Path Planning for Intelligent Robots Based on Deep Q-learning With Experience Replay and Heuristic Knowledge

Lan Jiang, Hongyun Huang, and Zuohua Ding, *Member, IEEE*

***Abstract*— Path planning and obstacle avoidance are two challenging problems in the study of intelligent robots. In this paper, we develop a new method to alleviate these problems based on deep Q-learning with experience replay and heuristic knowledge. In this method, a neural network has been used to resolve the "curse of dimensionality" issue of the Q-table in reinforcement learning. When a robot is walking in an unknown environment, it collects experience data which is used for training a neural network; such a process is called experience replay. Heuristic knowledge helps the robot avoid blind exploration and provides more effective data for training the neural network. The simulation results show that in comparison with the existing methods, our method can converge to an optimal action strategy with less time and can explore a path in an unknown environment with fewer steps and larger average reward.**

***Index Terms*—deep Q-learning (DQL), experience replay (ER), heuristic knowledge (HR), path planning.**

I. Introduction

With the development of science and technology, intelligent robots play an increasingly important role in human life. Avoiding obstacles in unknown environments and exploring a route are the most basic tasks of intelligent robots. Examples include sweeping robots, mining robots, and rescue robots. Due to the lack of detailed environment information and the unpredictable nature of the environment, it is difficult for intelligent robots to autonomously plan a path and avoid obstacles.

In the traditional method, researchers often regard the environment as a geometric world and construct a map [1], [2], but it is time-consuming to build and update maps and it is impossible to construct a map that includes all the scenarios. Fuzzy logic method can cope with uncertain data, and make the robot navigate whiling ensuring obstacle avoidance [3]. Heuristic algorithms are widely used in path planning. In [4], [5], particle swarm optimization algorithm is used to avoid obstacle collision. The ant colony algorithm is also used to do path planning in [6]. All of them adopt heuristic functions to coordinate the robot to explore in a good direction. The artificial potential field method regards the robot environment as a potential field, in which the target point produces gravitational force to attract to the robot, and obstacles generate repulsive force to repel the robot. The studies [7], [8] propose two modified artificial potential methods for path planning. Map construction and neural network are combined to sense the environment and avoid collisions in [9], the system constructs a grid-based map by using known informations and calculates the optimal trajectory by using a neural network.

In recent years, reinforcement learning has been widely used in intelligent robot path planning and obstacle avoidance [10], [11]. But there are several shortcomings in reinforcement learning. First, the "curse of dimensionality" occurs when the robot is put into a complex environment. In addition, slow convergence is still a problem in reinforcement learning. It takes a long time to train the robot [12]. The last issue is the poor portability and generalization of reinforcement learning, where a trained robot cannot move in a new unknown environment.

In this paper, we apply deep Q-learning (DQL) with experience replay (ER) [13], [14] and heuristic knowledge (HK) for robot path planning and obstacle avoidance. In this method, a neural network is used to replace the Q-table in reinforcement learning. We take the original sonar signal as the input of the neural network, which solves the problem of "curse of dimensionality". The experience replay mechanism maximizes the use of experience data that is collected by robots during moving and disrupts the correlation of the neural network's training data. Heuristic knowledge provides guidance for the actions selection of robots and helps the network converge faster. Simulation results shows that our method ensures the intelligent robot can path plan without collision in an unknown environment. They also show the effectiveness and general applicability of our method.

The structure of this paper is organized as follows. Section II introduces the framework of our approach; Section III presents a method to train the neural network with experience replay and heuristic knowledge; Section IV shows the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

This work was supported by the National Natural Science Foundation of China (61751210, 61572441). Recommended by Associate Editor Yebin Wang. *(Corresponding author: Lan Jiang and Hongyun Huang.)*

Citation: L. Jiang, H. Y. Huang, and Z. H. Ding, "Path planning for intelligent robots based on deep Q-learning with experience replay and heuristic knowledge," *IEEE/CAA J. Autom. Sinica*, DOI: 10.1109/JAS.2019.1911732.

L. Jiang and Z. H. Ding are with the Laboratory of Intelligent Computing and Software Engineering, Zhejiang Sci-Tech University, Hangzhou 310018, China (e-mail: jianglander@163.com; zouhuading@hotmail.com).

H. Y. Huang is with the Center of Multi-Media Big Data of Library, Zhejiang Sci-Tech University, Hangzhou 310018, China (e-mail: huanghongyun07@hotmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2019.1911732

simulation experiment and experimental result; Section V discusses some related works; and finally, Section VI gives some conclusions about this paper and introduces future works.

II. Our Approach

Our approach is based on modified reinforcement learning. First, we briefly describe commonly used reinforcement learning. Then, our approach is introduced.

A. Existing Reinforcement Learning

Reinforcement learning system is a system in which the agent learns action strategy from the mapping of the environment to behaviors to maximize the value of the reward. Rewards provided by the environment in reinforcement learning systems are evaluations of the quality of actions. Reinforcement learning systems gain knowledge in an action-evaluation environment and improves its action strategy to adapt to the environment.

Reinforcement learning is a learning technique that approximates dynamic programming. It determines the optimal strategy in a step-by-step manner and tries to find maximum cumulative reward value in every state as its optimization strategy [15]. Instead of requiring positive or negative labels, reinforcement learning enables a robot to autonomously discover an optimal behavior through trial-and-error interactions with the environment [16]. Fig. 1 is the framework of reinforcement learning, where the agent selects an action a according to the Q-table and executes it, then the environment returns a state s and a reward r to the agent. The most commonly used reinforcement learning algorithm is Q-learning. In Q-learning, the Q-table is an optimal strategy action value function $Q(s,a)$, it updates according to

$$Q(s,a) \leftarrow Q(s,a)+\alpha[r+\gamma \max_{a'} Q(s',a')-Q(s,a)] \quad (1)$$

where α is learning rate, γ is discount factor, r and s' are the instant reward and the next state after executing action a in state s, a' is a selected action in state s' under the current strategy, $\max_a Q(s,a)$ is the maximum cumulative reward value corresponding to state s'.

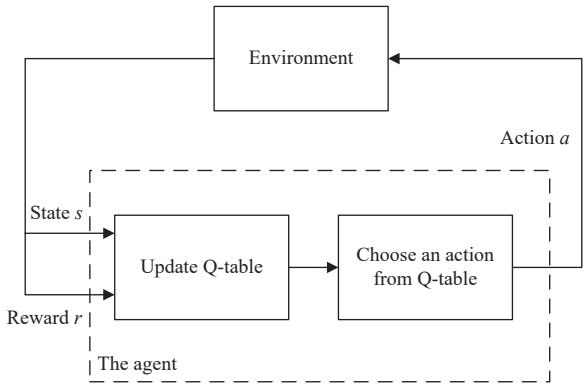

Fig. 1. The Framework of reinforcement learning system.

