The International Journal of Advanced Manufacturing Technology (2022) 122:1099–1111
https://doi.org/10.1007/s00170-022-09800-1

APPLICATION

Learning visual path–following skills for industrial robot using deep reinforcement learning

Guoliang Liu[1] · Wenlei Sun[1] · Wenxian Xie[1] · Yangyang Xu[1]

Received: 29 January 2022 / Accepted: 14 July 2022 / Published online: 17 August 2022

Abstract

The visual path–following technology is widely used in cutting, laser welding, painting, gluing, and other fields, which is a crucial content of robotics studies. As an important algorithm of artificial intelligence (AI), reinforcement learning provides a new insight for robots to learn path-following skills which has the ability of machine vision and decision making. In order to build a robotic agent with path-following skills, this paper proposes a visual path–following algorithm based on artificial intelligence deep reinforcement learning double deep Q-network (DDQN). The proposed approach allows the robot to learn path-following skill by itself, using a visual sensor in the Robot Operating System (ROS) simulation environment. The robot can learn paths with different textures, colors, and shapes, which enhances the flexibility for different industrial robot application scenarios. Skills acquired in simulation can be directly translated to the real world. In order to verify the performance of the path-following skill, a path randomly hand-drawn on the workpiece is tested by the six-joint robot Universal Robots 5 (UR5). The simulation and real experiment results demonstrate that robots can efficiently and accurately perform path following autonomously using visual information without the parameters of the path and without programming the path in advance.

Keywords Robot · AI · DDQN · Machine vision · Path following · Real experiment

1 Introduction

Along with the rise of the Fourth Industrial Revolution [1] and the era of artificial intelligence (AI) [2], the development of the manufacturing industry is ushering in a new opportunity for rapid upgrading. Robots equipped with a variety of sensors play an important role in the manufacturing industry [3]. Robots are entering a new era of Robot 4.0 from the era of Robot 3.0. Understanding the environment and autonomic decision-making will allow the robots to autonomously perform tasks in the era of Robot 4.0 [4]. Robots and agents are evolving towards having and even surpassing human capabilities in some fields [5]. However, teach-payback and offline programming [6, 7] are still the main programming methods in the industrial robot field. In the teaching and payback mode, the robot just repeats the actions programmed by the workers in advance. The perception and decision-making in the process of teaching programming are done by workers. It is suitable for simple and regular parts. In addition, when the parts are complicated, the teach-payback programming is time-consuming and error-prone. In the offline programming mode, a commercial programming software is used for visual programming based on the three-dimensional model of parts. The reverse reconstruction of the three-dimensional model of parts is very complicated. A position error exists between the part in the real environment and the programming simulation environment. Therefore, offline programming is not efficient. Artificial intelligence reinforcement learning has surpassed the level of human in gaming competitions [8], in which the agent can autonomously learn game skills from experience, can understand game images, and autonomously decide to win the game. Inspired by this breakthrough, reinforcement learning is used to allows the robots to learn visual path following skills. The robots can enhance their skills by continuously learning from experience using reinforcement learning [9], just like humans do.

Based on the research mentioned above, a six-axis robot path–following algorithm based on DDQN reinforcement

✉ Wenlei Sun
sunwenxj@163.com

Guoliang Liu
907379023@qq.com

[1] School of Mechanical Engineering, Xinjiang University, Urumqi 830017, China

learning was proposed in this paper. A visual attention–focused observation method which can extract important image features is proposed. A reward function and action configurations suitable for path following are then investigated. The robot operating system (ROS) is considered to develop a virtual simulation environment for robot skill learning. Robots start training without knowing anything and become proficient in path-following skills by the end of the training. The policy learned in the simulation environment can be directly translated to the real world. The UR5 robot can perform never-before-seen curved path–following using skills learned in a simulation environment.

1.1 Related works

In recent years, several researchers tackled the intelligent path planning methods based on sensor perception. For instance, a novel automated offline programming method using a depth camera is proposed in [10], which can automatically generate paths and reduce costs and integration time. In [11], a camera is used to detect the damaged area of the part. The robot then repairs the parts based on the information collected by the camera. The weld seam is identified and detected using global information about the workpiece acquired from the camera [12, 13]. A hybrid sensor is designed to extract the point cloud information of the part. It is successfully applied to the welding operation as shown in [14]. A structured light vision system which can identify the welding seam feature is designed in [15]. These algorithms greatly improve the programming efficiency and robot's perception capability, and sensors in those algorithms can be provided to the vision system for the robot to learn skills. Deep reinforcement learning has high performance in dealing with nonlinear problems, since it takes full advantage of deep learning and reinforcement learning [16]. Deep reinforcement learning has the ability to directly learn behavior from high-dimensional sensor information. It is important to combine robotics and artificial intelligence to make robots have the ability to learn skills. Three robot skills learning approaches exist: reinforcement learning, demonstration learning, and few data learning methods [17].

In [5], the deep reinforcement learning deep Q network (DQN) algorithm is proposed. The agent is trained to perceive images, make decisions, and beat professional human players. Significant breakthroughs in other games can be found in [18–20]. Since then, artificial intelligence reinforcement learning has become a crucial research topic in the field of robotics, and has greatly contributed to the development of intelligent robots. A robotic agent can stably play air hockey game based on the DQN algorithm [21]. The robotic agent learns the synergies between grasping and pushing after training, using reinforcement learning algorithms in the sorting task [22]. Similar results are shown in [23], where a robotic agent is learned to stack cubes. Reinforcement learning has also been successfully applied to the fields of assembly tasks, as shown in [24, 25]. A mobile robot performs automatic obstacle avoidance navigation in a complex environment using reinforcement learning [26]. Reinforcement learning is used to search for the optimal processing parameters during the processing [27]. In [9], a deep autoencoder neural network is used in reinforcement learning, where the robot gains the skill of throwing the ball to the target. The six-axis industrial robot autonomously plays the wire loop game without knowing the model of the wire or programming the robot motion beforehand, using reinforcement learning [28]. In reinforcement learning, sensors are used to sense the environment and the neural network of robot is used for decision-making. In addition, the robot can quickly adapt to the environment and independently perform the task. The reinforcement learning algorithms are highly adaptable and intelligent, which has led to a large number of applications and rapid development of reinforcement learning. Therefore, applying reinforcement learning algorithms to robotics can provide robots high adaptability, autonomy, and intelligence.

The main contributions of this paper are summarized as follows. The artificial intelligence reinforcement learning DDQN algorithm is used to solve the path following problem of industrial robots. The robotic agent can learn visual path following skills by itself in the simulation environment. During the training process, the robotic agent can improve its visual perception and decision-making ability by practicing. Skills acquired in simulation can be directly transferred to the real world. The robots can autonomously achieve efficient and accurate path following without the need to know the parameters of the path and program it in advance. This method allows the robot to learn path-following skills for the path with different colors, shapes, and textures, and it can be applied to areas such as laser cutting, laser welding, painting, and gluing, for example.

