Robotic Obstacle Avoidance: A Virtual Modeling And Reinforcement Learning

Ming-Fei Chen
National Changhua University of Education
Han-Hsien Tsai
National Changhua University of Education
Wen-Tse Hsiao (wentse@itrc.narl.org.tw)
Natioal applied Research Laboratories Taiwan Instrument Research Institute https://orcid.org/0000-0003-4085-7351

Research Article

Keywords: six-axis robotic arm, reinforcement learning, collision detection, route planning

Posted Date: July 6th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-601053/v1

Abstract

This study developed a robotic arm self-learning system based on virtual modeling and reinforcement learning. Using the model of a robotic arm, information concerning obstacles in the environment, initial coordinates of the robotic arm, and the target position, this system automatically generated a set of rotational angles to enable a robotic arm to be positioned such that it can avoid all obstacles and reach a target.

The developed program was divided into three parts. The first part involves robotic arm simulation and collision detection; specifically, images of a six-axis robotic arm and obstacles were input to the Visualization ToolKit library to visualize the movements and surrounding environment of the robotic arm. Subsequently, an oriented bounding box algorithm was used to determine whether collisions had occurred. The second part concerned machine-learning–based route planning. The TensorFlow was used to establish a deep deterministic policy gradient model, and reinforcement learning was employed for the response to environmental variables. Different reward functions were designed for tests and discussions, and the program's practicality was verified through actual machine operations. Finally, the application of reinforcement learning in route planning for a robotic arm was proved feasible by the experiment. Such an application facilitated automatic route planning and achieved an error of less than 10 mm from the target position.

Introduction

Taiwan's small and medium-sized enterprises are mostly processed or foundry, which is characterized by large output and labor-intensive. However, to solve the problem of lack of work and improve product competitiveness, the introduction of robotic arms to complete production line automation is its development goal. However, the robot arm is limited by the original controller form, and it is less flexible to introduce special function requirements. To solve the problem of introducing robotic arm applications in factory automation, it is solved that the staff actually conducts the actual machine teaching and control on-site. To this end, an offline programming system is developed. By offline programming, a set of paths is planned in advance, and it can be directly used on the machine after the production line is completed to solve the problem of human teaching. The purpose of the online monitoring system is to allow users to directly control and monitor online. In addition to introducing the basic functions of the robot arm, it also improves a lot of manpower requirements. It also makes up for the collision problem caused by the error between the field model and the actual field during offline programming. Many works of literature present machine learning for industrial applications. A recent event in which Google DeepMind's AlphaGo defeated Korean professional Go player Lee Sedol has attracted public attention to machine learning. Machine learning involves the analysis of large amounts of data, identification of useful characteristics from the data, and the utilization of such characteristics. García-Ordás et al. [1] present a new approach to categorize the wear of cutting tools used in edge profile milling processes. It's based on machine learning and computer vision techniques, especially using the B-ORCHIZ model, a novel shape-based descriptor computed from the wear region image. Experimental results show that B-

ORCHIZ outperforms other shape descriptors achieving accuracy values between 80.24 and 88.46 % in the different scenarios evaluated. Gao et al. [2] proposed a new approach for the material removal model for robotic belt grinding of Inconel 718 based on acoustic sensing and machine learning. By using discrete wavelet decomposition (DWD) and fast Fourier transformation (FFT) to analysis the grinding process. The testing results indicate that the values forecasted by the model are consistent with the measured values. Kothuru et al. [3] used the support vector machine learning model as a decision-making algorithm to detect the cutting tool wear and failure during the end milling operation. The sound sensor signals collected during the machining process are analyzed through frequency domain to extract signal features that correlate the actual cutting phenomenon. The performance evaluation results of the proposed algorithm have shown accurate predictions in detecting tool wear under various cutting conditions with a high-speed response rate. Li and Chang [4] proposed an automated visual positioning system for precision placement of a workpiece on the fixture. The experimental evidence of workpiece placement confirms that the low-resolution (640× 480 pixels) camera can obtain a translational precision of ± 0.2 mm; by using the binocular system can control the rotational error within ± 0.1°. At the 20 × 20 mm2 spatial tolerance of the mobile platform, that can achieve a success rate of 100 % in 200 workpiece placement tasks. Chen et al. [5] used the Iterative learning approach is applied to infer the error compensation of the next workpiece machining by using the historical data. The simulation results and the real system input and output responses show that the proposed intelligent error compensation scheme and the iterative learning approach have significant superiority. Segreto and Teti [6] proposed a machining learning method based on artificial neural networks (ANNs) that were developed. By using multiple sensor monitoring to realize an intelligent system capable to determine the state of the polishing process in terms of target surface roughness achievement.

The experimental results indicated that the multiple sensor monitoring, advanced signal processing, and machine learning-based on ANNs were confirmed to be suitable for on-line surface roughness assessment of polishing processes.

The present study established a virtual system with learning ability and incorporated the concept of machine learning into the system to facilitate self-learning and identification of the optimal route by which obstacles can be avoided. To verify the feasibility of this system, we employed the robotic arm developed by our laboratory in the experiment. A set of movement trajectories generated by machine learning were first tested using software simulation until a route that avoided barriers was produced. This route was then followed by the robotic arm to verify its feasibility.

Reinforcement Learning

To plan routes for robotic arms, forward kinematics is generally first used to obtain the position of the end of a robotic arm, position of the target, and distance between them. Subsequently, stochastic actions, instead of inverse kinematics, are adopted to complete the route planning. To confine the system to producing movements relevant to reaching the target instead of generating random rotational movements without a goal, this study adopted the deep deterministic policy gradient (DDPG) [7], a

reinforcement learning algorithm that features the use of rewards and penalties to impose constraints, which resulted in clear and traceable steps. In reinforcement learning, an intelligent agent observes the current state, selects an action, and uses the reward received as a reference for deciding the next action. Accordingly, the most prominent feature of reinforcement learning is its ongoing trial-and-error process, which facilitates the exploration of an unknown environment and identification of the method that yields the greatest reward. Figure 1 shows the flowchart of reinforcement learning.