Reinforcement learning tasks are usually described using Markov decision processes (MDPs). The essence of MDPs is that the probability and the rewards obtained of the transition from the current state to the next state only depend on the current state and action, and has nothing to do with the past states and actions [17].

Today, reinforcement learning is widely used in all aspects. Xue *et al*. [18] introduced reinforcement learning to design an adaptive strategy for the iterated prisoner's dilemma and simulation results illustrate the effectiveness of this method. Liu *et al*. [19] studied how to apply reinforcement learning to complex system control. They propose parallel reinforcement learning to solve difficulties encountered in complex control system, such as data inefficiency, data dependency and distribution. In [20], an unsupervised weightless neural network learning algorithm and Q-learning are combined into a self-learning algorithm, which is implemented in a mobile robot navigation and obstacle avoidance.

B. Our Approach: A Modified Reinforcement Learning Algorithm

The intelligence robot system in Fig. 2 is a modified reinforcement learning system. In our approach, we use a neural network to replace the Q-table and add heuristic knowledge. In this study, the input of the network is the state of the robot, and its output is the expected cumulative reward corresponding to each action. Instead of choosing actions by querying the Q-table, the robot selects actions directly according to the output value of the neural network or heuristic knowledge.

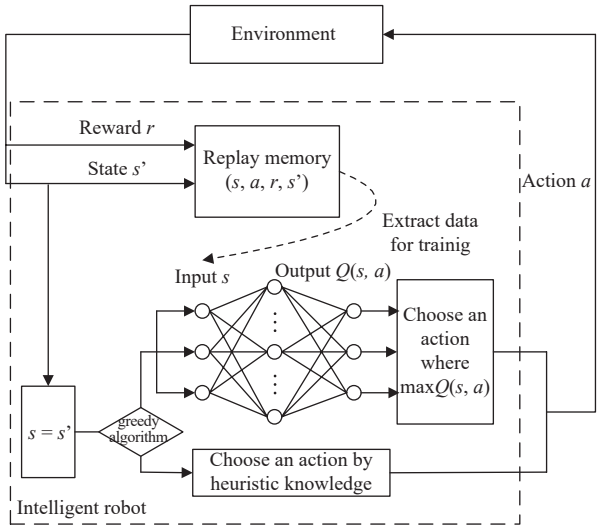

Fig. 2. The Framework of intelligence robot system.

Training a neural network requires a lot of data, but when the robot explores in a unknown environment, it is impossible to prepare enough training sample sets for it in advance. So, the robot collects experience data that are generated during its moving in a form as (s,a,r,s') and stores them in replay memory D. In this way, the quantity of training samples is guaranteed. Then, the robot samples random mini-batch experience data from D to train the neural network.

In some studies, neural networks have been used to replace the Q-table. In [21], Li et al. set up a neural network to learn a Q-function corresponding to traffic state and traffic system

performance. The study [22] uses the ε-greedy algorithm based on Q-learning and neural networks to make the robot arrive at the end line of the driving arena without any collisions. The neural Q-learning algorithm has been proven to be efficient in path planning on square grids in [23]. Determining how to make neural network learn more quickly with improved results is still a problem.

In order to provide effective training data to neural networks, we must add heuristic knowledge in this system. On the one hand, it can guide the behavior of the robot; on the other hand, it also increases the effectiveness of training the neural networks. With the help of heuristic knowledge, neural networks will converge to an optimal action strategy faster.

In this intelligence robot system, the robot implements path planning and obstacle avoidance tasks without a lack of prior training data and "curse of dimensionality". And, after training, we will obtain an adaptive obstacles avoidance model.

III. Training Deep Q-learning With Experience Replay and Heuristic Knowledge

A. Deep Q-learning With Experience Replay

Reinforcement learning and a neural network are combined to improve the generalization ability of the model and solve the "curse of dimensionality" in [23]. But the data samples for training neural networks are hard to obtain, and in above literature, data in Q-table are used to train neural networks. The data samples are always required to be independent in deep learning; however, the data samples in the Q-table are a sequence of highly correlated states produced in sequence. The actions selected by the robot have an impact on the environment in reinforcement learning. To alleviate those problems, DeepMind proposed a deep Q-learning with experience replay algorithm [14], which is proposed to play Atari. In this algorithm, experience data (s, a, r, s') is stored in the replay memory D. The size of the replay memory D is fixed at N, and the replay memory always stores the last N collected experience data. During the process of training, we sample mini-batch experience data from D randomly and use it to train the network according (1).

B. Heuristic Knowledge

The characteristics and quantity of training data are the most important factors determining the performance of a neural network model. Neural network is more likely to learn better representations by feeding it with sufficient data [24]. In order to learn an expected policy, it is very important to have sufficient and effective experience data in the replay memory for a robot.

There is a lot of randomness in traditional deep Q-learning. For example, the robot may hit obstacles when exploring randomly; the robot selects an action randomly with the ε-greedy algorithm. ε-greedy algorithm makes trade-off between exploration and exploitation base on a probability. Every time the robot selects an action, it randomly selects an action to explore with a probability of ε, $\varepsilon \in [0,1]$ or it exploits with a probability of $1-\varepsilon$, i.e., it selects the action with the highest reward value. For the neural network, the collision experience data and random action-selected experience data cannot contribute to neural network*s* training.

Heuristic knowledge is used in guiding the behavior of the robot, and it can reduce the randomness in an intelligent robot system. With the help of heuristic knowledge, the robot selects a suitable action, which provides characteristic training data for the neural network and accelerates the training process.

1) Obstacle Avoidance Knowledge: Because of the randomness of action choice in the early stage of reinforcement learning, the probability of the robot hitting obstacles is very high. If there is a large amount of collision experience data in replay memory, it will inevitably have a negative effect on the learning of the neural network, but if there is no collision experience as negative samples, the neural network can only learn one-sided knowledge. So, we equip the robot with obstacle avoidance knowledge, which helps the robot avoid obstacles as much as possible. In addition, the robot does not stop exploring when it hits obstacles and this collision experience data is also stored in replay memory.

In our work, we divide the state of the intelligent robot into four categories: 1) safe state (S), in which no matter what action be selected, the robot will not hit obstacles; 2) unsafe state (U), means the robot may hit an obstacle at next step; 3) failure state (F), in which the robot hits an obstacle and 4) winning state (W), where the robot arrives at the terminal.

