Walking Stability Control Method for Biped Robot on Uneven Ground Based on Deep Q-Network

Baoling Han[1], Yuting Zhao[1,✉] and Qingsheng Luo[2]
(1. School of Mechanical Engineering, Beijing Institute of Technology, Beijing 100081, China;
2. School of Mechatronic Engineering, Beijing Institute of Technology, Beijing 100081, China)

Abstract: A gait control method for a biped robot based on the deep Q-network (DQN) algorithm is proposed to enhance the stability of walking on uneven ground. This control strategy is an intelligent learning method of posture adjustment. A robot is taken as an agent and trained to walk steadily on an uneven surface with obstacles, using a simple reward function based on forward progress. The reward-punishment (RP) mechanism of the DQN algorithm is established after obtaining the offline gait which was generated in advance foot trajectory planning. Instead of implementing a complex dynamic model, the proposed method enables the biped robot to learn to adjust its posture on the uneven ground and ensures walking stability. The performance and effectiveness of the proposed algorithm was validated in the V-REP simulation environment. The results demonstrate that the biped robot's lateral tile angle is less than 3° after implementing the proposed method and the walking stability is obviously improved.

Key words: deep Q-network (DQN); biped robot; uneven ground; walking stability; gait control
CLC number: TP 242 **Document code**: A **Article ID**: 1004-0579(2019)03-0598-08

Biped robots are structurally similar to humans. Their bipedal walking movement has a wide range of applications, and the control problem of walking stability has always been the focus of research in the field of humanoid robots[1]. In recent years, many scholars at home and abroad have been focused on the problem of walking stability of bipedal robots on uneven ground. Many scholars at home and abroad have tried to use a combination of offline gait planning and various online gait correction algorithms to achieve stable and continuous walking for the actual walking environment[2-5]. For example, in Ref. [2], the traditional inverted pendulum model and zero moment point (ZMP) are used to obtain the planning gait and the real-time information of the angle sensor on the joint of the humanoid robot. The posture is adjusted according to this information. Subsequently, the data is compared to the preplanned gait data, and the resulting error is sent back to the control system for online gait adjustment control.

The aforementioned use of sensor feedback control for online correction has stringent requirements in order to achieve fast and effective real-time solutions to the algorithm, especially in an actual environment. Due to the influence of complex environmental parameters, the actual control effect of the robot is not ideal. The emergence and development of machine learning algorithms provides a new direction for solving the complex control problems of biped robots, and have been increasingly used in the robotics field. Refs. [6 – 8] apply a reinforcement learning algorithm to control robot movement, and have achieved very good control. For example, in Ref. [6] and Ref. [7], reinforcement learning algorithms are used

Received 2018-06-12
Supported by the National Ministries and Research Funds (3020020221111)
✉ Author for correspondence
E-mail: zytkite@163.com
DOI: 10.15918/j.jbit1004-0579.18059

in mobile robots to receive high-dimensional visual information as input data in order to learn actions such as avoiding walls and moving along the centreline. Ref. [8] adopts a deep reinforcement learning method to extend robot learning to complex 3D operation tasks such as opening doors.

In this study, the DQN algorithm[9] is applied to a biped robot's gait control system to address walking stability problems on uneven ground. The algorithm does not need to build a complex dynamic model of the robot. The gait of biped robot is adjusted based on offline gait planning using the DQN algorithm, combining the advantages of reinforcement learning and neural networks. In this study, the distance between the robot body and the planning position standing on one leg is considered the status, while the positional adjustment of the fuselage is considered the action. The fuselage posture information is subsequently used to build a reward function, which can effectively improve the stability of the robot when walking on uneven ground. This method not only avoids the complex process of a dynamic model when using the traditional control method, but also achieves a better control effect.

1 Generation of Offline Gait

The walking model of biped robot can be divided into a static walking model and dynamic walking model[10]. This paper studies the walking stability control of a humanoid robot under low speed and quasi-static conditions. The kinematic model of biped robot is established. The upper body of the robot is regarded as a point mass. The mass is concentrated on the robot's centroid position O_0. Humanoid robot has six degrees of freedom, three degrees of freedom for hip joints, one degree of freedom for knee joints, and two degrees of freedom for ankle joints. The centroid of the humanoid robot is established as the basic coordinate system and the D-H coordinate system as the research object of the right leg. The coordinate system is established as shown in Fig. 1. $OXYZ$ is the global coordinate system, $O_0X_0Y_0Z_0$ is the body coordinate system, $X_R Y_R Z_R$ is the right leg coordinate system, and each joint coordinate system is established in turn.