Reinforcement learning is divided into two types according to its action selection method; one selects actions on the basis of probability and the other does so on the basis of value. Probability-based reinforcement learning makes judgments according to current environmental factors, outputs the probability of each selection, and makes random selection. Value-based reinforcement learning executes the action with the highest value. The main strength of probability-based reinforcement learning is that it allows continuous action selection; its major weakness is its failure to update the reward for the current action. Therefore, Witten integrated the two types of reinforcement learning and proposed actor–critic learning [8], which the present study adopted in conjunction with deep learning to produce the DDPG.

2.1 Policy-Based Reinforcement Learning

Probability-based reinforcement learning achieves high performance in high-dimensional and continuous action spaces and boasts favorable convergence properties. Its most notable difference from value-based reinforcement learning is its ability to employ stochastic policy in certain cases of stochastic policy learning, enabling it to outperform deterministic policy learning in particular contexts. Nevertheless, stochastic policy learning exhibits problems such as low efficiency and the tendency to converge to local optimal solutions. A stochastic policy reinforcement learning method related to actor–critic learning is the policy gradient method, which has an artificial neural network (policy) that outputs the probability of each action for an input environment to influence the agent's selection. Because actions are selected according to probability and rewards are stochastically provided in certain given environments, the following steps are used in the policy gradient method (see Fig. 2): calculate the expected rewards, use gradient ascent to identify a direction that yields the highest reward, and update the probability and weight of each action progressively to achieve the optimal state.

2.1.1 Value-based reinforcement learning

The most frequently adopted value-based reinforcement learning method is Q-learning which, in contrast to the policy gradient method, requires the collection and sorage of data from various rounds, with the Q-function updated at each round according to each new batch of data collected. In Q-learning, a matrix is produced before each action to determine the expected reward for executing an action in a given environment, providing increasingly clear instructions regarding the next step with each step taken and revealing the relativeness between the current and the next steps. Accordingly, Q-learning is able to update at each step, which increases learning efficiency and prevents low learning efficiency caused by

an excessively large number of steps taken in a round. However, it is unable to process continuous actions when a large amount of information is involved. The detail flowchart of the Q-learning update as shown in Fig. 3.

To update matrix Q(s,a), a certain action (a_t) is first executed in a given environment (s_t) to obtain the preupdated matrix $Q(s_t,a_t)$. After the execution of a_t, the reward values (R) in the next environment ($s_{t+1)}$) are examined; the action that yields the highest reward (max $Q(s_{t+1},a_{t+1})$) in this environment is executed. This reward value is multiplied by the discount rate (γ) and then summed with R in s_{t+1}; the difference between this sum and $Q(s_t,a_t)$ is then obtained. Finally, this difference is multiplied by the learning rate (η) to update the old matrix ($Q(s_t,a_t)$).

2.1.2 Actor-Critic

The actor–critic framework is a concept proposed by Witten (1977). This framework integrates the strengths of value-based and policy-based reinforcement learning, thereby facilitating the processing of continuous and high-dimensional values and concurrently single-step updates. The actor incorporates the policy gradient selection feature of policy-based learning to promote stochastic action selection and interactions between an intelligent agent (π) and the environment (S). The critic involves the use of value-based reinforcement learning, with Q-learning being the most frequently adopted method, to judge the benefit of executing each action.

Because of the various possibilities involved in continuous actions, the policy gradient method typically involves the selection of a random value, referred to as the sampling method, for updates. However, when an extreme value is selected, the established learning process may be disrupted, resulting in failure to converge. Actor–critic learning is most different from policy gradient in that it does not directly multiply the stochastic gradient of a current condition with the sampled rewards of the current round during the updating of action probabilities. Instead, it calculates the expected total reward following an action selected in Q-learning, thereby avoiding the stochasticity in rewards sampled for the current round and effectively eliminating the possibility of the learning process being disrupted by extreme values. Figure 4 shows the flowchart of the actor–critic updates.

2.1.3 deep deterministic policy gradient (DDPG)

In actor–critic learning, the actor acts and performs updates on the basis of values defined by the critic; updates are continually being made because data change according to changes in the actor. The actor and critic functions usually fail to converge when they update concurrently. In response to this problem, the Google DeepMind team proposed an updated version of actor–critic learning, namely the DDPG, which integrates the concept of deep leaning to solve challenges in convergence.

The DDPG was proposed in 2016, and its underlying framework is based on three learning methods, actor–critic, deep Q-network (DQN) [9], and deterministic policy gradient (DPG) [10] learning. DPG learning is an extension of policy gradient learning, its key difference being that it incorporates the function μ into the probability output by its artificial neural network and outputs a final action, thus reducing unnecessary various calculations and accelerating convergence in high-dimensional or continuous actions. DQN learning is based on the concept of Q-learning, its main difference being that it calculates the Q value by using an artificial neural network instead of matrix storage. Specifically, it estimates the expected reward of the continuous actions of an intelligent agent (π) in the current state (*s*). The integrated artificial neural networks of the DQN and DPG are combined with experience replay and fixed Q-target mechanisms, which have been successfully applied with DQN, to remove the correlation between samples and increase the likelihood of convergence.

Reward Function

After a reinforcement learning framework has been established, defining rewards is the next focus [11]. A well-defined reward offers robotic arms good indicators with which to learn and improve, whereas a poorly defined reward may lead to undesirable convergence results, namely a cobra effect. Therefore, the present study discusses the definition of the reward function.