2 Path-following method-based DQN

2.1 Deep Q-network

In the reinforcement learning algorithm, the agent learns control strategies by interacting with the environment. The Markov decision process (MDP) describes the environment with transition experiences $e_t = (s_t, a_t, r_t, s_{t+1})$, where s_t represents the current image state observation, a denotes the action, r_t is the reward, and s_{t+1} represents the next image state observation. The robotic agent learns path-following control policies through interacting with the part, by trial and error in the simulation environment. The robotic agent gets an image state observation s_t of the path by the camera,

as shown in Fig. 1. A detailed description about s_t is provided in Sect. 3.1. The robotic agent decides which action should be taken according to the current state observation, observes a new physical state observation s_{t+1} after outputting action (a_t), and receives a scalar reward $r_t = r(s_t, r_t)$ from the environment. The ultimate goal of the robotic agent is to maximize the sum of rewards $R_t = \sum_1^T r_t$ obtained from interaction with the environment.

The end effector of the robotic agent continuously moves along the path with the orientation and position, as show in Fig. 1. There are several challenges for the robot to learn path-following skills, using reinforcement learning. Firstly, the visual observation of the path is variable and complex, and the reinforcement learning Q-learning algorithm does not have good convergence properties in this case, because it has a limited number of tables. In addition, a strong correlation exists between state observations, which makes the network difficult to converge. Mnih et al. [5] propose a DQN model including experience replay mechanism and Q-target network, which can solve the previously mentioned problems. The framework of a deep Q-network-based visual path following is shown in Fig. 2. DQN uses a neural network to approximate the action value function. DQN has a dual neural network design with the same structure. The first one is a deep Q network for learning while the second one is a target network for predicting Q values. After a period of study, the parameters of the target network are replaced by the parameters of the deep Q network, which interrupts the correlation of learning. A lot of experience is stored in the experience replay buffer. The experiences are randomly extracted from the experience replay buffer, and then used to update the parameters of the neural network, which interrupts the correlation between the state observations:

$$y = r + \gamma \max_a Q(s_{t+1}, a; \theta_t^-) \tag{1}$$

$$L_t(\theta_t) = E_{(s,a,r,s_{t+1})}[(y - Q(s, a; \theta_t))^2] \tag{2}$$

$$\nabla_{\theta_t} L_t(\theta_t) = E_{(s,a,r,s_{t+1})}\left[(y - Q(s, a; \theta_t))\nabla\theta_t Q(s, a; \theta_t)\right] \tag{3}$$

$$\theta_{t+1} = \theta_t + \alpha(\mathrm{y} - \mathrm{Q}(s_t, \mathrm{a}_t; \theta_t)\nabla_{\theta_t} Q(\mathrm{s}_t, \mathrm{a}_t; \theta_t) \tag{4}$$

Equation (1) presents the target function for the path following learning, where θ_t^- is the parameter of the target network and ϕ_i represents the discount factor. The loss function of the network is expressed in Eq. (2), where θ_t is the parameter of the deep Q network. Equation (3) presents the derivative function of Eq. (2). It is used to update the parameters of the neural network according to the back-propagation principle. The parameters of the neural network are calculated using Eq. (4), where ϕ_i is the learning rate.

2.2 Double DQN

DQN has solved several engineering problems since it was proposed. With the depth application and development of DQN, many improved versions have been proposed. For instance, Hasselt et al. [29] propose the double DQN (DDQN) algorithm inspired by the double Q-learning algorithm in order to improve learning efficiency. The target function Eq. (6) is replaced by Eq. (5).

$$y^{DQN} = r_{t+1} + \gamma Q(s_{t+1}, \underset{a}{\mathrm{argmax}} Q(s_{t+1}, a; \theta_t); \theta_t) \tag{5}$$

$$y^{DDQN} = r_{t+1} + \gamma Q(s_{t+1}, \underset{a}{\mathrm{argmax}} Q(s_{t+1}, a; \theta_t); \theta_t^-) \tag{6}$$

A large number of transition experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ are stored in an experience replay buffer, and randomly selected for parameter updating of the network. Due to the different importance of experiences to the network, the learning efficiency is not high in the process of network learning. In order to solve this problem, Schaul et al. [30] propose the prioritized experience replay mechanism, which can more efficiently manage transition experiences, and further improve the learning efficiency. Based on this analysis, the prioritized experience replay mechanism and the DDQN algorithm are combined to achieve path following strategy learning. The framework of the algorithm is shown in Fig. 2.