Based on the D-H coordinate system, the kinematics model is established, and the relationship between the foot end trajectory and the joint rotation angle is associated by inverse kinematics[11]. Tab. 1 shows the walking parameters. From the obtained offline gait sequence, we use the compound cycloid and the cubic polynomial curve for trajectory planning of the centroid and distal end of the robot foot[12-13].

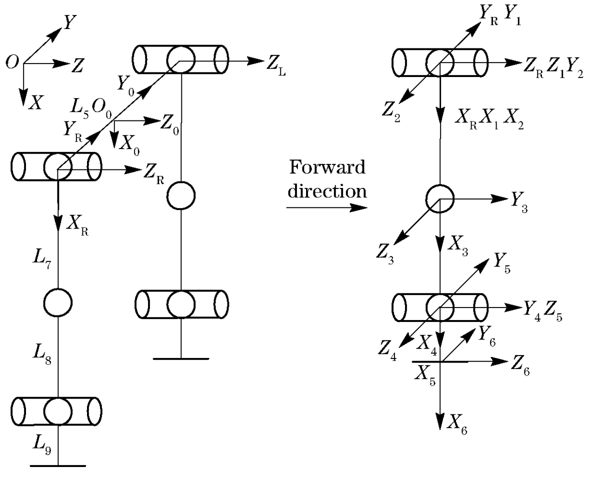


Fig. 1 World, body and the joint coordinate system

2 Gait Control Algorithm for Biped Robot on Uneven Ground

2.1 DQN algorithm

The DQN algorithm is based on the improvement of the Q-learning algorithm in the classical algorithm of reinforcement learning. The Q-learning algorithm is a kind of reinforcement learning algorithm that does not depend on the model[14].

Tab. 1 Gait planning related parameters

Parameter	Walking cycle T/s	Step length S_0/mm	Step height H_0/mm	Single leg support period T_s/s	Leg support period T_w/s
Value	2	30	8	0.6	0.4

The algorithm iterates using $Q(s,a)$ which is an estimation function of the sum of the state-action pair rewards. The optimal policy is selecting the action corresponding to the largest action in the state S. The basic form of the algorithm is

$$Q(S_t,A_t)=Q(S_t,A_t)+\alpha(R_{t+1}+\gamma\max_a Q(S_{t+1},a)-Q(S_t,A_t)) \tag{1}$$

The Q function $Q(S_t, A_t)$ corresponding to a policy is defined as the expected return from S_t after taking action A_t. α is the learning rate (or learning step), which reflects the extent of learning errors. γ is the decay value of future rewards, indicating the impact of future rewards on the present. An action is chosen via the greedy strategy until the value function converges to obtain the optimal strategy:

$$\pi(s)=\arg\max_{a\in A}Q(s,a) \tag{2}$$

This will find the best action for each state. However, for the robot control problem, the state space is often high-dimensional. If the state-action values are calculated and stored individually, a dimensionality disaster easily occurs. In order to solve this problem, the DQN algorithm obtains the Q value by the approximation of the value function as

$$Q(s,a)=f(s,a) \tag{3}$$

The function relation in Eq. (3) refers to learning the relationship between the Q value and state and action through neural network learning. The neural network uses two fully connected neural networks of the same structure and different parameters: the "current value" network and the "target value" network. During the training process, the current Q value and the target Q value are updated. The square of the difference between the current Q value and target Q value is used as the loss function in reverse transmission. The algorithm flow is shown in Fig. 2.