Default rewards exist in each environment in all types of reinforcement learning applications and are typically defined according to the goals attained and missions failed. In Atari 2600 Pong, a table tennis arcade game in which players score points by hitting a white ball out of the boundaries by moving their rackets up and down, the default environmental rewards of the game are defined according to the number of points gained or lost. In the field of robotic arm route planning, the present study defined the default rewards on the basis of whether it reaches its destination and whether collisions occur, with two types of collisions defined: a collision is considered to have occurred when the robotic arm reaches its maximum rotation angle or when a collision with an object in the external environment is detected. Negative feedback is given when a collision occurs, and positive feedback is given when the destination is reached.

Learning based solely on rewards provided by the environment is ineffective because such rewards are usually sparse rewards. In the example of AlphaGo, if no rewards are given during the playing of chess and a reward is only given in the end when the winner is decided, then the quality of each chess move cannot be determined in machine learning, thus increasing the time required for training. The best solution to the sparse reward problem is to establish another reward function and generate a set of rewards that are not provided by the environment and that can guide learning toward an ultimate goal. This study adopted the two most frequently used reward function definitions, namely reward shaping and curriculum learning.

3.1 Reward shaping

Reward shaping is a function constructed to solve the sparse reward problem and is aimed at leading a robotic arm to the target positions through the establishment of continuous rewards. The most intuitive application in a robotic arm is determining the distance between the end of the robotic arm and the target position. This study incorporated the concept of forward kinematics to obtain the position of the end of a robotic arm and compared this position with the target position, the result of which served as a reference for reward shaping; a large reward was given when the distance between said positions was short, thereby preventing the robotic arm from rotating randomly when approaching the target position. Because a robotic arm taking unnecessary routes is undesirable in route planning, the reward in the reward shaping process was a negative value; a small negative value was provided when the distance from the target position was small, and the robotic arm was expected to use this value to reach the defined goal in the shortest possible time.

3.2 Curriculum learning

The function of curriculum learning is different from that of reward shaping in that reward shaping focuses on solving the sparse reward problem, whereas curriculum learning aims to solve overly complex problems. It is difficult for a robotic arm to determine a set of axial rotational movements that will enable it to move between two points (A and B) without additional information and with minimal error (see Fig. 5). Therefore, this study defined layers surrounding the target position with several colors and provided rewards progressively as the robotic arm approached this position; specifically, a reward was given when the robotic arm reached the yellow layer, and the reward was greater when it reached the blue layer, thus enabling the robotic arm to approach the target gradually.

Experiments

This study divided the program into three major parts. The first part concerns robotic arm simulation and collision detection; this part was developed with the Visualization ToolKit to reveal the robotic arm's current posture and detect collisions by using an oriented bounding box algorithm. The second part is related to the establishment of reinforcement learning networks; this part was constructed using Tensorflow to establish a communication network between the actor and critic. The third part entails the establishment of environments and rewards required by forward kinematics and reinforcement learning; in this part, the forward kinematics mentioned in Section III are used to assess the environment and define rewards.

To verify the practicality of route planning, this study employed a six-axis robotic arm developed in our laboratory for simulation of the pick-and-place operations of a lathe to execute the routes in the actual field in addition to using software to conduct calculations. The route between the origin and the simulated lathe gripper was planned and verified using the actual machine as shown in Fig. 6.

Table 1 Training parameters for integration and verification in the actual field.

Parameters	Values
Target position	[x: 700, y: -370, z: 860]
Tolerable error of target	±10 mm
Number of training cycle	5000
Number of rotations in a single cycle	500
Memory capacity	80000

In the verification process, the routes for a robotic arm were generated using the reinforcement learning system established by the present study in place of manually taught routes. Subsequently, the Solidworks drawing software was used for software cross-verification. After a route for the robotic arm was confirmed to be collision-free and the training result was confirmed to be correct, the route was input to offline programming software previously developed by our laboratory and executed using the actual robotic arm for further integration and verification were summarized in Table 1.

To perform the pick-and-place operations with a lathe, the posture of the robotic arm needed to be changed through reinforcement learning. Therefore, this study conducted point-to-point training by using the reward function established on the basis of forward kinematics and incorporated the concept of curriculum learning as the target position was reached. The posture of the robotic arm when it reached the target position was assessed; the closer the posture was to the designated posture, the greater the reward was. Figure 7 shows the schematic diagram of the simulation of reinforcement learning training including (a) the posture reward was not incorporated and (b) the posture reward established using curriculum learning was incorporated model. After the training results of reinforcement learning could be seen in Fig. 8. Figure 8 (a) shows the actual field integrated with total rewards in training. Figure 8 (b) shows the actual field integrated with the number of rotations in each training cycle.

Finally, the angles of the robotic arm were input to Solidworks to verify the accuracy of estimations made using forward kinematics, investigate whether potential collisions existed, and confirm the accuracy of the graphical interface. After the conclusion of software confirmation, the angles were incorporated into the offline human–machine interface to verify the practicality of the planned route. Figure 9 show the actual machine operation. As the Fig. 9 shows the photo of the (a) side view when posture rewards were not incorporated and (b) side view with posture rewards incorporated, respectively.

Conclusions

This study employed reinforcement learning to develop a self-learning six-axis robotic arm simulation system, which can be used to move a robotic arm away from obstacles and plan routes. The following conclusions were drawn according to the experiment results:

(1) The DDPG can be applied in route planning for six-axis robotic arms. The high applicability of the DDPG is attributable to its employment of fixed Q-target and experience replay, which was successfully applied in a DQN, and the strength of the DPG, namely its deterministic properties, both of which solve the low-convergence problem and increase the training speed.

(2) On the basis of the definition of a reward function, the learning direction in machine learning can be determined; however, the cobra effect may occur if an inappropriate definition is used.

(3) Reward shaping effectively improves the training outcome and provides continuous rewards in reinforcement learning rather than making the robotic arm rotate randomly to reach the target position, thereby enabling each move to be tracked.