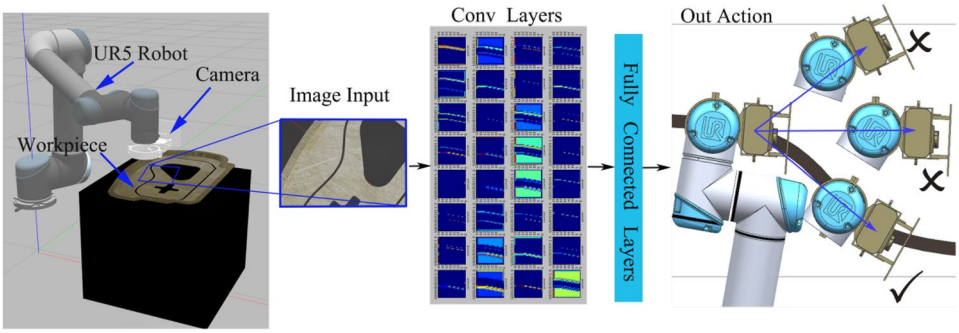

Fig. 1 Principle of path-following method-based DDQN

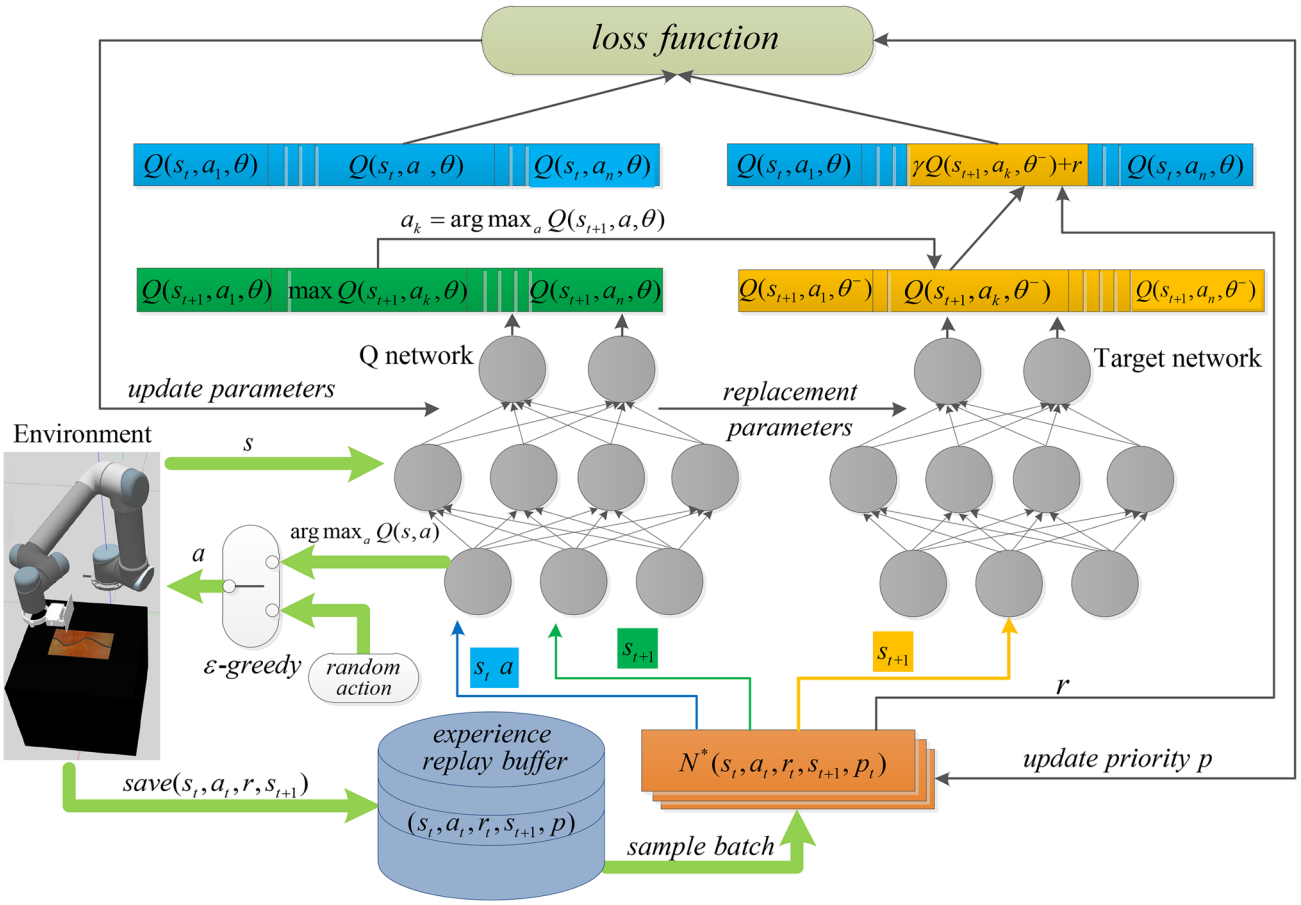

Fig. 2 The framework of DDQN with the prioritized experience replay mechanism

3 Path-following method

3.1 State observation

The workspace of the robot containing the workpiece is a complex high-dimensional space. The full-state observation of the workspace by the sensor produces a large amount of input information, and consumes lot of memory and computing time. If a large amount of useless information is observed, the robotic agent even fails to learn the path-following strategy. Therefore, how to eliminate the useless information and retain the useful information is the focus of the state observation. Workers are skilled in path-following operations. In the process of learning the path-following operation, they focus on the intersection of the end effector and the path. After a period of practice, they skillfully master the skill with the help of vision systems. Therefore, the camera is used to collect image information for the robot. In order to perform the observation mode similar to that of workers, the installation mode of the camera in hand is considered. It can be assumed that the tool center point (TCP) coincides with the center of the camera image, so that the region of interest of the state observation is always concentrated around TCP. Consequently, the camera observation center can be considered as the reference point for path following. Since the area of interest observed by workers has a certain range, the 36×36 pixel section around the center of the grayscale image is considered as the state observation space s_t, as shown in Fig. 3.

3.2 Action space

The training environment is shown in Fig. 5. The trial-and-error method is used to train the robotic agent to learn the control strategy, which controls the motion of the TCP of the manipulator along the plane trajectory with the position and orientation. The deep Q network outputs the Q value ($Q(s_i, a_i)$) of each action after receiving the state observation of the path, as shown in Fig. 3. The action corresponding to the maximum Q value is considered as the output action. The movement of the TCP with the position and orientation can be computed as:

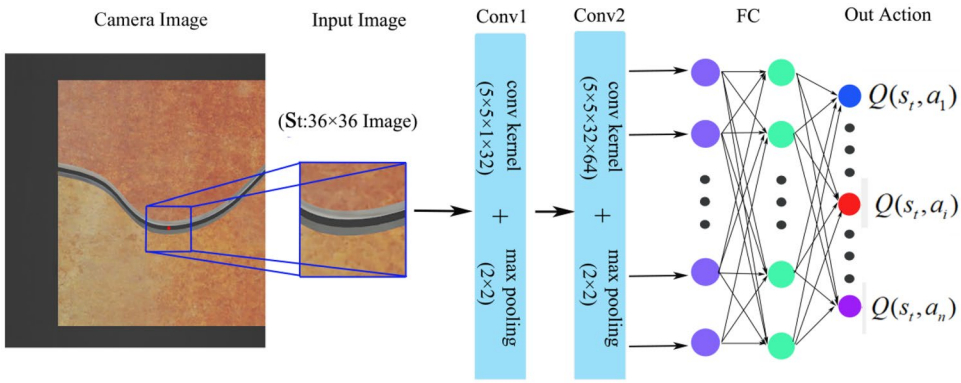

Fig. 3 The architecture of a deep Q network

$$\begin{cases} x_{i+1} = x_i + lcosa_i \\ y_{i+1} = y_i + lsina_i \\ z_{i+1} = z_i \end{cases} \quad (7)$$

13 possible actions being intervals of 10 from -60 to 60

where a_i denotes the directional angle of the TCP movement. In Eq. (7) $a_i \in [-60, -50, -40 \ldots 0 \ldots 40, 50, 60]$. The number of actions is 13. The maximum output angle of the absolute action is 60°, and the subdivision angle is 10°. x_i, y_i, z_i is the current position of the TCP, while $x_{i+1}, y_{i+1}, z_{i+1}$ is the position of the TCP after the robotic agent takes action, and the step size is $l = 2mm$.

3.3 Reward function

The robotic agent learns visual path–following control strategies by exploring the path from the starting point to the end point, as shown in Fig. 5, and a large number of experiences (s_t, a_t, r_t, s_{t+1}) is gathered in the experience replay buffer during the exploring. By maximizing the sum of reward mechanism, the parameters of the DDQN are updated to form the optimal action output control strategy using those experiences, as shown in Fig. 3. The reward function is used in the process of visual path–following skill learning. After finishing the learning, the robot uses the learned visual control strategy to complete path following without using the reward function. The TCP of the robot moves along a complex path. The closer the TCP is to the center of the path, the more reward the robot should get. Assuming that there are six possibilities for the action output of the robotic agent, as shown in Fig. 4a, the TCP will reach different positions when different actions are selected from the action space. The reward function is set using the distance from the point of TCP to the center line of the path. The center line of the path is discretized into a point cloud, as shown in Fig. 4a, b. The closest point on the center line of the path to TCP can be searched using the K-dimensional tree (Kdtree) method, so the distance between TCP and the target path can be computed as:

$$l(s_t, a_t) = \sqrt{(x_{ti} - x'_{pi})^2 + (y_{ti} - y'_{pi})^2 + (z_{ti} - z'_{pi})^2} \quad (8)$$

where point x_{ti}, y_{ti}, z_{ti} is the position of the TCP, point $x'_{pi}, y'_{pi}, z'_{pi}$ is the point on the target path closest to point TCP, as shown in Fig. 4c, d. In industrial robotics applications, there are different paths that have different shapes, widths, colors, and other appearance features. The method can calculate the distance for different paths when taking different actions. The closer TCP is to the path centerline, the smaller $l(s_t, a_t)$ is. As shown in Fig. 4b, $l(s_t, a_1) > l(s_t, a_2) > l(s_t, a_3)$. Therefore, the final reward function is defined as:

$$r(s_t, a_t) = \begin{cases} 50 + \sum_0^t 1 - 10 \times l(s_t, a_t) \; l(s_t, a_t) < K \\ -100 \; l(s_t, a_t) \geq K \\ +100 \text{ end point} \end{cases} \quad (9)$$

