Task 1 (**b**) Task 2

Figure 11. Training times of algorithms.

4.2. Agent Training for the Balance Control of a Biped Robot

In this work, we trained the agent to control the upper-body pitching motion of the biped robot for adjusting the ZMP. We used the two algorithms achieving the best performance for tasks 1 and 2 in Section 4.1, namely, the DDPG and model-based MPC algorithms, to train the agent in this experiment. Each episode finished when the robot fell or remained standing for 3 s. The simulation time step was 0.01 s. In this experiment, the reward function of reinforcement learning was

$$r_t = -p_x^2 - p_y^2 + 10, \tag{24}$$

where p_x and p_y are the position coordinates of the ZMP. The two feet of the robot in this experiment did not leave the ground, and p_x and p_y were thus located at the center of the support polygon of the robot's sole when the biped was standing still.

According to the previous analysis, irrespective of whether the biped robot is subjected to an external force in the positive x-direction or the upper body is bent forward with a small acceleration, the ZMP moves to the front half of the foot until the edge of the toe, as shown in Figure 12. Conversely, when the robot slowly leans back from the initial position, the ZMP moves to the back half of the foot until the edge of the back heel. In this experiment, we only considered the pitching action of the robot in the x-direction under the influence of the external force, without considering the force in the y-direction, and at the same time, the feet did not actively leave the ground when the robot was standing. It was, therefore, reasonable to use the movement of the ZMP in the x-direction under one-foot support in a simplified analysis.

We used the two DRL methods with the best performance in Section 4.1 to train the agent. The training results are shown in Figure 13. Both algorithms were trained five times to reduce errors generated by randomness. The thick lines in the two figures represent an average score, which is the sum of 0.95 times the cumulative score and 0.05 times the current score. Such a curve is guaranteed to appear sufficiently smooth.

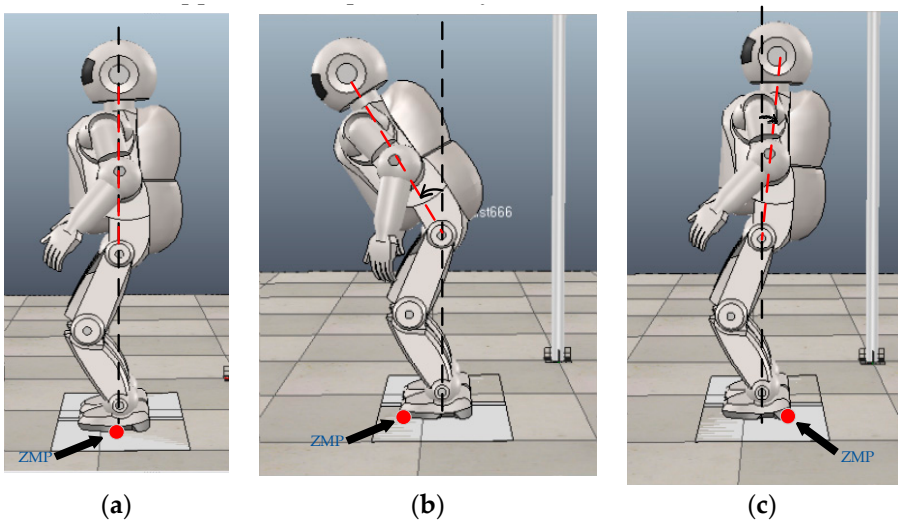


Figure 12. Zero moment point (ZMP) movement in the x-direction: (**a**) robot standing upright, (**b**) center of gravity of the robot moving forward and (**c**) center of gravity of the robot moving back.