For a high-dimensional state space, DQN algorithm takes state S as the input and outputs a vector: $[Q(s,a_1), Q(s,a_2), \cdots, Q(s,a_n)]$, which is Q value corresponding to all possible actions in the current state, then chooses the best

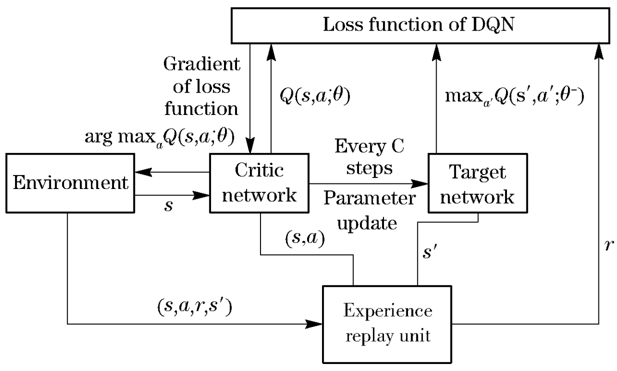


Fig. 2 Algorithm flow of DQN

action. The algorithm process is described in the following pseudocode.

```
Input: Environment $E$; Action $A$; Starting state $x_0$; State $S$; Decay value of future rewards $\gamma$; Learning rate $\alpha$.
Process:
1: Initialize replay memory $D$ to capacity $N$;
2: Initialize action-value function $Q$ with random weights $\theta$;
3: Initialize target action-value function $\hat{Q}$ with weights $\theta^-=\theta$;
4: For episode $=1,M$ do:
5:     Initialize sequence $s_1=\{x_1\}$ and preprocessed sequence $\phi_1=\phi(s_1)$
6:     For $t=1,T$ do:
7:         With probability $\varepsilon$ select a random action $a_t$
8:         otherwise select $a_t=\arg\max_a Q(\phi(s_t),a;\theta)$
9:         Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
10:        Set $s_{t+1}=s_t,a_t,x_{t+1}$ and preprocess $\phi_{t+1}=\phi(s_{t+1})$
11:        Store transition $(\phi_t,a_t,r_t,\phi_t)$ in $D$
12:        Sample random mini-batch of transitions $(\phi_j,a_j,r_j,\phi_j)$ from $D$
13:        Set $y_j=\begin{cases} r_j, & \text{if episode terminates at step } j+1 \\ r_j+\gamma\max_{a'}\hat{Q}(\phi_{j+1},a';\theta^-), & \text{otherwise}\end{cases}$
14:        Perform a gradient descent step on
```

```
$(y_j - Q(\phi_j, a_j; \theta))^2$ from $D$
15:             Network parameters $\theta$
16:             Every C steps reset $\widehat{Q} = Q$
17:       End For
18: End For
Output: The $Q$ value corresponding to all possible actions in the current state
```

2.2 Algorithm implementation

2.2.1 Robot state space

Select the robot's posture information that can intuitively reflect whether the robot is stable or not. That is, the angle θ_z at which the x-y plane rotates about the Z_0-axis at each sampling time and the angle θ_y at which the x-z plane rotates about the Y_0-axis. So the state space of DQN algorithm is $S = \{\theta_z, \theta_y\}$ and the state value is $s_t (s_t \in S)$.

2.2.2 Robot action space

The humanoid robot adopts a "one-stop and one-stop" strategy, that is, when single-phase support is activated, the swinging leg is manipulated such that the centroid of the leg support phase is adjusted to ensure that the robot's centroid is always in the stable support zone. When the robot is in the single-legged support phase, the projection of the centroid on the ground is located in the centroid of the support foot[15]. However, when the gait is applied to a complex terrain, the centroid position of the robot changes, so it is necessary to adjust the position of the fuselage. Therefore, the action is discretized as the adjustment value of the centroid in the front-back and left-right directions. The humanoid robot platform used in this article is 100 mm long and 60 mm wide. The step size s_0 is set to 30 mm. The centroid adjustment distance is set to 0 mm, 5 mm, 10 mm and 15 mm as the centroid adjustment direction for the z-axis and y-axis direction. Each axis is divided into positive and negative directions, that is $\{\pm Z_0, \pm Y_0\}$. Therefore, during the adjustment of each step, the size of the action value space A is 64, and the gait adjustment action recorded is represented as $a (a \in A)$.

2.2.3 Mechanism of RP

In the design of the DQN algorithm's reward and punishment function, which determines whether or not to fall after being adjusted by the humanoid robot, is the final evaluation. When the robot falls, the reward value is negative ten, and the steady reward value is fifty when the robot does not fall.

$$R = \begin{cases} -10, & \theta_z \geqslant 60^\circ \text{ or } \theta_y \geqslant 60^\circ \\ 50, & \text{otherwise} \end{cases} \tag{4}$$

The robot is falling when $\theta_z \geqslant 60^\circ$ or $\theta_y \geqslant 60^\circ$, while other cases count as a positive reward.