If the robot is in a safe state, it selects an action randomly using the ε-greedy strategy; if the robot is in an unsafe state, the obstacle avoidance mode is enabled, and robot will select the action which makes it move away from obstacles as far as possible without thinking about the path planning. In this paper, if the robot is in an unsafe state, it will move in the direction of a sonar, the value of which is greater than the obstacle avoidance distance where it is farthest from the sonar and has the minimum sonar value.

Using obstacle avoidance knowledge can reduce the number of times the robot hits obstacles and makes a contribution to improve the quality of training data.

2) Goal-directed Knowledge: When the robot is at a safe state, it usually uses the ε-greedy strategy to select actions. The robot selects the action through the neural network with the probability $1-\varepsilon$ and selects an action randomly with probability ε, which increases the randomness of the action selection and also makes the data samples used for training become noisy. In order to reduce the blind exploration of the robot, goal-directed knowledge is used to guide the robot's action selection. With probability ε, the robot no longer selects actions randomly any more, but selects the action that takes it closer to the end point according to the goal-directed knowledge.

In this paper, we use the angle between the robot's direction and the end point as the basis for the goal-directed knowledge. The angle is defined as the rotating angle at which the robot rotates counterclockwise until it point to the end point. The range of the angle is from 0 to 360 degrees. From the size of the angle, we can know the positional relationship between the robot and the end point, e.g., if the angle is 180 degrees, it means that the robot direction is opposite to the end point.

Those provide guidance for the robot's action selection, so that the robot can move toward the end point. For example, when the angle is 30 (or 330) degrees, if the robot rotates 30 degrees to the left (or right), it will be in the direction of the end point.

Goal-directed knowledge provides good assistance for the selection of robot actions, and it is also helpful for speeding up the training process of the neural networks.

C. Training the Neural Network

Different from traditional Markov evaluation, we use a neural network to replace the Q-table in deep Q-learning with experience replay and heuristic knowledge. Without prior experience data training sets, the neural network should be trained during movement of the robot. At each step of the training, the value of the neural network is changing. In the neural network training process of this paper, there is a lack of target values. If we train a neural network with a series of continuously changing values as the target value, the neural network have difficulty converging. The network may not work because it falls into a feedback loop between the target value and the estimated value. Therefore, we adopt two neural networks to complete error back propagation and update the weights. We use a slower-updating network to provide target values and gradually optimize the weights of the neural network.

Those two neural networks work as shown in Fig. 3. One of them is called *evaluate_net*, which is used to generate an estimates value, denoted by *q_evaluate*. Another is called *target_net*, which generates a target Q value, denoted by *q_target*. The two neural networks have exactly the same structure. The *evaluate_net* always has the latest weight, it is constantly updated. The *target_net* is a historical version of *evaluate_net*, it records the old weights of the *evaluate_net* and updates periodically. We initialize the two neural networks with the same random weights at the beginning of training. During the training, we regard the difference between the two neural networks' output values as an error and propagate it back to the *evaluate_net*. By modifying the weight of each neuron, the error is minimized.

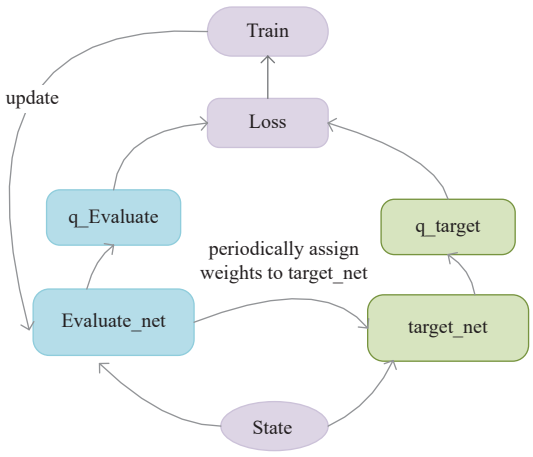

Fig. 3. Two neural network diagrams.

D. Using Deep Q-learning With Experience Replay and Heuristic Knowledge on Robots

We use deep Q-learning with experience replay and heuristic knowledge on robots in performing path planning and obstacles avoidance; the algorithm is presented in Algorithm 1.

Algorithm 1 Routing algorithm

Initialize replay memory D to capacity N
Initialize learning frequency K, target_net weight updates frequency L
Initialize *target_net* and *evaluate_net* with the same random weight
for episode = 1, M **do**
Initialize environment and set the robot to the start point
for $t = 1$, T **do**
Determine the state of the robot s_t
if s_t is at an unsafe state **then**
Select an action a_t through obstacle avoidance knowledge
end if
if s_t is at a safe sate **then**
With probability ε select an action a_t through goal-directed knowledge
Otherwise select an action a_t according to the output of *evaluate_net*
end if
Execute action a_t and observe immediately reward r_t and new state s_{t+1}
Store the experience data (s_t, a_t, r_t, s_{t+1}) in D
while $t \% K == 0$ **do**
Sample random mini-batch of experience data (S_j, A_j, R_j, S_{j+1}) from D
Set the random samples into the two networks, obtain q_tar_j, q_eva_j
$y_j = R_j + \gamma \max_{A_j}(q\_tar_j)$
$loss = (y_j - q\_eva_j)^2$
Update the weights of *evaluate_net* through a gradient descent procedure on *loss*
end while
if $t \% L == 0$ **then**
Assign the weights of the *evaluate_net* to *target_net* at interval
end if
Determine the state of the robot s_{t+1}
if s_{t+1} is at a winning state **then**
Finish this episode
else if s_{t+1} is at a failure state **then**
Step back and continue to learn
else
Continue this episode
end if
end for
end for

In the early exploration-exploitation process, the robot just collects and stores experience data in the replay memory until

there is enough data for the robot to learn. Instead of the weight being adjusted when the robot accomplishes an action, in our algorithm, every K steps the robot moves, it samples mini-batch experience data randomly from replay memory to train the neural network.

The neural network used in this algorithm is a three-layer backpropagation neural network. It has i input nodes, h hidden nodes and j output nodes. The size of mini-batch is m. The complexity of Algorithm 1 consists of two parts: select actions and train the neural network.

1) Select actions. The time complexity of using heuristic knowledge to select actions is $O(1)$. Using neural networks to select actions is a feedforward propagating process, the time complexity is $O(m \times h \times (i+j))$. Because the neural network is used to select action in most of time, the complexity of selection action is $O(m \times h \times (i+j))$.