(4) Curriculum learning effectively reduces the difficulty of successfully performing target movements; specifically, it facilitates the process of deconstructing and learning movements, thereby greatly improving the learning success rate and ultimate learning results.

Declarations

a. Funding

- No funding was received.

b. Conflicts of interest/Competing interests (include appropriate disclosure

-This manuscript without Conflicts of interest and Competing interests

c. Availability of data and material (data transparency)

- The development technology that based on the TensorFlow for Robotic obstacle avoidance.

d. Code availability (software application or custom code)

- The development technology based on TensorFlow to establish a deep deterministic policy gradient model, and reinforcement learning was employed for the response to environmental variables.

e. Ethics approval (include appropriate approvals or waivers)

- Not applicable

f. Consent to participate (include appropriate statements)

- All of the authors have read and agree to publish the submit manuscript.

g. Consent for publication (include appropriate statements)

- All of the authors have read and agree to publish the submit manuscript.

h. Authors' Contributions

- M.F. Chen_ conceptualization and instruct of manuscript.

- W.T. Hsiao_ validation, review, writing original draft, and editing.

- H.H. Tsai_ experimental setup, source code writing, data correction, data analysis.

References

1. García-Ordás MT, Alegre E, González-Castro V, Alaiz-Rodríguez R (2017) A computer vision approach to analyze and classify tool wear level in milling processes using shape descriptors and machine learning techniques. Int J Adv Manuf Technol 90:1947–1961
2. Gao K, Chen H, Zhang X, Ren XK, Chen J, Chen X (2019) A novel material removal prediction method based on acoustic sensing and ensemble XGBoost learning algorithm for robotic belt grinding of Inconel 718. Int J Adv Manuf Technol 105:217–232
3. Kothuru A, Nooka SP, Liu R (2018) Application of audible sound signals for tool wear monitoring using machine learning techniques in end milling. Int J Adv Manuf Technol 95:3797–3808
4. Li CH, Chang YM (2019) Automated visual positioning and precision placement of a workpiece using deep learning. Int J Adv Manuf Technol 104:4527–4538
5. Chen T, Tian X, Li Y (2013) Intelligent dimensional error pre-compensation in CNC grinding using iterative learning approach. Int J Adv Manuf Technol 67:1825–1832
6. Segreto T, Teti R (2019) Machine learning for in-process end-point detection in robot-assisted polishing using multiple sensor monitoring. Int J Adv Manuf Technol 103:4173–4187
7. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D (2015) Continuous control with deep reinforcement learning. arXiv preprint arXiv 1509:02971
8. Witten IH (1977) An adaptive optimal controller for discrete-time Markov environments. Inform Control 34:286–295
9. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D (2013) Playing Atari with deep reinforcement learning. arXiv preprint arXiv 312:5602
10. Silver D, Lever G, Heess N, Degris T, Wierstra D, Riedmiller M (2014) Deterministic policy gradient algorithms. In 2014 Proc. ICML'14: I387-I395
11. Dewey D (2014) Reinforcement learning and the reward engineer principle. In 2014 AAAI Spring Symposium

Figures

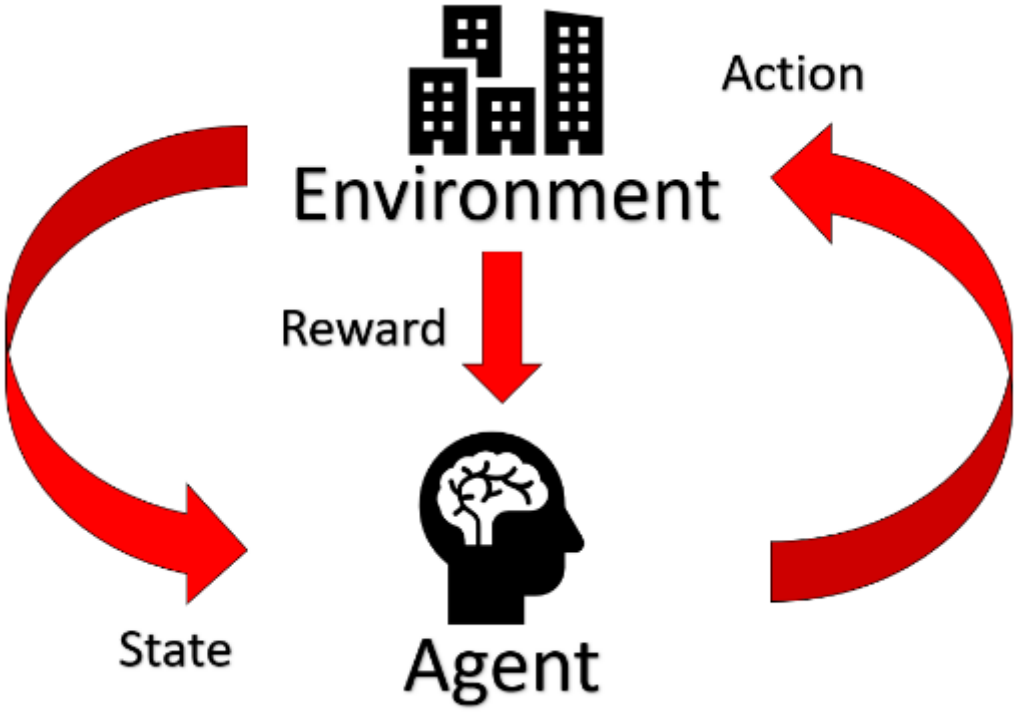

Figure 1

Flowchart of reinforcement learning.