2.2.4 Walk controller learning process

In the above sections, the state value, action value, reward and punishment function and network structure of DQN algorithm are designed. The network structure parameters in the DQN algorithm and the robot's centroid coordinates are updated on each step the robot moves. Based on the robot's gait planning and pitch sensor feedback information, the agent continuously interacts with the environment via the DQN algorithm to obtain information detailing the relationship between the centroid adjustment and robot pose. The learning process is shown in Fig. 3.

3 Simulation Verification and Result Analysis

3.1 Simulation environment

In this study, V-REP is used as a simulation environment and uses an Asti model biped robot, 10 mm square obstacles with a thickness of 4 mm were randomly distributed in the robot's direction of movement to simulate "uneven ground" as shown in Fig. 4. The gait is based on the offline gait programming and is imported into the "Path" of the robot constructed by V-REP. The co-simulation of ROS and V-REP is constructed by vrep_ ros_ bridge. The algorithm outputs the corresponding action instruction to the controller and adjusts the basic gait in real-time. The robot records one round for each fall or when the robot

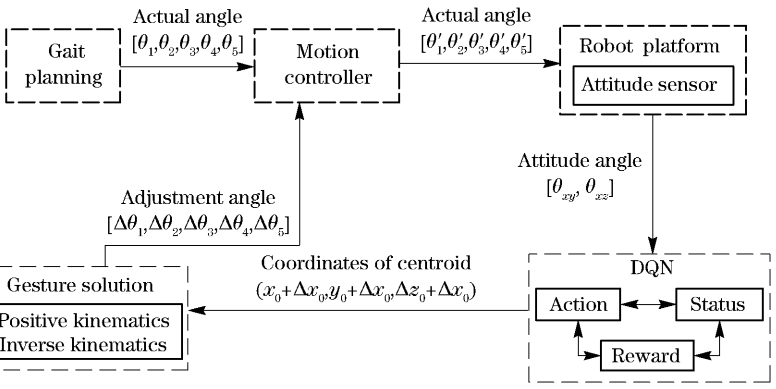

Gait planning
Actual angle $[\theta_1,\theta_2,\theta_3,\theta_4,\theta_5]$
Motion controller
Actual angle $[\theta'_1,\theta'_2,\theta'_3,\theta'_4,\theta'_5]$
Robot platform
Attitude sensor
Attitude angle $[\theta_{xy}, \theta_{xz}]$
Adjustment angle $[\Delta\theta_1,\Delta\theta_2,\Delta\theta_3,\Delta\theta_4,\Delta\theta_5]$
Gesture solution
Positive kinematics
Inverse kinematics
Coordinates of centroid $(x_0+\Delta x_0,y_0+\Delta x_0,\Delta z_0+\Delta x_0)$
DQN
Action
Status
Reward

Fig. 3 Learning process of walking controller

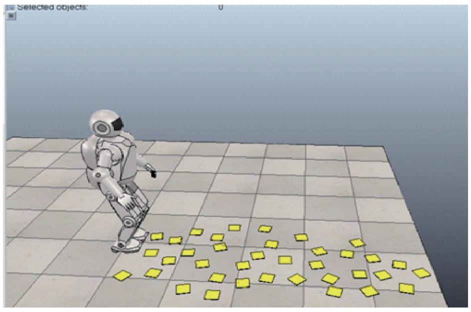


Fig. 4 Simulation of experimental environment

successfully reaches the end point.

3.2 Algorithm parameter design

The adjustment of the algorithm parameters plays an important role in effective learning and efficiency. The learning rate α (update step length) reflects the speed at which the value function reaches the optimal value. If the value of the learning rate is too small, the efficiency and convergence will be insufficient. If the value of the learning rate is too high, the cost function will oscillate, missing the optimal value, and probably will not be converging. In this study, the learning rate is selected as 0.001.

The reward/punishment discount factor γ takes values between 0 and 1. The learning effect needs to consider not only the current reward, but also the future return. The closer the reward is to the current time, the higher the degree of consideration. After experimentation, this article sets the discount factor to 0.9.