2) Train the neural network. Before training the neural network, training data needs to be extracted from the experience repay memory. Because the input matrix dimensions of samples are fixed in dimension, the time complexity is $O(1)$. In a three-layer backpropagation neural network, the total time complexity for one training is $O(m \times h \times (i+j))$. Under the limitation of the learning frequency K, the training times of each epochs is T/K. So, the complexity of training the neural network per epochs is $O((T/K) \times (1+m \times h \times (i+j)))$, that is $O((T/K) \times (m \times h \times (i+j)))$.

The number of training epochs is M, and according to the above analysis, we can determine that the time complexity of Algorithm 1 is $O(M \times (T/K) \times m \times h \times (j+i))$.

This algorithm can also be used to solve other problems, such as, playing flappy bird, walking through a maze etc. It can create a very good model for certain tasks, but its final model can not apply to other tasks. This is because this model only learns one specific goal at a time and works for a specific task. When the learning task is completely different, it is necessary to retrain the model. But if the learning task is very similar, the obtained model has a certain degree of generalization.

IV. Path Planning for Robots

A. Simulation Environment

It is difficult to apply reinforcement learning to robots directly. Intelligence robots need thousands of repeated trainings to get a good behavior strategy. In this paper we use the simulation environment to train the robot.

In our experiment, the task of the robot is to move from the start point to the end point without collision in an unknown environment; the start point and the end point have been told. The distribution of the robot's sensors is shown in Fig. 4. There are 5 sonar sensors located in the robot, the angle difference between different sonar is 30 degrees. Those sonars' measured distances are denoted by $(s_1, s_2, s_3, s_4, s_5)$ respectively. The motion directions of robots are also divided into five kinds:

action 1: turns left 60 degrees and moves forward 30 cm;
action 2: turns left 30 degrees and moves forward 30 cm;
action 3: moves froward 30 cm;
action 4: turns right 30 degrees and moves forward 30 cm;

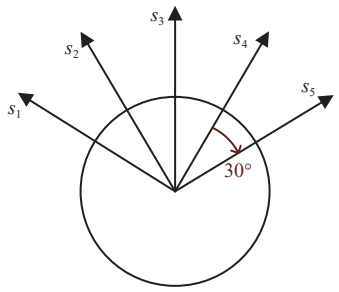

Fig. 4. The distribution of the robot's sonar.

action 5: turns right 60 degrees and moves forward 30 cm.

The angle between the robot current coordinate and the end point coordinate is denoted as β. According to the five actions of the robot, we design the goal-directed knowledge as follows:

1) If $0 \le \beta < 15$ or $345 < \beta \le 360$, *action* 3 will be selected;
2) If $15 \le \beta < 45$, *action* 2 will be selected;
3) If $45 \le \beta < 180$, *action* 1 will be selected;
4) If $180 \le \beta < 315$, *action* 5 will be selected;
5) If $315 \le \beta \le 345$, *action* 4 will be selected.

We combine 5 sensors distances and the angle β as the robot state s, where $s = (s_1, s_2, s_3, s_4, s_5, \beta)$.

Fig. 5 shows the structure of the neural network we used in this paper. It is a three-layer neural network. Its input is a robot state s, and its output are cumulative reward values corresponding to different actions under state s. There are 6 neurons in the input layer, 10 neurons in the hidden layer and 5 neurons in the output layers. The activation function for hidden layer is rectified linear unit (ReLU). The activation function of output layer is a linear function. We use stochastic gradient descent to train the neural network in our study.

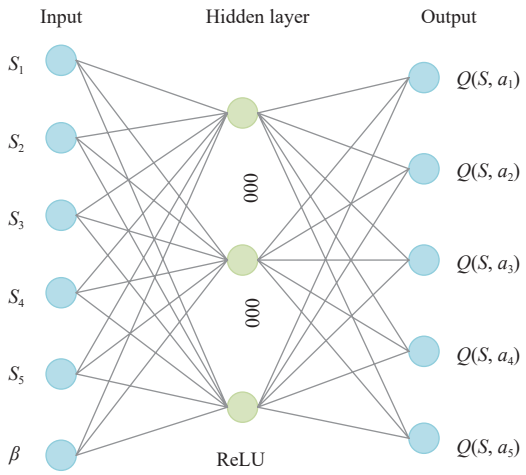

Fig. 5. Structure of the neural network.

Reward function is used for judging the merits of the action. According to the reward function, the robot interacts with the environment and adjusts its action strategy by reward value. Reward function helps to strengthen expected behaviors and punish unsuitable behaviors. As the only feedback to motivate the network convergence, the negative reward for the collision between the robot and obstacles must be very large [25]. The

reward function adopted in this paper is shown in (2), "$\rightarrow$" represents the transfer of states, AG (AO) means the robot stays away from the end point (obstacles), CG (CO) means the robot is close to the end point (obstacles).

$$reward = \begin{cases} -50, & F \\ -2, & S \rightarrow U; U \rightarrow U, CO \\ -1, & U \rightarrow U, AO \\ 0, & S \rightarrow S, AG \\ 1, & S \rightarrow S, CG \\ 2, & U \rightarrow S \\ 100, & W \end{cases} \tag{2}$$

B. Simulation and Analysis

We conduct comparative tests in the same experimental environment, by using the method in [23], DQL with ER and DQL with ER and HR. The comparison is divided into two parts, the first part is training phase and the second part compares generalization of the obtained model.

1) Training Phase: In [23], initial training phase is training a Q-table, we quantify the sonar values and the angle into 4 and 8 degrees respectively. Our approach will train a neural network and obtain a trained model.

Table I shows the detail parameters used in DQL training.

TABLE I
TRAINING PARAMETERS AND THEIR VALUATE

Parameter	Value
learning rate	0.01
discount factor	0.9
replay memory	2000
mini-batch size	32
greedy factor ε	0.1

There are three different map environments with two, three and four obstacles, which are recorded as $M1$–$M3$ respectively. The above mentioned three methods were used in those map environments. The training finishes when the average reward of smart cars tends to be stable. Fig. 6 shows 9 trajectories that the robot obtained by using three methods in three different maps, the map of each row is the same and the method of each column is the same. In each map, the blue dot in the lower right corner is the start point, the red "×" in the upper left corner is the end point, and the black areas are four walls and obstacles.

As we can see, those three methods can guide the robot at the end point without collision. The trajectories of the three methods are almost the same. In general, trajectories obtained using DQL are straighter than trajectories obtained by Q-learning. Once the robot has learned heuristic knowledge, it can travel in a more straight path than other robots. However, the robot will always try to choose the action that allows it to be further away from obstacles to avoid collisions, which makes trajectories not as smooth.