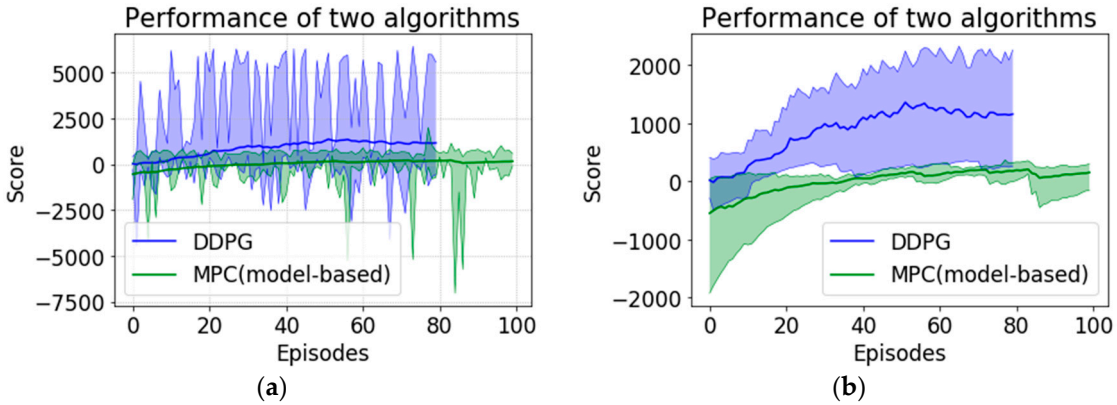


Figure 13. Learning curves for the performance of the policies trained with different methods: (**a**) maximum-to-minimum interval of each episode in five experimental runs and the average performance and (**b**) maximum-to-minimum interval of the average score of each episode in five experimental runs and the average performance.

Figure 13a shows that when combining the results of five experimental runs, the biped robot trained using the DDPG method has a high probability of returning to an upright position after being trained for about 30 episodes. However, it is difficult for the agent trained by the model-based MPC method to have an effective robot upper body control strategy.

Figure 13b shows the performance difference between the two methods more clearly. The shaded area is the interval between the maximum and minimum average scores of the two methods in five experimental runs. The lines are the average of the average scores for five experimental runs. Notably, in the five implementations of the model-based MPC method, the first-episode score at a certain start of training was large and negative (i.e., about −2000). However, this did not affect the trend of the curve. When the curve became flat after 60 episodes, the performance of the algorithm was still far inferior to that of the DDPG method, as confirmed by the training video in the supplementary material.

The two algorithms were trained five times. Figure 14 shows the training time curves. Combining Figures 13 and 14 reveals that the DDPG algorithm has a higher efficiency than the model-based MPC algorithm in this experiment. It is also seen from the video in the supplementary material that the former has better performance. The average training time of the DDPG algorithm is 0.17 h, while that of the model-based MPC algorithm is 0.2 h.

The hyperparameters of the two algorithms in the biped robot balancing training task are given in Table 1.

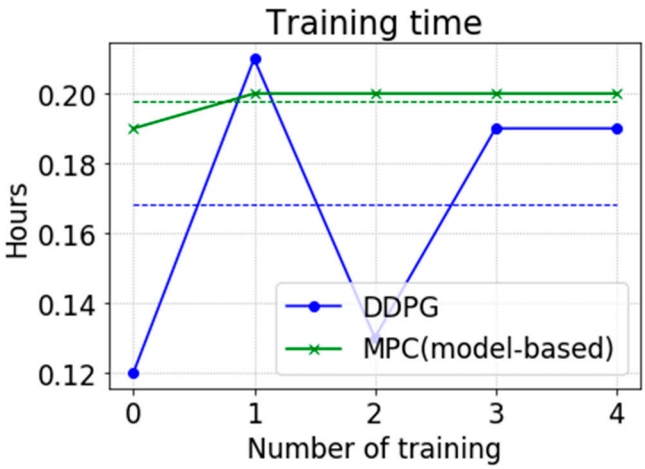


Figure 14. Comparison of training times.

Table 1. Hyperparameters of the two algorithms.