The ε-greedy method randomly selects an action with the probability of ε, then selects the current optimal action with a probability of $1-\varepsilon$. The current average reward value for each action and the number of times it is selected is recorded, and updated incrementally. The general value of ε is small, and the value used in this study is 0.01.

Tab. 2 DQN algorithm related parameters

Parameter	Learning rate/α	Discount factor/γ	Exploration probability/ε	Memory updating step/t	Initial experience step/s
Value	0.001	0.9	0.01	100	1 000

As mentioned above, previous experiences are stored in the memory library. Each time the DQN is updated, previous experiences can be randomly selected for learning. The random sampling method can reduce correlation between different experiences to improve the update efficiency of neural networks. The relevant parameters of the algorithm are selected as shown in Tab. 2 based on experience and hardware requirements.

3.3 Simulation training process

In this study, the following three scenarios were simulated for a typical walking situation in order to verify the effectiveness of the algorithm. ① The robot walks on flat ground and there are no uneven ground obstacles in the direction of off-line gait planning; ② The robot walks on uneven ground, the obstacle thickness is 3 mm without DQN algorithm control; ③ The robot walks on uneven ground, the obstacle thickness is 3 mm with DQN algorithm control. Changes in the pitch angle from the walking process are recorded in the three simulation conditions.

As shown in Fig. 5, A1 – A4 depict the walking conditions of the robot on uneven ground when the DQN algorithm is not used. A1 is the initial state of the robot, A2 shows that the robot can walk normally when it does not touch an obstacle and A3 – A4 shows that the robot fell because it did not adjust the centroid. Then, using the DQN algorithm for posture adjustment, the robot was trained for 360 rounds (in which the robot fell or reached the end of a round). The final training results are shown as B1 – B4. B1 is the initial state, B2 – B4 shows that the robot can still walk steadily to the end even it encounters obstacles.

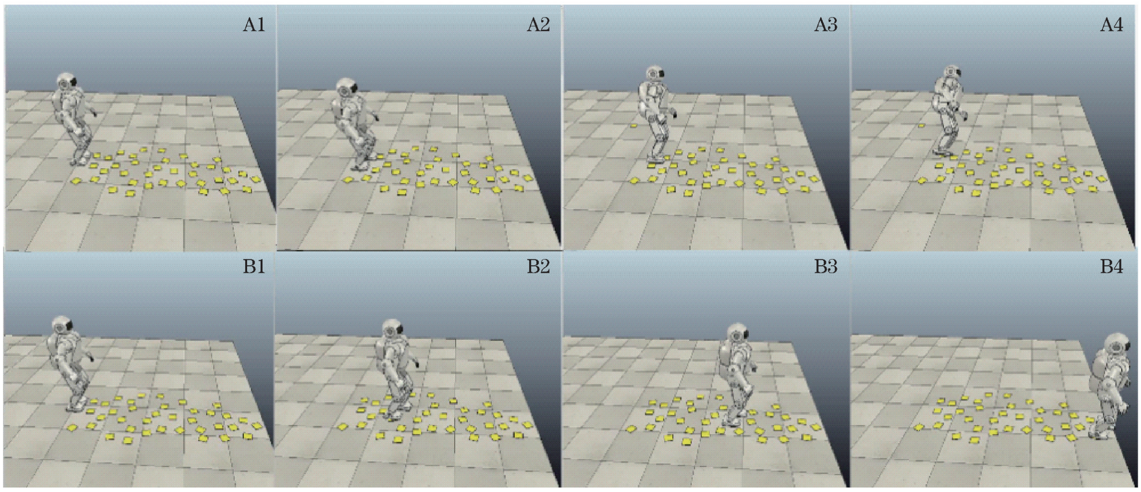
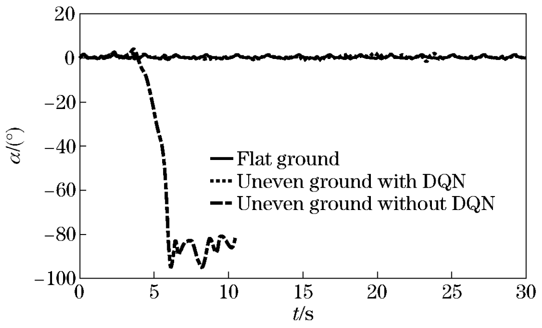