The moving step count and the average training epochs of the robot when it reaches the end point under different maps are listed in Table II. The difference between the moving step count is small; in addition, the two DQL methods result in the robot reaching the end point with the same moving step count. This shows that heuristic knowledge gives little effect on the moving step count. But in terms of training epochs, DQL performs much better than Q-learning does. Comparing to DQL+ER, our method's training epochs have been reduced by 33.33%, 15.84%, and 23.38% on each map. Notice that, after applying heuristic knowledge, the training rounds are greatly reduced again.

TABLE II
MOVING STEP COUNT AND AVERAGE TRAINING ROUNDS

	Model	$M1$	$M2$	$M3$
Moving step count	Q-learning	66	51	60
	DQL + ER	**60**	**50**	**57**
	DQL + ER + HK	**60**	**50**	**57**
Training epochs	Q-learning	102	185	258
	DQL + ER	84	101	154
	DQL + ER + HK	**56**	**85**	**118**

We choose the average reward the robot collects in an epoch as our evaluation metric. Fig. 7 shows the average reward when those three algorithms work on $M3$. As we can see, in the early stage of training, the average reward is very noisy. One reason is that the robot explores in the map, which makes it take a lot of steps to reach the end point and decreases the average reward. Without heuristic knowledge, Q-learning and DQL + ER may hit obstacles in the first few training rounds, which is also a reason why the average reward is low. In general, DQL + ER + HK converges earliest in 118 epoch and has the highest average reward, around 2.27. Although, the final average reward value of the three algorithms differs slightly, DQL + ER + HK has the highest average reward and obtains a better action strategy, allowing the robot to arrive at the end point more directly.

In summary, DQL with ER and HK makes the robot converge to the best trajectory faster than traditional Q-learning, and takes fewer steps to reach the target. With the help of heuristic knowledge, the robot can accelerate learning and obtain a better strategy.

2) Generalization and Flexibility: According to the training process in [23], we use the 323 data in the Q-table obtained from $M3$ to train a neural network (NN), and combines this Q-table with the network as an adaptive model (Q + NN). DQL with ER, and DQL with ER and HK obtain adaptive models from $M3$ too. Two tests are performed to test the generalization and flexibility of these three models. We record trajectories of the robot in the new environment for the first time, and compare the trajectories obtained by the three models.

Test 1: The Robot Is Initialized From Arbitrary Points on M3. Fig. 8 shows trajectories when the robot starts from three arbitrary starting points. The horizontal axis is three models, and the vertical axis is three arbitrary start point maps. In the top row of this figure, the start point is on the left side of the original start point. In this case, the differences among three trajectories obtained by the three models are not huge. In the

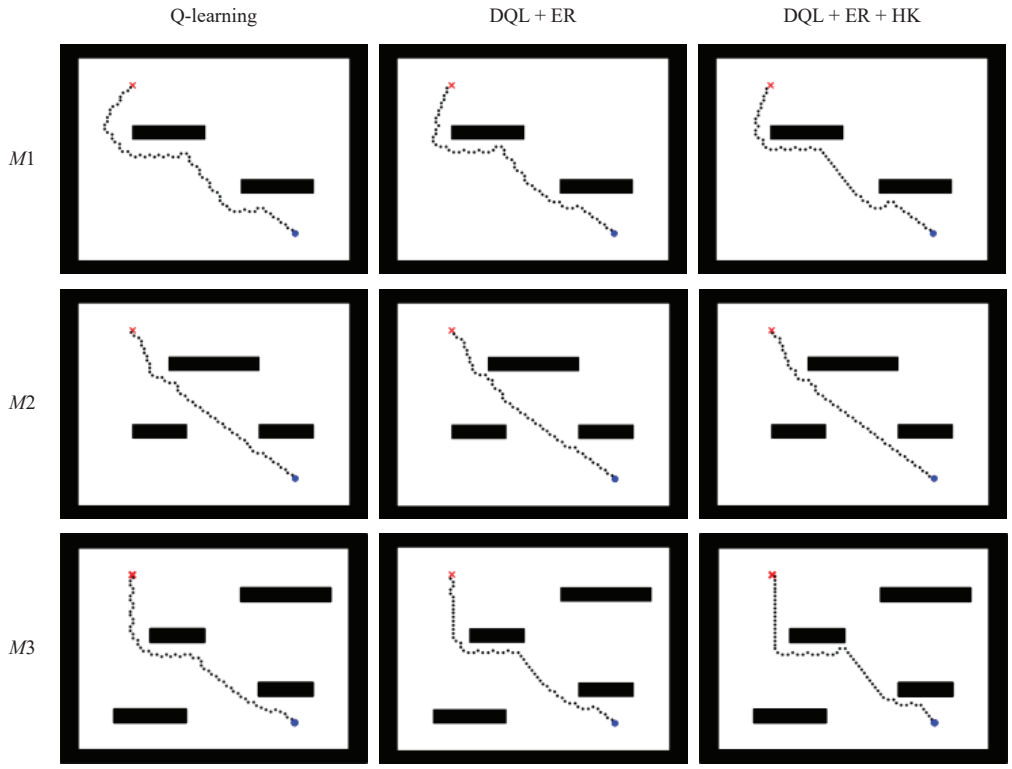

Fig. 6. Nine trajectories that the robot obtained by using three methods in three different maps.

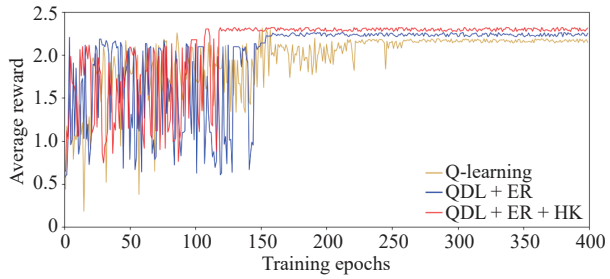

Fig. 7. The plot shows average reward per episode on *M*3 when uses Q-learning, DQL + ER, and DQL + ER + HK.

middle row, the start point is above the obstacle in the lower right corner. In this situation, the robot that uses our method almost goes straight to the end point. But the robot that uses Q-learning with a neural network goes to a detour and the trajectory obtained by DQL with ER is also more tortuous than that obtained by our method. In the bottom row, the start point is on the right side of the original start point. We can see, the left trajectory is the most tortuous. The middle trajectory almost has the same outline with the right one, but the right one is more straight.

Detailed data about the moving step count of arriving to the end point are showed in Table III. Bold numbers are the minimum moving step count used in different situations. It shows that the robot that uses our method can reach the end point with fewer steps.