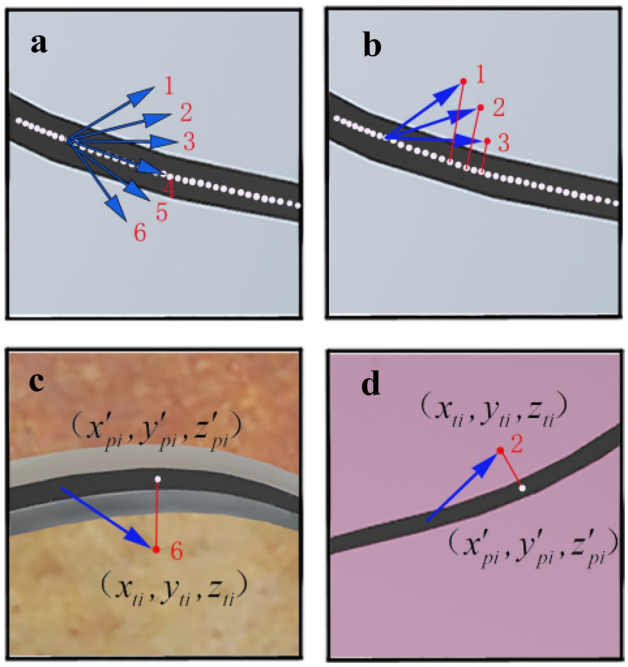

Fig. 4 Principle of convolution reward function

Table 1 List of hyper-parameters used by the deep Q network

Parameter	Description	Value
Learning rate	α	0.0001
Discount rate	γ	0.9
$\varepsilon-greedy$ exploration probability	ε	0.5
Parameter replication cycle	n	10
Size of the experience replay buffer	m	4000
Batch size sampled from buffer	Mini-batch	64

where the first component is equal to 50 which is a fixed reward, the second component $\sum_0^t 1$ is a reward which encourages the robotic agent to move forward along the path, the third component $-l(s_t, a_t)$ is a reward for following accuracy which indicates how far the TCP is from the centerline of the path, and K is a threshold (set to 0.7) used to prevent the TCP from moving away from the path during the training process. The robot will be punished by -100 when $l(s_t, a_t)$ is greater than K. Finally, the robotic agent receives a positive reward of 100 when TCP reaches the end point.

3.4 Network settings

Convolutional neural networks [31] can extract features from high-dimensional information such as sound signal, and image information, for example. It is widely used in computer vision fields such as image object recognition and tracking, image segmentation, and registration [32]. In this paper, a convolutional neural network is used to process the images of path. The overall architecture of the DDQN is shown in Fig. 3. The input image, which is captured by a camera mounted at the end of the manipulator, is preprocessed to $36 \times 36 \times 1$ image state observation s_t. Afterwards, the observation s_t is fed into the DDQN which consists of convolution layers and fully connected layers. The convolution layers are used to extract features from the input image, while the fully connected layers are used to map the relationship between image features and output actions. The hyper-parameters in the deep Q network are shown in Table 1.

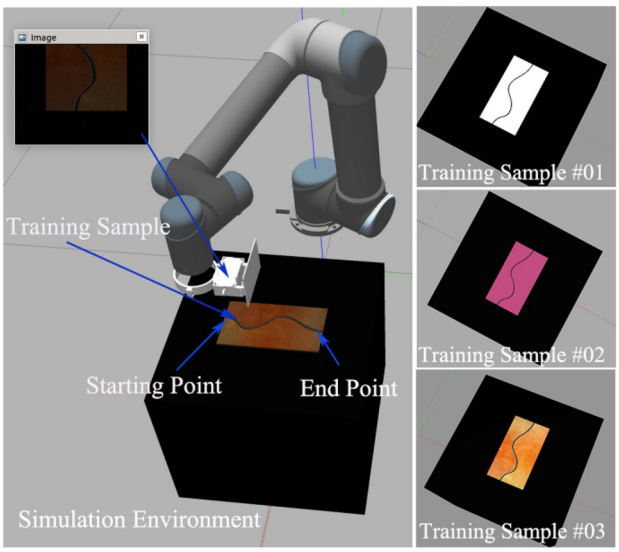

Fig. 5 Path-following simulation environment

3.5 Exploration

The robotic agent explores the path from the starting point to the end point, as shown in Fig. 5. A two-stage training method is used to solve the issue of exploration. In the first stage, the $\varepsilon-greedy$ method is used to explore the environment. The robotic agent chooses the action based on Eq. (10). A random action is then chosen from the action space when the random probability is less than ε. The deep Q-network determines how to select action when the random probability is greater than ε. The initial value of ε is set to 0.5, and then decreases with the exploration times increase.

$$a_t = \begin{cases} random(a_i), & \varepsilon \\ \arg\max Q(s_t, a), & 1-\varepsilon \end{cases} \tag{10}$$

The robotic agent has the ability to control its TCP following along the path after the first stage of training. However, the following accuracy is not high. To further refine the accuracy, an improved training method based on the normal distribution principle is proposed. The robot continues to explore the environment using Eq. (11) in the second stage.

$$a_t \sim N(\mu, \sigma^2) = N(\arg\max_a Q(s_t, a_t)\sigma^2) \tag{11}$$

The robot selects action based on the normal distribution, which reduces the blindness of $\varepsilon-greedy$ exploration, and continues to improve the decision-making ability based on what has been learned in the first stage. The initial value of the standard deviation σ is set to 1. It then decreases with the exploration steps increase. The normal distribution mean μ is equal to $\arg\max_a(s_t, a_t)$.

4 Training and experiment

4.1 Simulation environment

The robotic agent gathers experiences as it explores its environment, and learns from those experiences. Two

approaches exist to collect experience information. The first one consists in collecting experience information from the simulation environment, and the second one consists in collecting experience information from the real environment. The first approach is relatively feasible and convenient, because collecting information is expensive and dangerous when the robotic agent explores the real environment. Therefore, the robot is first trained to learn path-following skills in a simulation environment, and the learned skills are then transferred to the real environment. A virtual simulation platform based on ROS [33] is built, as shown in Fig. 5. A virtual robot system consisting of an RGB camera, a UR5 robot, a workbench, and workpiece samples is built in the Gazebo simulation environment. PyCharm is used as the programming software, while Python is used as programming language. TensorFlow is used for the construction of the neural network, and Moveit is used to control the UR5 robot.

Different industrial robot tasks, such as laser cutting, welding, and gluing, possess paths with different appearance characteristics. In order to increase the flexibility of the robot, the robot should have the ability to learn path-following skills directly from path images with different textures, colors, and shapes through training. Three training samples with different appearance characteristics represent three different task skills learning, as shown in Fig. 5. Skill 1 is learned from training sample 1. Skill 2 and skill 3 correspond to training sample 2 and training sample 3, respectively. Training sample with a complex B-sample curve is used to for skill learning. Test samples with different B-sample curves from those of the training samples are used to verify the performance of the learned skill.