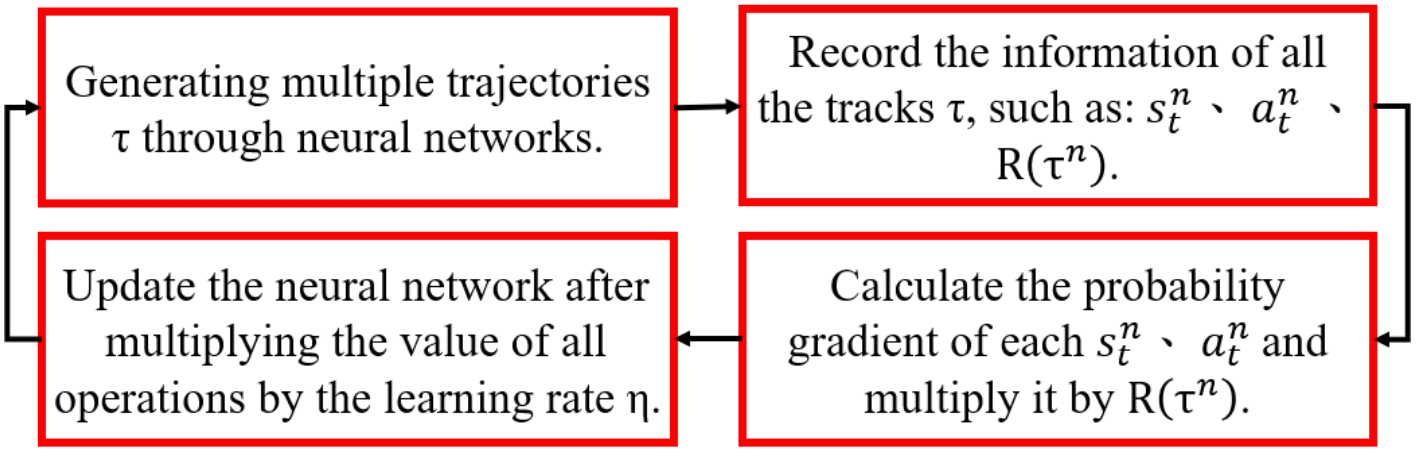

Figure 2

Flowchart of policy gradient updates.

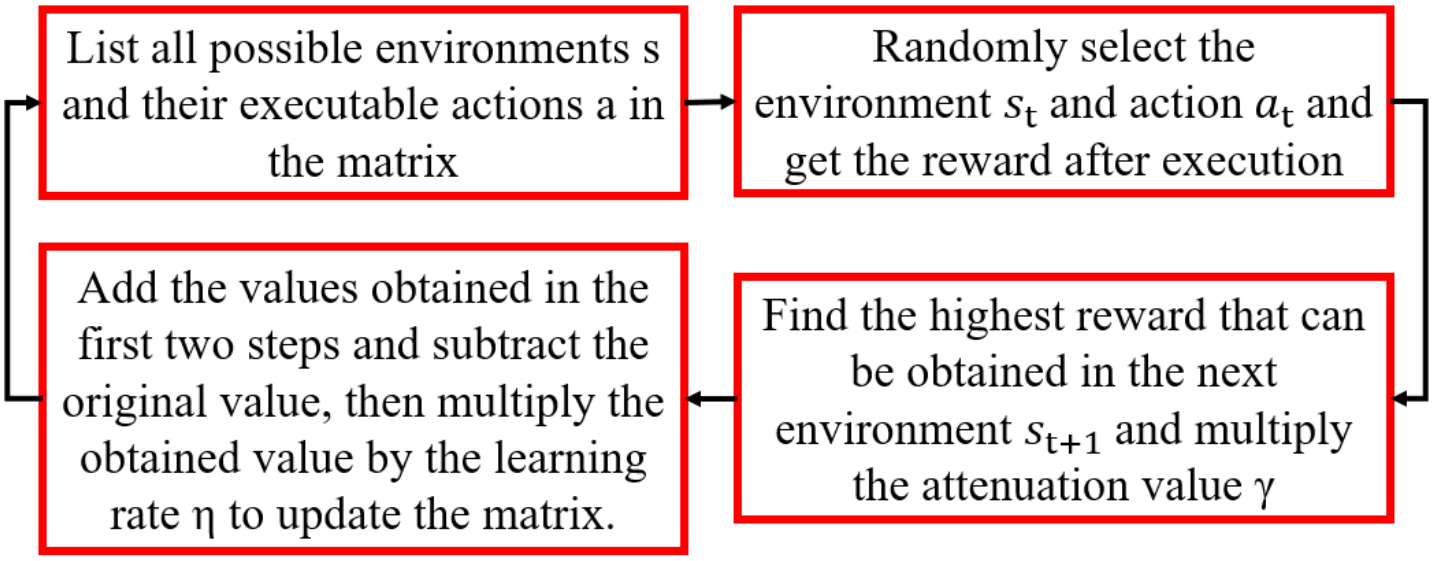

Figure 3

Flowchart for Q-learning update.