(a) DDPG algorithm	
DDPG Setup Hyper-Parameters	
Actor/Critic learning rate	1×10^{-3}
Reward discount factor γ	0.9
Soft replacement τ	0.01
Batch size	8
Running episodes	80
Experience buffer	5000
Number of neuron on actor network	400
Number of neuron on critic network	200
(b) Model-based MPC algorithm	
Model-based MPC Setup Hyper-Parameters	
learning rate	1×10^{-2}
Batch size	16
Running episodes	100
Buffer size	10,000
Number of candidate action sequences	200
Number of actions per candidate action sequence	20
Number of neuron on input layer	800
Number of neuron on hidden layer	400

5. Conclusions

We studied the possibility of combining two cutting-edge technologies of advanced artificial intelligence: the humanoid robot and DRL. Adopting co-simulation, we first used several DRL algorithms to train the agent's policies, so that the table in the classic cart–table model adopted in the field of humanoid robots can maintain balance for as long as possible. The cart–table model is the theoretical basis of the model ZMP control method for whole-body stability control of the humanoid robot. We then adopted DDPG and model-based MPC algorithms, which performed well in two tasks of the table fixed at an initial angle and had a random initial angle in training in a second experiment. The experimental results were generally satisfactory, as shown by learning curves and simulation results. Our research shows that the DDPG algorithm has important advantages in robot control problems with a high-dimensional state and action space.

In future work, we will change the weight and height of the iron ball to simulate different external forces in the x-direction of the biped robot. We will record the ZMP and pitch angle and speed information of the upper body of the robot and map them to the actions output by the trained strategy in the simulation, so that the state space and robot action space can be quickly mapped and real-time performance and effectiveness ensured.

Since we considered the rigid connection of the arm and the body as a whole, we do not know whether the model ZMP control method is enhanced or weakened by the arm movement. Our next task also includes increasing the arm movement to make the humanoid robot closer to the actual situation when the robot needs to maintain a balance. The situation when there is no external force impact on the axis of the robot also needs to be studied, because this may cause the robot to rotate around the z axis.

Owing to the limitations of cart–table model theory and the mechanical structure of the humanoid robot, the robot cannot take effective upper body control actions in the y-direction. In the next stage of research, in addition to adding a method of model ZMP control for walking, it will be necessary to study the anti-disturbance control method of the robot under an external force in the y-direction, so that the biped robot can use DRL algorithms to resist disturbances in all directions when standing upright and walking.

Author Contributions: C.L. is respondsible for data curation, methodology, writing-original draft, visualization and software. J.G. is respondsible for funding acquisition. D.T. is respondsible for software. X.Z. is respondsible for writing-review & editing. H.L. is respondsible for formal analysis. L.M. is respondsible for validation. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Project under Grant 2019YFB1309501, National Natural Science Foundation of China under Grant 91748202 and 61903038.

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

References

1. Kajita, S.; Morisawa, M.; Harada, K.; Kaneko, K.; Kanehiro, F.; Fujiwara, K.; Hirukawa, H. Biped walking pattern generator allowing auxiliary zmp control. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–15 October 2006; pp. 2993–2999.
2. Vukobratović, M.; Borovac, B.; Surla, D.; Stokić, D. *Biped Locomotion—Dynamics, Stability, Control and Application*; Springer: Berlin/Heidelberg, Germany, 1990.
3. Hyon, S.-H.; Osu, R.; Otaka, Y. Integration of multi-level postural balancing on humanoid robots. In Proceedings of the IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 1549–1556.
4. Stephens, B.J.; Atkeson, C.G. Dynamic balance force control for compliant humanoid robots. In Proceedings of the International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 1248–1255.
5. Li, Z.; VanderBorght, B.; Tsagarakis, N.G.; Colasanto, L.; Caldwell, D.G. Stabilization for the compliant humanoid robot COMAN exploiting intrinsic and controlled compliance. In Proceedings of the International Conference on Robotics and Automation, Saint Paul, MI, USA, 14–18 May 2012; pp. 2000–2006.
6. Hwangbo, J.; Lee, J.; Dosovitskiy, A.; Bellicoso, D.; Tsounis, V.; Koltun, V.; Hutter, M. Learning agile and dynamic motor skills for legged robots. *Sci. Robot.* **2019**, *4*, eaau5872. [CrossRef] [PubMed]
7. Wu, Y.; Yao, D.; Xiao, X.; Guo, Z. Intelligent controller for passivity-based biped robot using deep Q network. *J. Intell. Fuzzy Syst.* **2019**, *36*, 731–745. [CrossRef]
8. Vuga, R.; Ogrinc, M.; Gams, A.; Petric, T.; Sugimoto, N.; Ude, A.; Morimoto, J. Motion capture and reinforcement learning of dynamically stable humanoid movement primitives. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 5284–5290.
9. Wu, W.; Gao, L. Posture self-stabilizer of a biped robot based on training platform and reinforcement learning. *Robot. Auton. Syst.* **2017**, *98*, 42–55. [CrossRef]
10. Tedrake, R.; Zhang, T.W.; Seung, H.S. Stochastic policy gradient reinforcement learning on a simple 3D biped. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No. 04CH37566), Sendai, Japan, 28 September–2 October 2004; Volume 3, pp. 2849–2854.
11. Gil, C.R.; Calvo, H.; Sossa, H. Learning an efficient gait cycle of a biped robot based on reinforcement learning and artificial neural networks. *Appl. Sci.* **2019**, *9*, 502. [CrossRef]