4.2 Training results

The TCP of the robot is set at the beginning of the path. The robotic agent uses the $\varepsilon-greedy$ policy (cf. Sect. 3.5) to explore the path from the starting point to the end point, as shown in Fig. 5. The robot gets the current image state observation (s_t) through the camera, outputs the action (a_t), and reaches the next state (s_{t+1}). If the robot receives a positive reward (r_t), it continues to follow the path. Otherwise, it returns and explores again. The experiences (s_t, a_t, r_t, s_{t+1}) generated during exploration are stored in the experience replay buffer and used to update the deep Q network. The robot's DDQN learns from experience while exploring the environment.

The learning principle flow of the DDQN with the prioritized experience replay mechanism is shown in Fig. 2. A large number of experiences enter the DDQN. The DDQN learns the experience by updating the parameters using Eq. (4),where the learning rate has a large impact on the efficiency. The learning efficiency of DDQN at training sample #01 is compared with different learn rates, as shown in Fig. 6. It can be seen that the cumulative rewards of DDQN-PE with learn rate 0.0001 (DDQN-R-0.0001) are the fastest, that of DDQN with learn rate 0.00001 (DDQN-R-0.00001) is the second, and the cumulative rewards of DDQN with learn rate 0.001 (DDQN-R-0.001) is not

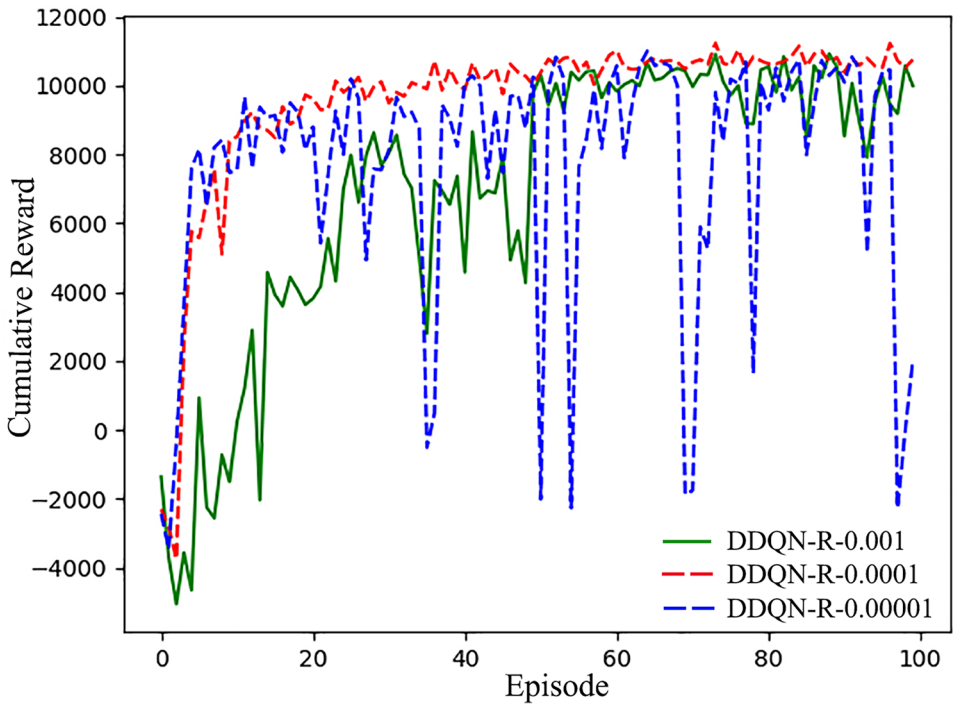

Fig. 6 Reward at different learn rates

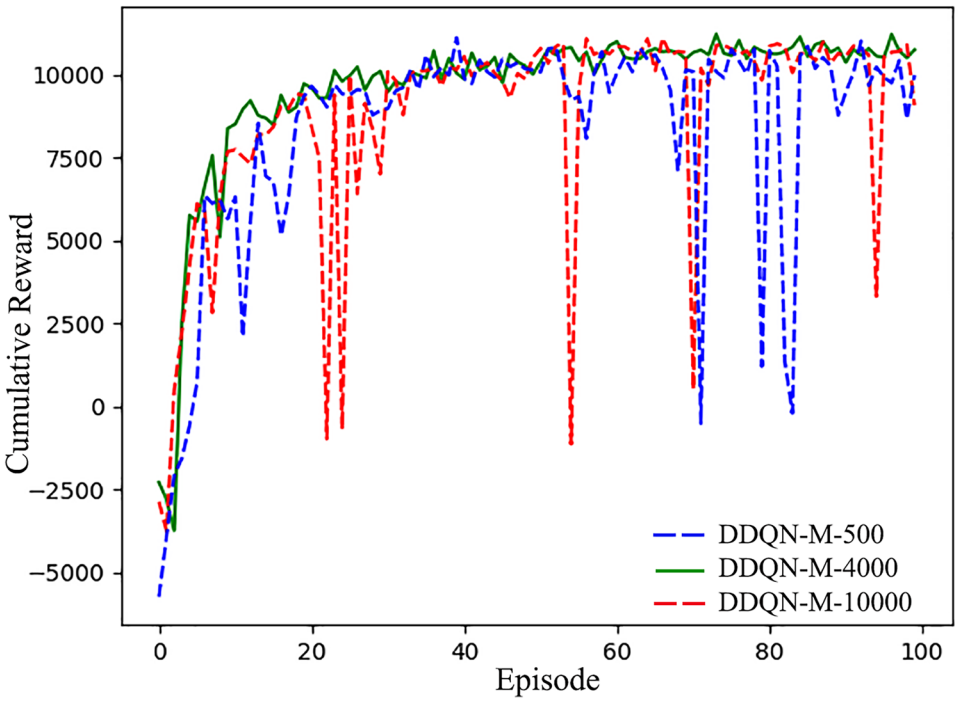

Fig. 7 Reward at different size of the experience replay buffer

stable. DDQN-R-0.0001 can quickly and consistently get rewards, which means the robot gradually learns how to output actions based on the path image from the beginning point to the end point on the path. DDQN-R-0.00001 is not efficient, which indicates that the robot needs to spend more time to learn the path-following control strategy. DDQN-PE-0.01 is not convergent, which means that the robot cannot learn the path-following control strategy.

Experiences are stored in the experience replay buffer while the robotic agent explores the environment. Some experiences are extracted from the experience replay buffer for updating the DDQN, as shown in Fig. 2. The learning efficiency of DDQN at training sample #01 with three sizes of the experience replay buffer (DDQN-M-500, DDQN-M-4000, and DDQN-M-10000) is compared. Figure 7 shows the comparison results of the cumulative rewards obtained

Fig. 8 Reward for different training task

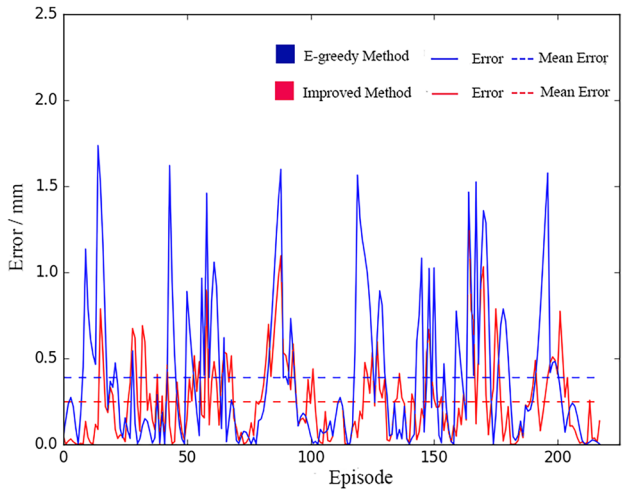

Fig. 9 Training effect of the second stage

by the robot during the training process. It can be seen that the learning performance of DDQN-4000 is more stable than that of DDQN-M-500 and DDQN-M-10000. The capacity of DDQN-M-500 is too small, and the experience replay buffer does not store enough experience, which makes learning unstable. The capacity of DDQN-M-10000 is too large, and the experience replay buffer stores too much experience, which makes learning difficult.