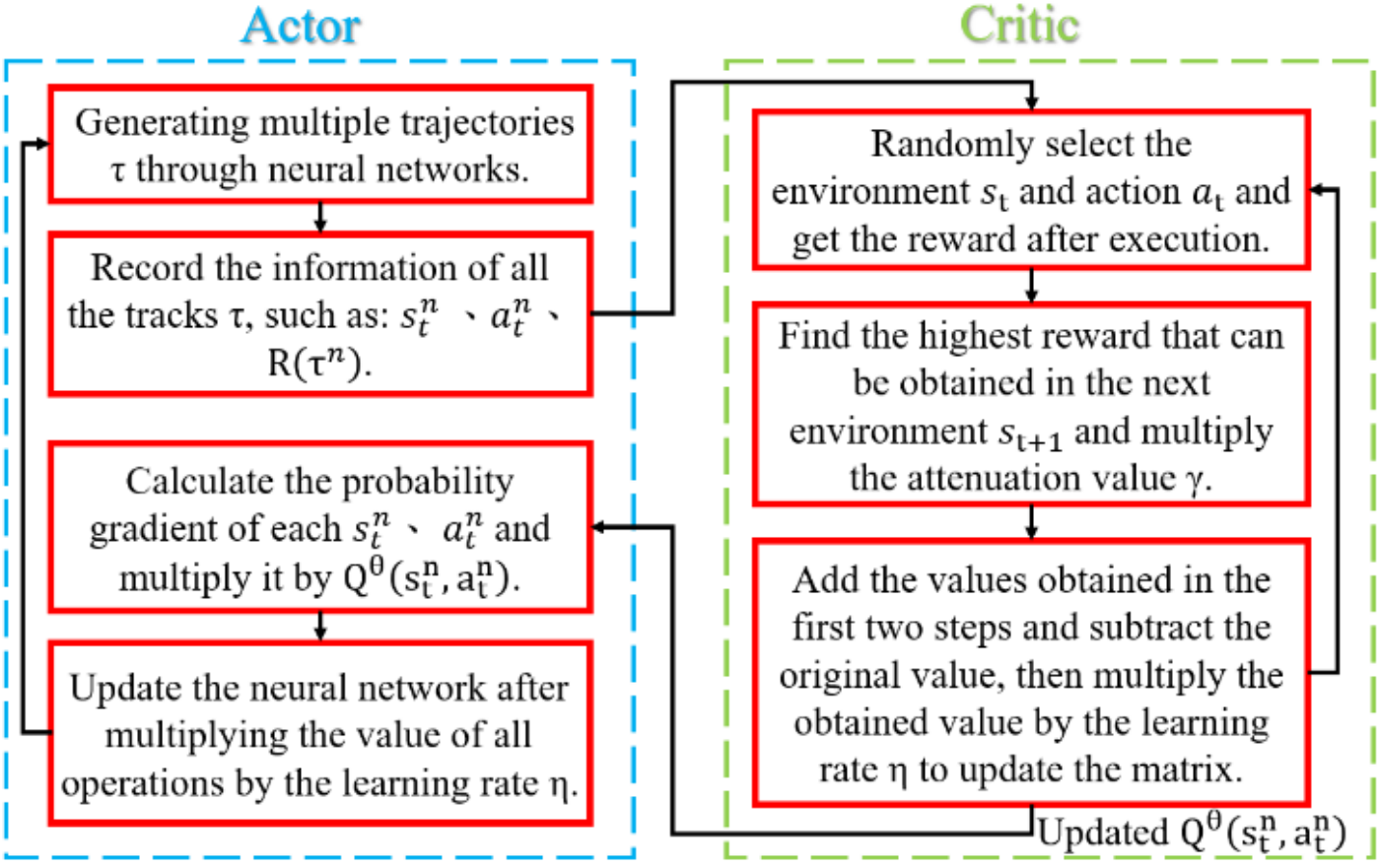

Figure 4

Flowchart of actor–critic updates.

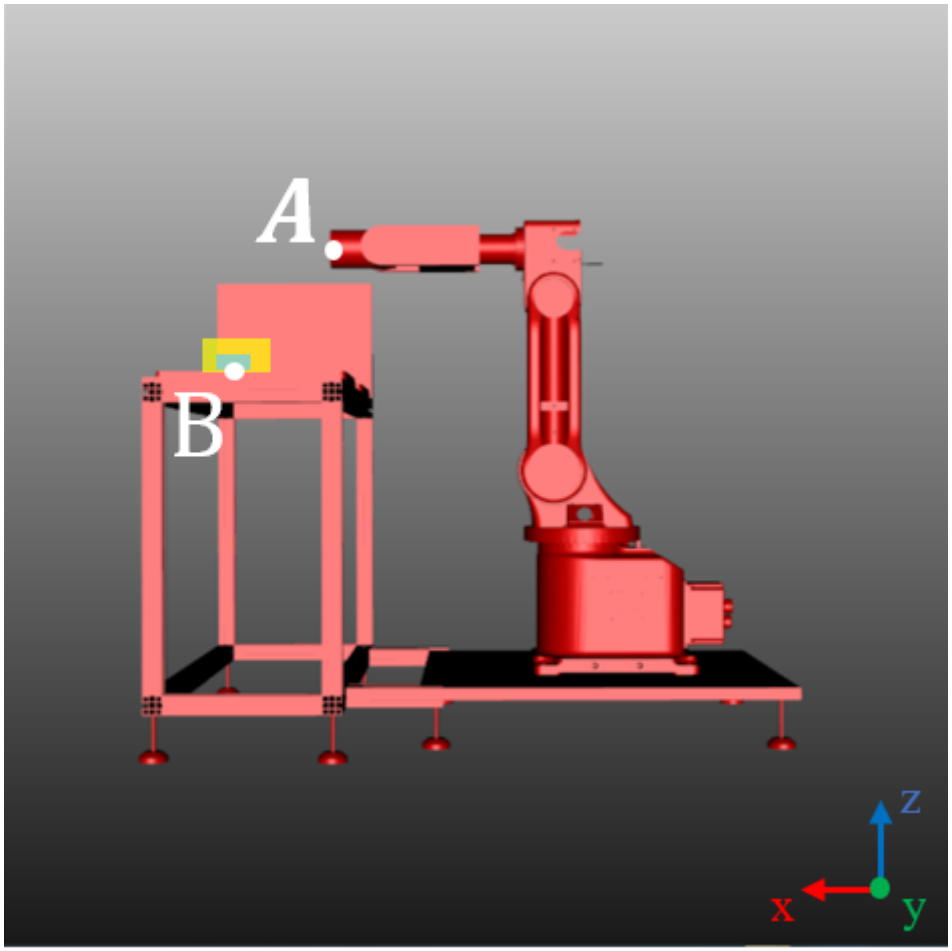

Figure 5

Conceptual graph of curriculum learning.