Table IV shows the average reward that those three models obtained when they start from three arbitrary start point. When the robots have similar trajectories, their average reward values are similar. But the fewer detours the robot takes, the greater the average reward obtained. The robot that uses our method always has the largest average reward.

TABLE III
MOVING STEP COUNT FROM ARBITRARY START POINT

	Model	Start point1	Start point2	Start point3
Moving step count	Q + NN	53	47	77
	DQL + ER	52	42	67
	DQL + ER + HK	**49**	**39**	**61**

TABLE IV
AVERAGE REWARD FROM ARBITRARY START POINT

	Model	Start point1	Start point2	Start point3
Average reward	Q + NN	2.7358	3.1064	2.0260
	DQL + ER	2.75	3.2143	2.2597
	DQL + ER + HK	**2.9787**	**3.5385**	**2.3607**

Test 1 shows that all the models can make robots reach the end point without collision when the robots are initialized from arbitrary start points. By comparing the experiment results, we find our approach provides a better action strategy, which helps the robot go on a shorter path and obtain a larger average reward.

Test 2: Changing the Position of Obstacles in M3.

In Test 2, the position of obstacles in M3 are changed, and the changed maps and obtained trajectories are shown in Fig. 9. The horizontal axis is the three models, the vertical axis is the three obstacle-changed maps. The number of obstacles are changed from 4 to 3 in the changed Map1, and the middle obstacle is located on the line connecting the start point and

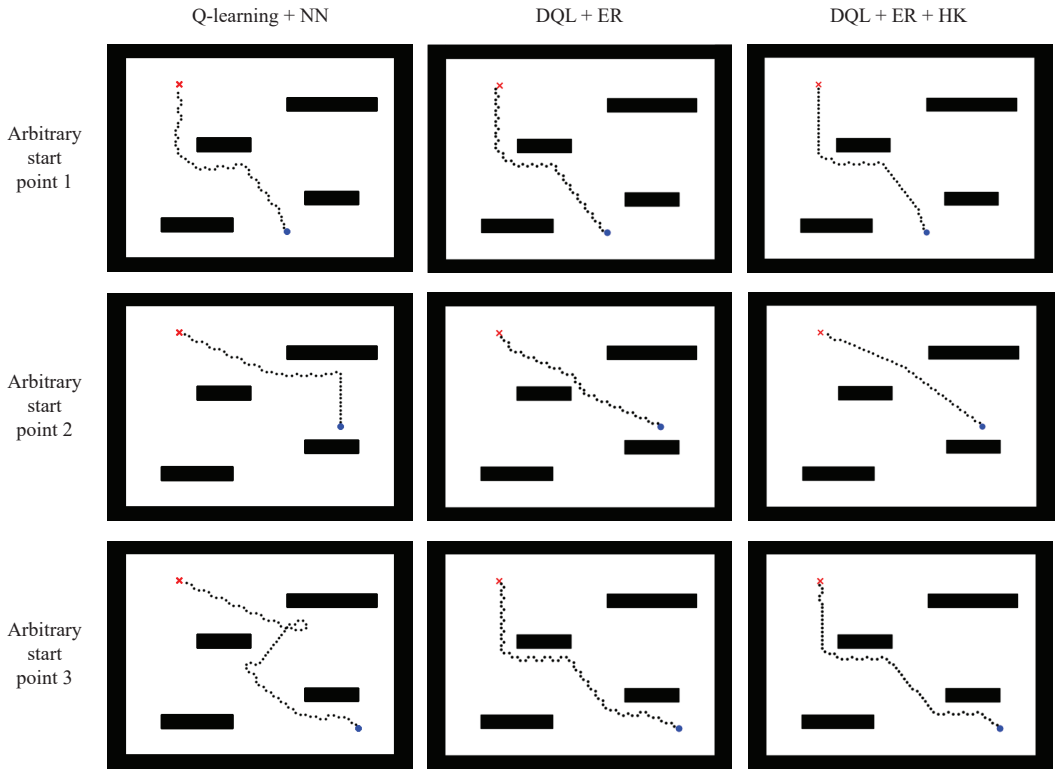

Fig. 8. Nine trajectories that the robot obtained by using three models when starting from three arbitrary start points.

the end point. The left figure in the top row shows the robot has a tendency to stay away from the end point in the process of moving. However, the right two robots bypass the obstacle directly and reach the end point. Their trajectories are similar. The middle row is changed Map2, where the start point is surrounded by a inverted U-shaped obstacle. It can be determined from trajectory that all of the robots explore to the bottom of the inverted U-shaped obstacle at first, and then finally find the exit and reach the end point. During the exploration, the left trajectory shows that the robot collides with obstacles 4 times, with collision points represented by red dots. However, the right two trajectories show that robots walk to the bottom of the inverted U-shaped obstacle, turn around straightly, and reach the end point successfully. The bottom row is changed Map3, where the position of the obstacles has been changed tremendously and there are 10 block obstacles. The left trajectory shows that during the movement, the robot exhibits the phenomenon of turning circles. In contrast, the right two trajectories show that the robots which used DQL explore a shorter moving route. Although the right two models get very similar trajectories, intuitively, DQL with ER and HK can obtain a more straight route.

Table V shows the moving step count the robot takes to reach the end point in three changed maps. Comparing to Q-learning with neural networks, the robot that used that our model takes fewer steps to reach the end point. The more complex the map is, the more obvious the advantages of our model is. But comparing to DQL with ER, our model has a slight advantages in the moving step count. In the changed Map2, those two models have the same moving step count.

TABLE VI shows the average reward those three models obtained when they are in three changed maps. From the table, it can be known that Q-learning with NN can not adapt well to the changed map, and it always has the lowest average reward. Also, it causes the robot to hit obstacles 4 times in the changed Map2, so the average reward is close to 0. The average reward of DQL + ER + HK is always higher than DQL + ER.