12. Xi, A.; Chen, C. Stability Control of a Biped Robot on a Dynamic Platform Based on Hybrid Reinforcement Learning. *Sensors* **2020**, *20*, 4468. [CrossRef]
13. Matsubara, T.; Morimoto, J.; Nakanishi, J.; Sato, M.A.; Doya, K. Learning CPG-based biped locomotion with a policy gradient method. *Robot. Auton. Syst.* **2006**, *54*, 911–920. [CrossRef]
14. Li, C.; Lowe, R.; Ziemke, T. A novel approach to locomotion learning: Actor-Critic architecture using central pattern generators and dynamic motor primitives. *Front. Neurorobot.* **2014**, *8*, 23. [CrossRef] [PubMed]
15. Kasaei, M.; Abreu, M.; Lau, N.; Pereira, A.; Reis, L.P. A Hybrid Biped Stabilizer System Based on Analytical Control and Learning of Symmetrical Residual Physics. *arXiv* **2020**, arXiv:2011.13798.
16. Wu, X.-G.; Liu, S.-W.; Yang, L.; Deng, W.-Q.; Jia, Z.-H. A Gait Control Method for Biped Robot on Slope Based on Deep Reinforcement Learning. *Acta Autom. Sin.* **2020**, *46*, 1–12.
17. Kasaei, M.; Ahmadi, A.; Lau, N.; Pereira, A. A Robust Model-Based Biped Locomotion Framework Based on Three-Mass Model: From Planning to Control. In Proceedings of the 2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Ponta Delgada, Portugal, 15–17 April 2020; pp. 257–262.
18. Koolen, T.; Bertrand, S.; Thomas, G.; De Boer, T.; Wu, T.; Smith, J.; Englsberger, J.; Pratt, J. Design of a momentum-based control framework and application to the humanoid robot atlas. *Int. J. Hum. Robot.* **2016**, *13*, 1650007. [CrossRef]
19. Herzog, A.; Rotella, N.; Mason, S.; Grimminger, F.; Schaal, S.; Righetti, L. Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Auton. Robot.* **2016**, *40*, 473–491. [CrossRef]
20. Birjandi, S.A.B.; Haddadin, S. Model-Adaptive High-Speed Collision Detection for Serial-Chain Robot Manipulators. *IEEE Robot. Autom. Lett.* **2020**, *5*, 6544–6551. [CrossRef]
21. Hirai, K.; Hirose, M.; Haikawa, Y.; Takenaka, T. The development of Honda humanoid robot. In Proceedings of the 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146), Leuven, Belgium, 20 May 1998; Volume 2, pp. 1321–1326.
22. Kajita, S.; Hirukawa, H.; Harada, K.; Yokoi, K. *Introduction to Humanoid Robotics*; Springer: Berlin/Heidelberg, Germany, 2014.
23. Web Site of CoppeliaSim. Available online: https://www.coppeliarobotics.com/ (accessed on 3 February 2021).
24. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
25. Williams, G.; Wagener, N.; Goldfain, B.; Drews, P.; Rehg, J.M.; Boots, B.; Theodorou, E.A. Information theoretic MPC for model-based reinforcement learning. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 1714–1721.