Fig. 5 Key frame contrast of simulation results

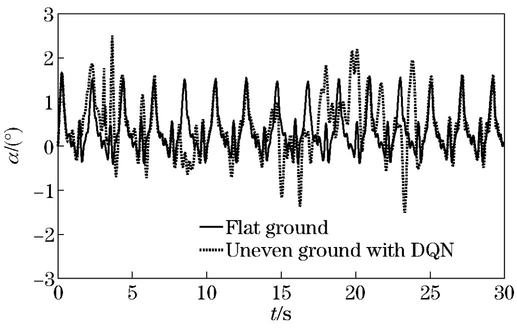
3.4 Result analysis

Fig. 6 and Fig. 7 are the pitch angles of the front-rear direction and the left-right direction of the robot during walking. Fig. 6a, Fig. 7a in both figures depict the data from the three simulation situations. Fig. 6b, Fig. 7b is a comparison of the two situations on "flat ground" and "uneven ground with the DQN algorithm".

From Fig. 6 and Fig. 7, it can be seen that when walking on flat ground, the robot's left and right posture angle α fluctuate within 2°, and the front and rear posture angle β fluctuate within 0.5°; When the walking process encounters uneven ground, the pitch angles of the left-right direction fluctuate around 90° without the DQN algorithm for pitch adjustment. When the pitch angle of the front-rear direction changes from 10° to 20°, the robot falls. After learning through the DQN algorithm, the angle only fluctuates within a 2° range when encountering uneven ground. Furthermore, the extreme positions only exhibit fluctuations of about 3°. The fluctuation range is in

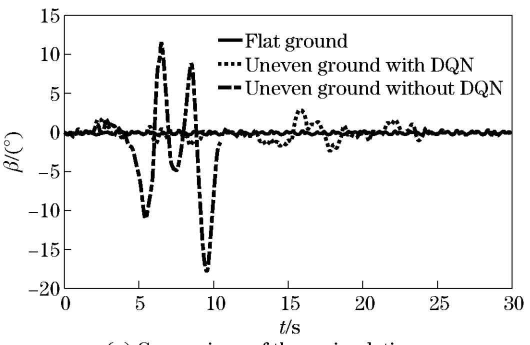


(a) Comparison of three simulation cases

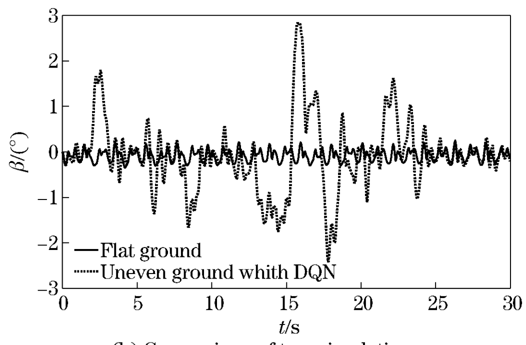


(b) Comparison of two simulation cases

Fig. 6 Robot left and right attitude data curve



(a) Comparison of three simulation cases



(b) Comparison of two simulation cases

Fig. 7 Robot before and after attitude data curve

an acceptable range compared to Ref. [5], which uses ZMP control within 2° of the fluctuation range of the attitude angle of the uneven road surface, which indicates that the DQN algorithm of the biped robot can be achieved on an uneven road surface through the adjustment of the centroid position.

After 360 rounds of training, the data of the loss function and reward value function of the DQN algorithm is obtained. Fig. 8 shows the change in the loss function during the entire training process. It can be seen that the loss function gradually decreases with the increase in training steps until becoming steady.

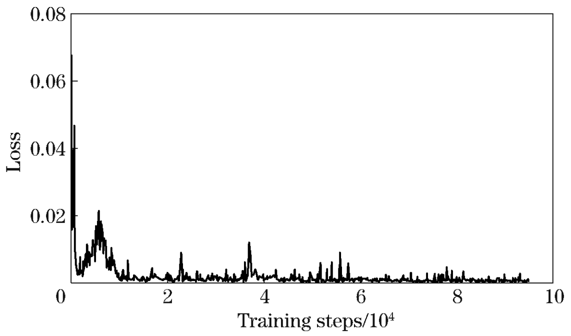


Fig. 8 Error curve of the loss function

Through the analysis of the above simulation results and experimental data, it can be seen that the loss function eventually converges and the cumulative reward converges to a stable value. The robot learned to walk steadily on uneven ground using the DQN based walking controller.