Different industrial robot tasks have paths with different colors, textures, and shapes. Training sample #02 and training sample #03 are used to test the learning performance of the proposed method for different paths, as shown in Fig. 5. Robotic agents learn visual path-following skill 2 and skill 3 from training sample #02 and training sample #03, respectively. The learning results are shown in Fig. 8. The results indicate that the robot can consistently get rewards, which means the proposed approach achieves good learning performance on the above two paths with different appearance characteristics.

The robot continues to be trained for 100 episodes using the improved method in the second stage (cf. Sect. 3.5). In order to assess the training effect of the first and second stages, a path-following testing is conducted on the training sample. Figure 9 shows the error curves. After training, the mean error decreases from 0.3906 to 0.2524 mm. By comparing the errors of the two stages, it can be deduced that the decision-making ability of the robot has been significantly improved using the improved training method. The parameters of the deep Q network are fixed when the training is completed, and the robotic agent completes visual path–following skill learning.

4.3 Test results

In order to verify the performance of the path-following skill, the workpiece with a curve different from the one on the learning sample should be used as a test sample. Therefore, several samples containing random B-sample curves are designed for testing (cf. Fig. 10). Each sample is tested twice in a row. Figure 11 shows the test results of sample #01 using skill 1. The mean error of sample #01 is 0.2647 mm. Figure 12 presents the test results of sample #02 using skill 3. The mean error is 0.2618 mm. These results show that the robot is able to complete path following with a high accuracy on different curves.

4.4 Real experiment

It is shown in Fig. 13 that the experimental platform consists of a control cabinet, a UR5 robot (Universal Robot), a computer, a camera, a laser pointer, and a workpiece. The camera

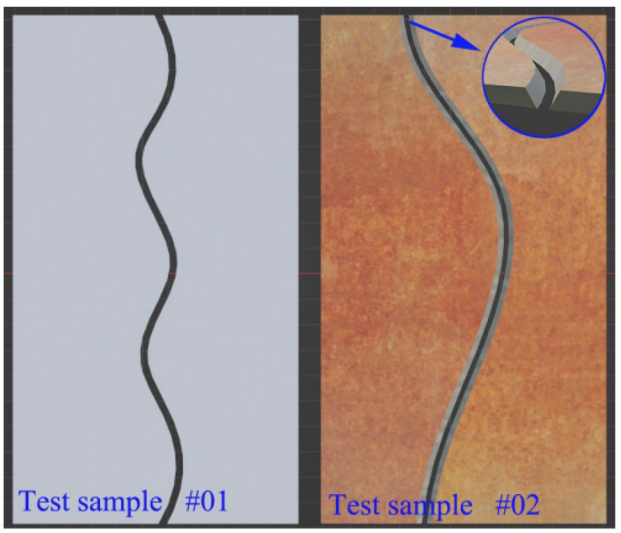

Fig. 10 Test samples

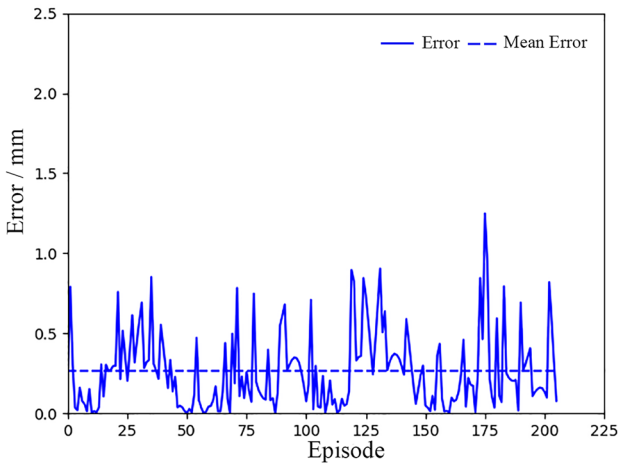

Fig. 11 Test results of sample #01

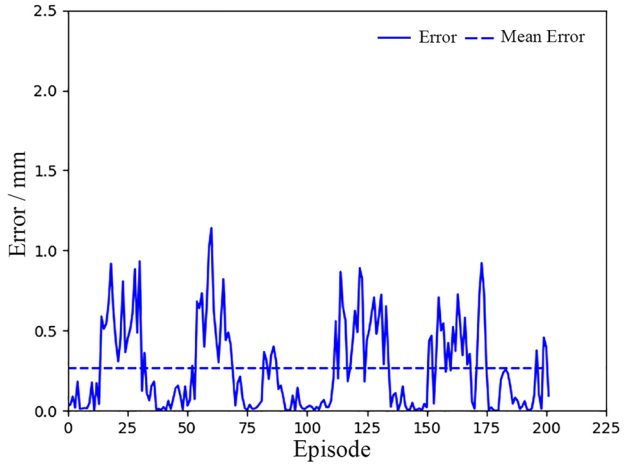

Fig. 12 Test results of sample #02

has a resolution of 640 × 480. The laser pointer mounted on the end of the robot is used to simulate a laser for cutting and laser welding. The laser spot coincides with the center of the camera image when the laser irradiates the workpiece (Fig. 14).

In Sect. 4.2, the robot completes the learning of the path-following strategy in the simulation environment. Afterwards, two random hand-drawn curves on the workpieces are tested on the real robot platform in order to verify the correctness of the path following strategy, as shown in Fig. 14. Experimental sample #01 is tested using skill 1, and experimental sample #02 is tested using skill 2. The observation center of the camera is set at the beginning of the curve before the test. Through ROS message topics, the computer acquires the image information from the camera and sends control commands to the control cabinet in order to control the robot operation. The signal flow diagram is shown in Fig. 13. During the experiment, the robot observes the path through the camera and decides how to output the action through the trained deep Q network. The observation center of the camera will reach a new point on the path after the output action. The robot then continues to observe and follows the path. During the experiment, the coordinates of the TCP are recorded and used as the interpolation points. The experimental results of simulated laser cutting are shown in Fig. 14. They confirm that the robot can efficiently and autonomously complete path following in the real environment. The center of the laser beam is located on the centerline of the path, which indicates that the robot has high path following skills.

4.5 Comparison and discussion

In this section, the proposed approach is compared with other state-of-the-art methods in industrial robot path planning.