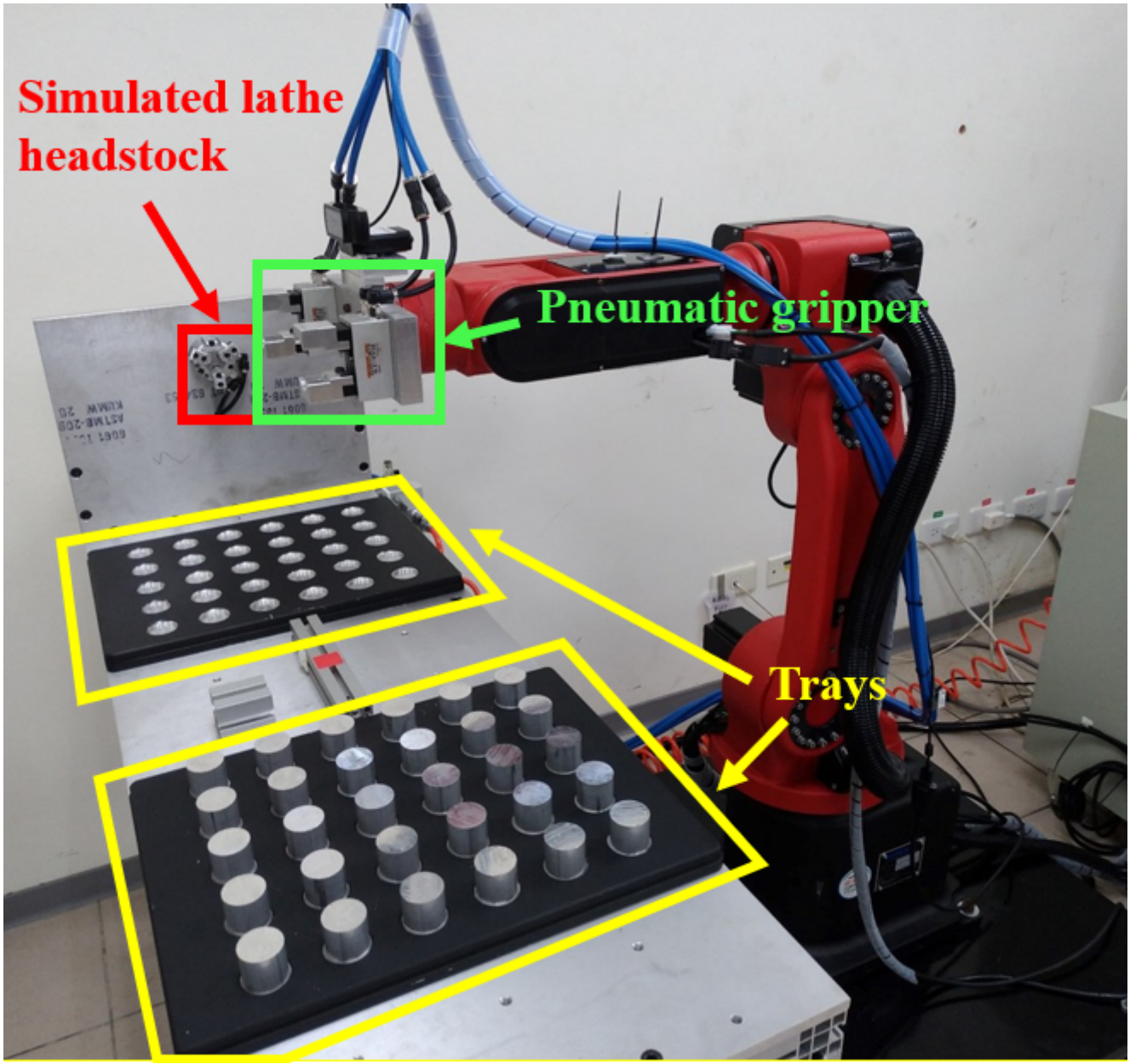

Figure 6

Actual field.