In Test 2, robots that used DQL can reach the end point without hitting obstacles, even though obstacles have been changed drastically and the environment is more complex. Furthermore, we find that our model can adapt to new maps better than Q-learning with the neural network and DQL with ER. The reasons why the generalization ability of the model combining the Q-table with the neural network are not optimal are as follow:

TABLE V
MOVING STEP COUNT IN THREE CHANGED MAPS

	Model	Changed Map 1	Changed Map 2	Changed Map 3
Moving step count	Q + NN	66	165	96
	DQL + ER	57	**119**	57
	DQL + ER + HK	**53**	**119**	**55**

TABLE VI
AVERAGE REWARD IN CHANGED MAPS

	Model	Changed Map 1	Changed Map 2	Changed Map 3
Average reward	Q + NN	2.1035	0.0353	1.8438
	DQL + ER	2.4848	0.8739	2.2632
	DQL + ER + HK	**2.6792**	**1.0924**	**2.4727**

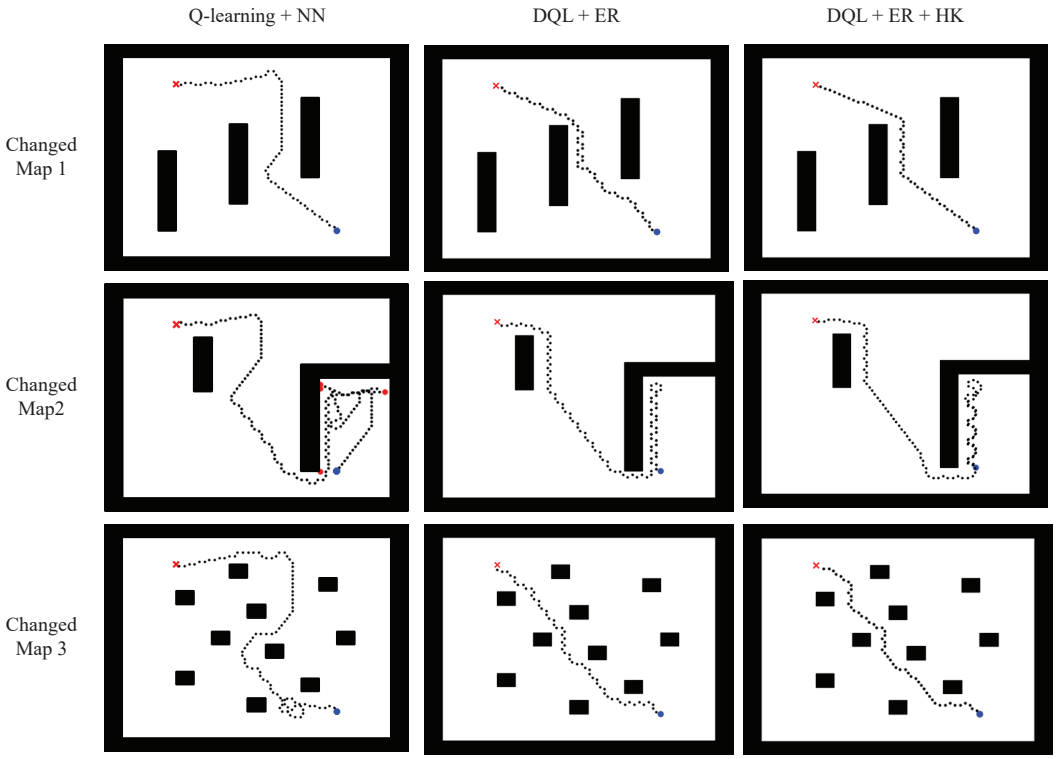

Fig. 9. Nine trajectories that the robot obtained by using three models in the three obstacles-changed maps.

1) Fuzzy states have an impact on the learning space of the robot. In the Q-learning training process, states of the robot had been fuzzed. An excessive degree of fuzzification leads to inaccurate execution of robot actions; a small degree of fuzzification will greatly increase the learning space of the robot and the learning time.

2) Limited amounts of data were used for training neural network. It is critical to train a neural network with sufficient training data. There are some limitations in training the neural network when using the data in the Q-table as training data.

DQL with ER also has good generalization ability. After combining heuristic knowledge, it can move more optimally towards the end point. This, DQL with ER and HK can reach the end point with fewer steps and get a larger average reward.

In DQL with the ER and HK algorithm, the state of the robot is represented by raw data directly, which is used as the input of the neural network. Training of the neural network runs through the moving process of the robot, the data of each step the robot moves is applied to the training of the neural network. Heuristic knowledge guides the behavior of the robot, and it makes the actions selected by the robot more purposeful. However, the trajectories obtained by our approach in the new map is not the shortest one. This is because that when avoiding obstacles, the robots will try to keep away from obstacles, which leads to detours. The accuracy of neural network is also a reason, which will bring uncertainty.

The above two tests verify the generalization and flexibility of our approach, and our approach shows stronger robustness and adaptability than Q-table with neural network and DQL with ER. It can provide a relatively superior action strategy.

V. Discussion of Related Work

Q-learning is widely used in path planning and obstacle avoidance of an intelligent robot. Jaradat et al. proposed a new definition for states space in [26] to reduce the size of the Q-table; in the literature [27], [28], a method based on reinforcement learning and fuzzy logic is proposed. However, those just limit the number of states and cannot solve "curse of dimensionality" completely. Also, the degree of fuzzification limits the learning states of the robot. In our work, a neural network is used to deal with this problem, where the input of the neural network is the state of the robot and consists of raw sonar data.

To solve the problem of slow convergence and low efficiency in path planning, reinforcement learning based on virtual potential field has been proposed in [29]. It regards the unknown environment as a potential field, and Y Zheng et al. propose a new algorithm based on hierarchical reinforcement learning and an artificial potential field [30]. But as we known, the potential field method easily falls into a local optimum.

Neural networks have been used to enhance reinforcement learning's generalization ability, but a large amount of training samples are needed to train a neural network. In [23], [31], researchers use traditional reinforcement learning to generate training samples for neural networks and then, train the neural network with those training samples. There are two training processes in those algorithms, and they are time-consuming. In [32], [33], the neural network is trained while the robot is moving, but because the neural network trains once each step the robot moves, the training efficiency is lowered. Furthermore, the correlation of training samples will affect the representation of the neural network. However, the

experience replay mechanism solves the above problems perfectly, where training samples are stored in replay memory and each step of experience data can be used for updating many weights and the relevance of training data is disrupted. It reduces the burden of collecting prior experience data to train the neural network and improves the efficiency of experience data utilization.

The work of [24] is similar to our work, and the system in this literature is based on deep Q-network which combines a CNN (convolutional neural network) with deep Q-learning. In contrast to our work, this method uses CNN to process image data and takes the obtained result as the input of deep Q-learning. We also add heuristic knowledge based on the original DQL. Heuristic knowledge provides suitable and effective data for training the neural network, and it helps to train a satisfactory neural network.

VI. Conclusion

In this paper, we combined deep Q-learning with experience replay and heuristic knowledge for path planning and obstacle avoidance of intelligent robots. This method has been tested in three different environments, and the robot converges to an optimal strategy faster and reaches the end point with fewer steps than Q-learning with the neural network and normal DQL with ER. The experiments have also shown that our model has better adaptiveness in an unknown environment.