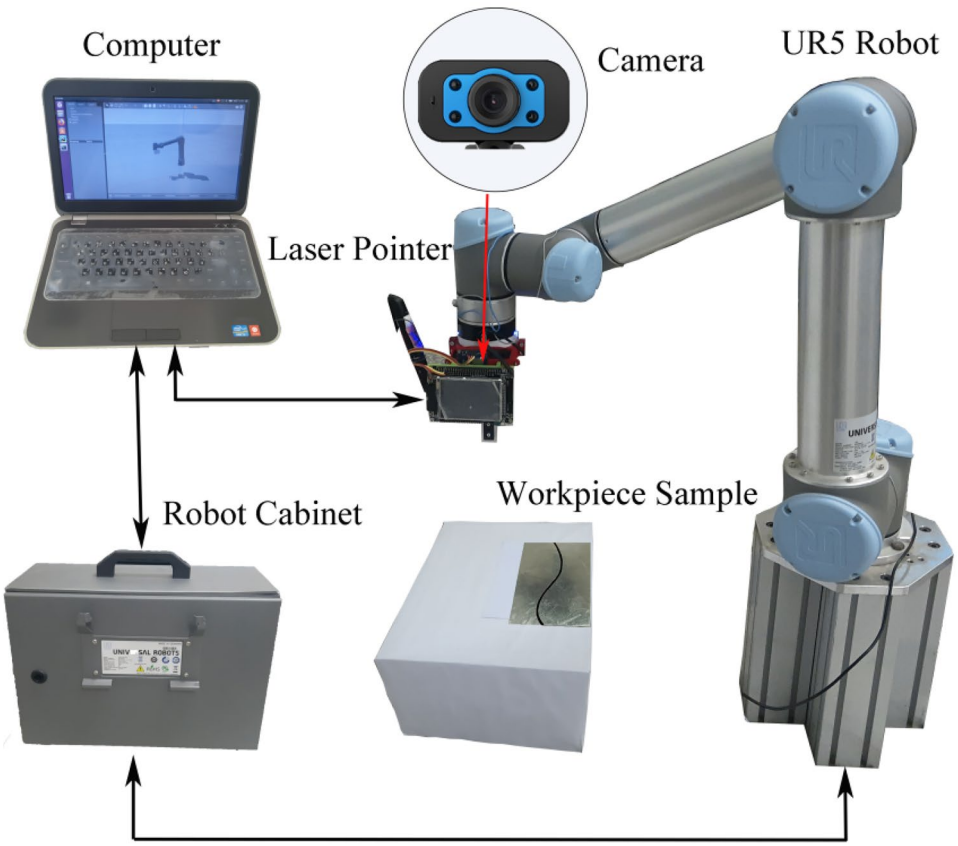

Fig. 13 Physical experimental platform for path following

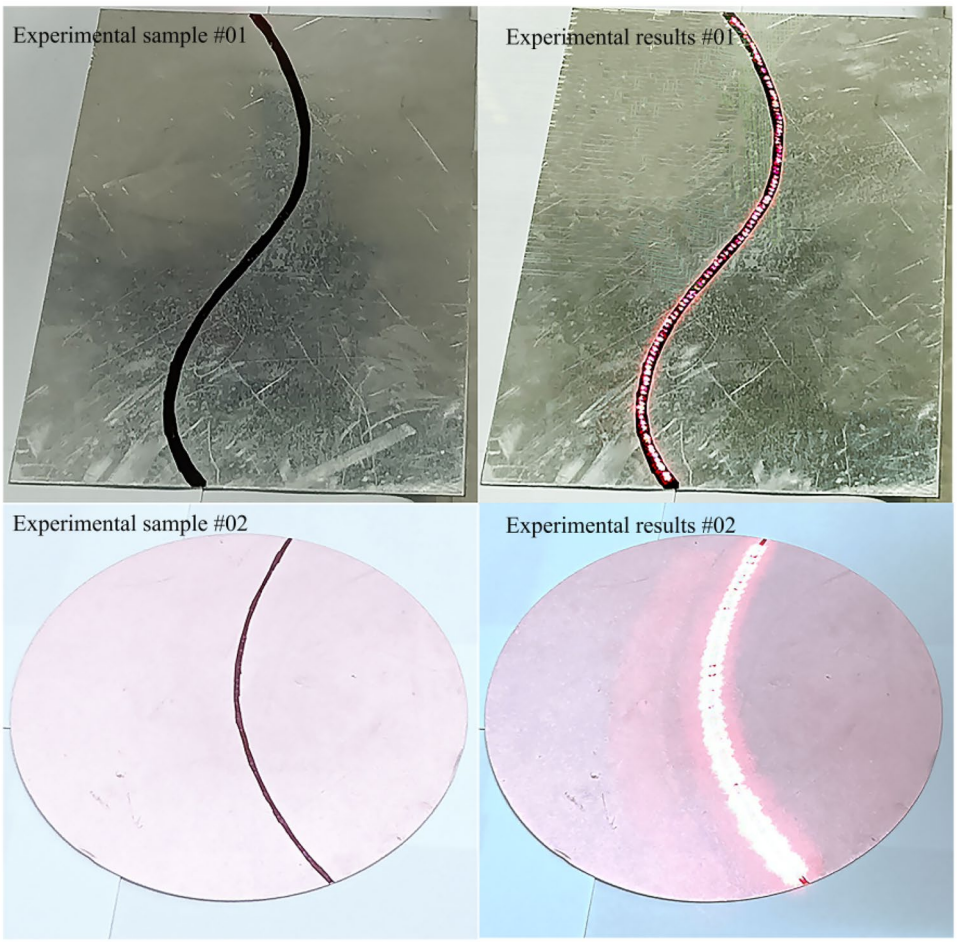

Fig. 14 Path following experimental results

Compared with the AOLP method [10], the proposed method is more precise and shows some new advantages. The error of AOLP originates partly from the sensor error and the pose estimation of the target 3D model. The robot cannot adjust the output action according to the scenario of the path while executing the specified path planning command. The advantage of the proposed method is that the robot can autonomously perform path following without the 3D model of the workpiece and programming the path beforehand. The robot can correct the error by adjusting the output action according to the vision information during the path following.

Compared with the Q-learning method [28], the proposed method has higher performance in handling complex scenarios and in path-following accuracy. Q learning uses tables to store the values between states and actions. In fact, it is difficult to map the relationship between high-dimensional image states and actions using tables. Therefore, the path image captured by the camera is simplified in [28]. The robot can follow the path while avoiding any contact between the path and the gripper, while its path-following accuracy is not high. In the proposed approach, a deep Q network is used to learn the value function between image state and path following action. The proposed method can handle complex images with different textures extending the range of applications.

In general, the proposed approach allows the robot to learn path-following skills directly from the image sensors. It provides a novel solution to cope with the complex working environment of industrial robots. The simulation technology is rapidly evolving. Simulation platforms can simulate more realistic industrial robot work environments and complex sensors. Then, the proposed approach can incorporate more advanced sensors [14, 15] in order to allow the robots to learn more complex manipulation skills.

5 Conclusions

In order to equip the robot with intelligent path-following skills, a reinforcement learning DDQN-based robot path-following algorithm is proposed. In this algorithm, a DDQN is used to extract complex path features and map the output action. The proposed reward function and action mode in this paper can meet the requirements of path following. In addition, the robotic agent can learn path-following skills by itself in the virtual simulation

environment. The path-following simulation tests of several workpieces including random B-spline curves are performed. They verify the correctness of the proposed algorithm. The path-following experiment of the hand-drawn curve is conducted on the real UR5 robot, which indicates that the path-following strategy learned in the simulation environment can be directly applied to the real environment. Finally, with the development of the simulation technology, the proposed method can also train robots to learn complex special machining operation skills.