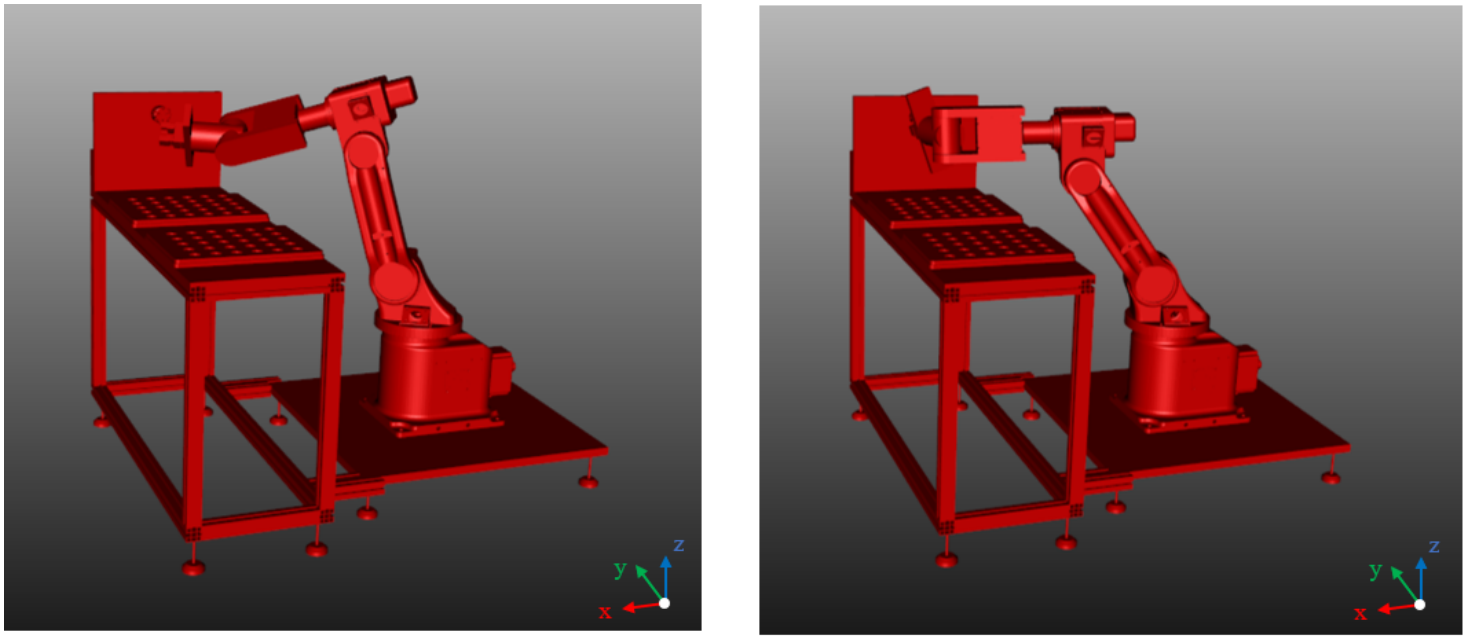

Figure 7

Simulation of reinforcement learning training. (a) Results when the posture reward was not incorporated. (b) Results when the posture reward established using curriculum learning was incorporated.

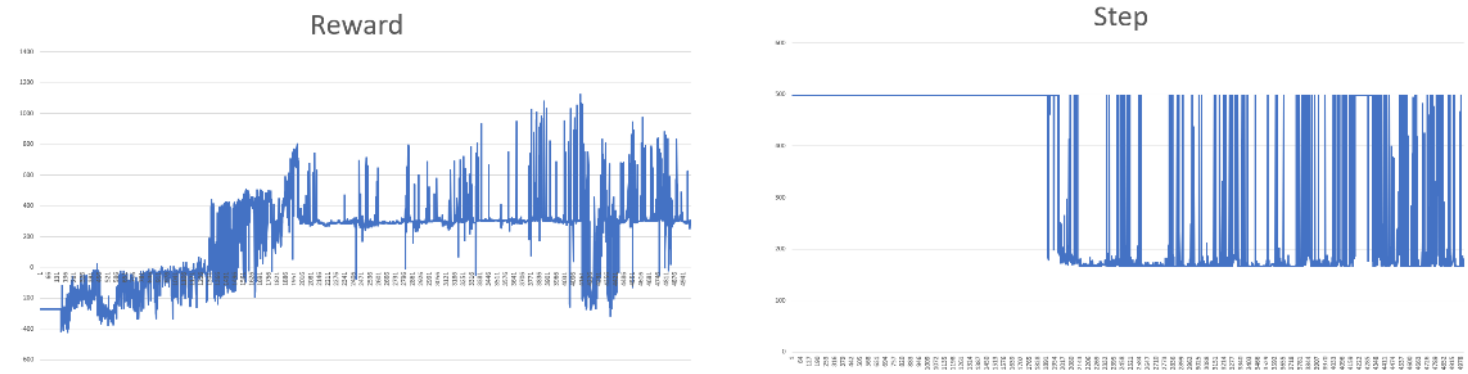

Figure 8

Training results of reinforcement learning. (a) actual field integrated with total rewards in training. (b) actual field integrated with the number of rotations in each training cycle.

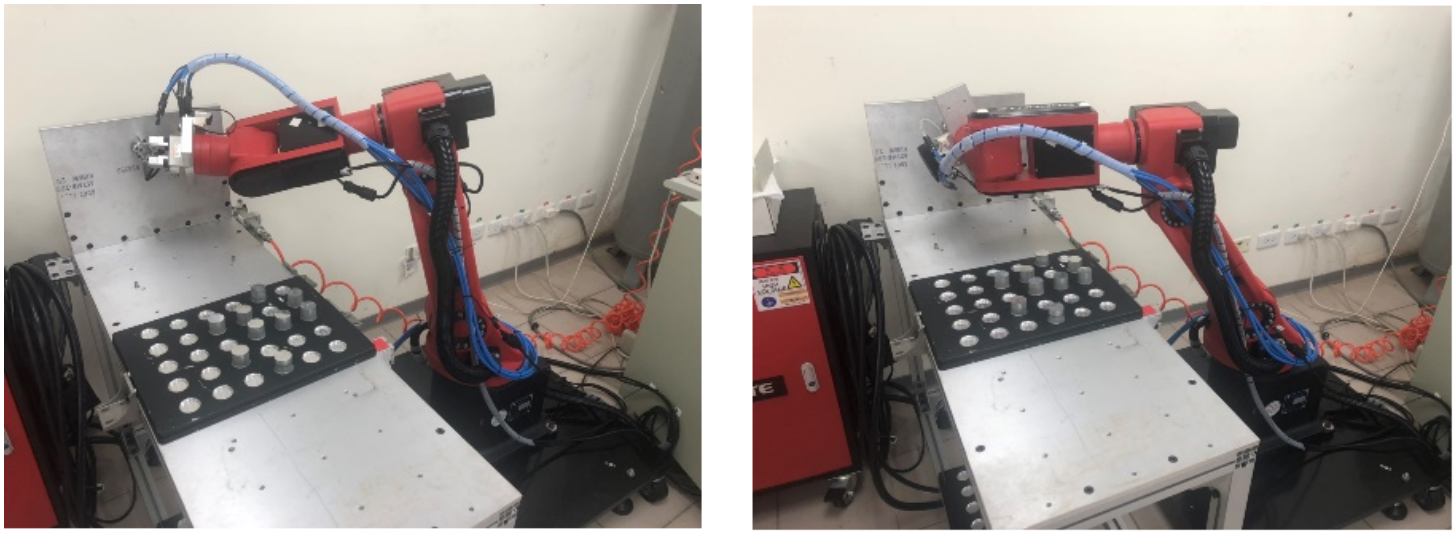

Figure 9

Actual machine operation. (a) side view when posture rewards were not incorporated.(b) side view with posture rewards incorporated.