4 Conclusion

This paper presents a new walking stability control method for biped robots based on deep Q-network to address the fact that biped robots tend to lose their stability when walking on uneven ground. This work mainly includes two parts: ① generation of off-line gait; ② gait adjustment algorithm based on DQN algorithm. In the V-REP environment, an uneven ground simulation environment is built. The simulation results show that after the robot encounters uneven ground, most of the robot's fluctuations in posture are within 2° after the algorithm's adjustment, with fluctuations of about 3° at the most extreme positions. This shows that the robot has learned to adjust its gait and can maintain its stability. This proves the effectiveness of the algorithm. Compared to the traditional control method, this method does not require complex dynamic model processing. It provides a new reference method to solve the stability problem of the biped robot when walking on uneven ground.

References:

[1] Tan Yantao, Sun Zhongbo, Li Hongyang, et al. A review of optimal and control strategies for dynamic walking bipedal robots [J]. Acta Automatica Sinica,2016, 42(8):1142 – 1157. (in Chinese)

[2] Dang Van Chien, Ki Je Sung, Jong-Wook Kim. Sensory reflex control of a humanoid robot using FSR sensor[C]//IEEE International Conference on Advanced Intelligent Mechatronics, IEEE, 2015:1406 – 1409.

[3] Kim J W, Tran T T, Dang C V, et al. Motion and walking stabilization of humanoids using sensory reflex control[J]. International Journal of Advanced

Robotic Systems, 2016, 13(2):1.

[4] Chen Guangrong, Wang Junzheng, Wang Liping. Gait planning and compliance control of a biped robot on stairs with desired ZMP[J]. IFAC Proceedings Volumes, 2014, 47(3):2165 – 2170.

[5] Li Jian, Chen Weidong, Wang Lijun, el at. Stability control for biped walking on unknown rough surface [J]. Acta Electronica Sinica, 2010, 38(11):2669 – 2674. (in Chinese)

[6] Sasaki H, Horiuchi T, Kato S. A study on behavior acquisition of mobile robot by deep Q-network[J]. Journal of Advanced Computational Intelligence & Intelligent Informatics, 2017, 8(4):727 – 733.

[7] JAFRI Ali Raza, Huang Qiang, Yang Jie. et al. Motion planning of humanoid robot for obstacle negotiation[J]. Journal of Beijing Institute of Technology, 2008, 17(4):439 – 444.

[8] Gu Shixiang, Holly Ethan, Lillicrap Timothy, et al. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates [C] // IEEE International Conference on Robotics and Automation, IEEE, 2017:3389 – 3396.

[9] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning [J]. Nature, 2015, 518(7540):529.

[10] He Yudong, Wang Junzheng, Ke Xianfeng, et al. Stable walking for legged robots [J]. Journal of Mechanical Engineering, 2016, 52(21):1 – 7. (in Chinese)

[11] Zhang Bin. Joint-space Trajectory planning for robots under multiple constraints [J]. Journal of Mechanical Engineering, 2011, 47(21):1 – 6. (in Chinese)

[12] Wang Lipeng, Wang Junzheng, Zhao Jiangbo, et al. Foot trajectory generation and gait control method of a quadruped robot on uneven terrain based on zero moment point theory[J]. Transactions of Beijing Institute of Technology, 2015(6): 601 – 606. (in Chinese)

[13] Yang Jie, Huang Qiang, Li Jianxi, et al. Walking pattern generation for humanoid robot considering upper body motion [C] // IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2007:4441 – 4446.

[14] Watkins C J C H, Dayan P. Q-learning[J]. Machine Learning, 1992, 8(3 – 4):279 – 292.

[15] Liu Yiping, Wensing P M, Orin D E, et al. Dynamic walking in a humanoid robot based on a 3D Actuated Dual-SLIP model[C] // 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2015: 5710 – 5717.

(**Edited by** Jianying Cai)