Author contribution Guoliang Liu: investigation, conceptualization, code, software, experiment, data curation, and writing of the manuscript. Wenlei Sun: funding acquisition, conceptualization, project administration. Wenxian Xie: software, code, methodology, experiment. Yangyang Xu: ROS platform, system, code, experiment, software.

Funding This research is supported by the Key Laboratory Open Fund in Autonomous Region (2020520002) and the Key Research and Development Program in Autonomous Region (202003129).

Declarations

Ethics approval This manuscript has not been published elsewhere, in whole or in part. All authors have made significant contributions and agree with the content of the manuscript.

Conflict of interest The authors declare no competing interests.

References

1. Ghobakhloo M (2022) Industry4.0, digitization, and opportunities for sustainability. J Clean Prod 252:119869
2. Duan YQ, Edwards JS, Dwivedi YK (2019) Artificial intelligence for decision making in the era of Big Data – evolution, challenges and research agenda. Int J Inform Manage 48:63–71
3. Pérez L, Rodríguez Í, Rodríguez N, Usamentiaga R, García DF (2016) Robot guidance using machine vision techniques in industrial environments: a comparative review. Sensors 16(335)
4. Wang ZG, Wang HT, She Q, Shi XS, Zhang YM (2020) Robot4.0: continual learning and spatial-temporal intelligence through edge. J Comp Res Dev 57(9)1854–1863
5. Mnih V, Kavukcuoglu K, Silver D, Antonoglou I, Wierstra D, Riedmiller M (2013) Playing Atari with deep reinforcement learning. Comput Sci
6. Neto P, Mendes N, Ara´ujo R, Pires JN, Moreira AP (2012) High-level robot programming based on CAD: dealing with unpredictable environments. Ind Robot Int J Robot Res Appl 39(3):294–303
7. Polden J, Pan Z, Larkin N, Van Duin S, Norrish J (2011) Offline programming for a complex welding system using DELMIA automation. In: Chen SB, Fang G (eds) Robotic welding, intelligence and automation. Springer, Berlin, pp 341–349
8. Mnih V, Kavukcuoglu K, Silver D (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
9. Pahič R, Lončarević Z, Gams A, Ude A (2021) Robots kill learning in latent space of a deep autoencoder neural network. Robot Auton Syst 135:103690
10. Bedaka AK, Vidal J, Lin CY (2019) Automatic robot path integration using three-dimensional vision and offline programming. Int J Adv Manuf Technol 102:1935–1950
11. Deng D, Polden JW, Dong JF, Tao PY (2018) Sensor guided robot path generation for surface repair tasks on a large-scale buoyancy module. Ieee-Asme T Mech 23:636–645
12. Tian YX, Liu HF, Li L, Wang WB, Feng JC, Xi FF, Yuan GJ (2020) Robust identification of weld seam based on region of interest operation. Adv Manuf 8:473–485
13. Shah HNM, Sulaiman M, Shukor AK, Kamis Z (2018) Butt welding joints recognition and location identification by using local thresholding. Robot Cim-Int Manuf 51:181–188
14. Yang L, liu YH, Peng JZ, Liang ZZ, (2020) A novel system for offline 3D seam extraction and path planning based on point cloud segmentation for arc welding robot. Robot Cim-Int Manuf 64:101929
15. Zou YB, Wei XZ, Chen JX (2020) Conditional generative adversarial network-based training image inpainting for laser vision seam tracking. Opt Laser Eng 134:10614
16. Zhou CM, Huang BD, Fränti P (2022) A review of motion planning algorithms for intelligent robots. J Intell Manuf 33:387–424
17. Liu NJ, Tao L, Cai YH, Wang S (2019) A review of robot manipulation skills learning methods. Acta Automatica Sinica 45(3):458–470
18. Moravčík M, Schmid M, Burch N, Lisý V, Morrill D, Bard N, Davis T, Waugh K, Johanson M, Bowling M (2017) DeepStack: expert-level artifcial intelligence in heads-up no-limit poker. Science 356:508–513
19. Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, Lanctot M, Sifre L, Kumaran D, Graepel T, Lillicrap T, Simonyan K, Hassabis D (2017) Mastering chess and shogi by self-play with a general reinforcement learning algorithm
20. Jaderberg M, Czarnecki WM, Dunning I, Marris L, Lever G, Castañeda AG, Beattie C, Rabinowitz NC, Morcos AS, Ruderman A, Sonnerat N, Green T, Deason L, Leibo JZ, Silver D, Hassabis D, Kavukcuoglu K, Graepel T (2019) Human-level performance in 3D multiplayer games with population-based reinforcement learning. Science 364:859–865
21. Taitler A, Shimkin N (2017) Learning control for air hockey striking using deep reinforcement learning. Int Conf Cont Artif Intel Robot Optim IEEE
22. Zeng A, Song S, Welker S (2018) Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. IEEE/RSJ Int Conf Intel Robot Syst (IROS) 4238–4245
23. Hundt A, Killeen B, Greene N, Wu HT (2020) "Good robot!": efficient reinforcement learning for multi-step visual tasks with sim to real transfer. Ieee Robot Autom Let 5(4):6724–6731
24. Guo xw, Peng GZ, Meng YY (2021) A modified Q-learning algorithm for robot path planning in a digital twin assembly system. Int J Adv Manuf Technol
25. Li FM, Jiang Q, Zhang SS, Wei M, Song R (2019) Robot skill acquisition in assembly process using deep reinforcement learning. Neurocomputing 345:92–102
26. Wen SH, Zhao YF, Yuan X, Wang ZT, Zhang D, Manfredi LG (2020) Path planning for active SLAM based on deep reinforcement learning under unknown environments. Intel Serv Robot 13:262–272
27. Zhang T, Xiao M, Zou YB, Xiao JD (2020) Robotic constant-force grinding control with a press-and-release model and model-based reinforcement learning. Int J Adv Manuf Technol 106:589–602
28. Meyes R, Tercan H, Roggendorf S, Thiele T, Büscher C, Obdenbusch M, Brecher C, Jeschke S, Meisen T (2017) Motion planning for industrial robots using reinforcement learning. The 50th CIRP Conf Manufac Syst

29. Hasselt HV, Guez A, Silver D (2016) Deep reinforcement learning with double Q-learning. Proceed 13th AAAI Conf Artif Intel 2094–2100
30. Schaul T, Quan J, Antonoglou I (2016) Prioritized experience replay. Proceed 4th Int Conf Learn Represent 322–355
31. Shelhamer E, Long J, Darrell T (2017) Fully convolutional networks for semantic segmentation. Ieee T Pattern Anal 39(4):640–651
32. Le N, Rathour VS, Yamazaki K, Luu K, Savvides M (2021) Deep reinforcement learning in computer vision: a comprehensive survey. Artif Intell Rev
33. Quigley M, Gerkey B, Conley K, Faust J (2009) ROS: an open-source Robot Operating System. ICRA workshop on open source software