In the future, we will work on the following aspects:

1) Optimizing the planned path for robots. We should design a better strategy to collaborate path planning and obstacle avoidance.

2) Explore more complicated neural network architectures. In this paper, we complete the simulation experiment with the simplest neural network architecture. We hope that more complex neural network architectures can further improve the experimental results.

3) Dynamic obstacle avoidance. In this paper the obstacles are static. We consider dynamic obstacles which will increase the difficulty of robot path planning.

4) Applying our method to real robots. In this paper, we only perform the simulation for our method in an ideal environment, which is hard to satisfy in real life. We will implement our method on a robot in a real environment. Due to the uncertainties in the environment, we may need to adjust our method.

References

[1] M. Liu, F. Colas, F. Pomerleau, and R. Siegwart, "A Markov semisupervised clustering approach and its application in topological map extraction, " in *IEEE International Conference on Intelligent Robots and Systems*, Vilamoura, Algarve, 2012, pp. 4743–4748.

[2] M. Xu, L. Jaesung, and K. Bo-Yeong, "Scalable coverage path planning for cleaning robots using rectangular map decomposition on large environments, " in *IEEE Access*, 2018, pp. 1–1.

[3] B. Sandeep and P. Supriya, "Analysis of fuzzy rules for robot path planning, " in *International Conference on Advances in Computing, Communications and Informatics*, Jaipur, India, 2016, pp. 309–314.

[4] N. Zeng, H. Zhang, Y. Chen, B. Chen, and Y. Liu, "Path planning for intelligent robot based on switching local evolutionary PSO algorithm," *Assembly Automation*, vol. 36, no. 2, pp. 120–126, April. 2016.

[5] E. Masehian and D. Sedighizadeh, "A multi-objective PSO-based algorithm for robot path planning, " in *IEEE International Conference on Industrial Technology*, Vina del Mar, Chile, 2010, pp. 465–470.

[6] H. Shuyun, S. Tang, B. Song, M. Tong, and M. Ji, "Robot path planning based on improved ant colony optimization," *Computer Engineering*, vol. 34, no. 15, pp. 1–3, 2008.

[7] B. Farid, G. Denis, P. Herve, and G. Dominique, "Modified artificial potential field method for online path planning applications, " in *IEEE Intelligent Vehicles Symposium (IV)*, Redondo Beach, California, USA, 2017, pp. 180–185.

[8] L. Liu, R. Shi, S. Li, and W. Jiang, "Path planning for UAVS based on improved artificial potential field method through changing the repulsive potential function, " in *IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, Nanjing, Jiangsu, China, 2016, pp. 2011–2015.

[9] Y. Chen and W. Chiu, "Optimal robot path planning system by using a neural network-based approach, " in *Automatic Control Conference*, Taichung, Taiwan, China, 2016, pp. 85–90.

[10] J. Zhang, J. Zhang, Z. Ma, and Z. He, "Using partial-policy q-learning to plan path for robot navigation in unknown enviroment, " *10th International Symposium on Computational Intelligence and Design (ISCID)*, VOL. 1, pp. 85–90, December, 2017.

[11] V. Babu, U. Krishna, and S. Shahensha, "An autonomous path finding robot using q-learning, " in *International Conference on Intelligent Systems and Control*, 2016, pp. 1–6.

[12] Z. Wu, "Application of optimized q learning algorithm in reinforcement learning," *Bulletin of Science & Technology*, vol. 36, no. 2, pp. 74–76, Feb. 2018.

[13] L. J. Lin, "Reinforcement learning for robots using neural networks, " *Ph.d.thesis Carnegie Mellon University*, 1993.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," *Computer Science*, Dec. 2013.

[15] T. Liang, "A speedup convergent method for multi-agent reinforcement learning, " in *International Conference on Information Engineering and Computer Science*, December, 2009, pp. 1–4.

[16] J. Kober, J. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, September. 2013.

[17] Y. Gao, S. F. Chen, and X. Lu, "Research on reinforcement learning technology: A review," *Acta Automatica Sinica*, vol. 30, no. 1, pp. 86–100, January. 2004.

[18] L. Xue, C. Sun, D. Wunsch, Y. Zhou, and F. Yu, "An adaptive strategy via reinforcement learning for the prisoner u+02bc s dilemma game," *IEEE/CAA Journal of Automatica Sinica*, vol. PP, no. 1, pp. 1–10, 2018.

[19] T. Liu, B. Tian, Y. Ai, L. Li, D. Cao, and F. Y. Wang, "Parallel reinforcement learning: a framework and case study," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 4, pp. 827–835, 2018.

[20] Y. Yusof, H. M. A. H. Mansor, and H. M. D. Baba, "Simulation of mobile robot navigation utilizing reinforcement and unsupervised weightless neural network learning algorithm, " in *Research and Development*, 2016, pp. 123–128.

[21] L. Li, Y. Lv, and F. Y. Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 3, pp. 247–254, 2016.

[22] A. Sharma, K. Gupta, A. Kumar, A. Sharma, and R. Kumar, "Model based path planning using q-learning, " in *IEEE International Conference on Industrial Technology*, Chengdu, China, 2017, pp. 837–842.

[23] S. Parasuraman and S. Yun, "Mobile robot navigation: neural qlearning," *International Journal of Computer Applications in Technology*, vol. 44, no. 4, pp. 303–311, October. 2013.

[24] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks, " in *International Conference on Neural Information Processing Systems*, Doha, Qatar, 2012, pp. 1097–1105.

[25] T. Lei and L. Ming, "A robot exploration strategy based on q-learning network, " in *IEEE International Conference on Real-Time Computing and Robotics*, Angkor Wat, Cambodia, 2016, pp. 57–62.

[26] M. Jaradat, M. Al-Rousan, and L. Quadan, "Reinforcement based mobile robot navigation in dynamic environment," *Robotics & Computer Integrated Manufacturing*, vol. 27, no. 1, pp. 135–149, February. 2011.

[27] L. Cherroun and M. Boumehraz, "Intelligent systems based on reinforcement learning and fuzzy logic approaches, 'application to mobile robotic', " in *International Conference on Information Technology and E-Services*, 2012, pp. 1–6.

[28] H. Boubertakh, M. Tadjine, and P. Y. Glorennec, "A new mobile robot navigation method using fuzzy logic and a modified q-learning algorithm," *Journal of Intelligent & Fuzzy Systems*, vol. 21, no. 1–2, pp. 113–119, 2010.

[29] J. Liu, W. Qi, and X. Lu, "Multi-step reinforcement learning algorithm of mobile robot path planning based on virtual potential field," *Springer*, vol. 728, pp. 528–538, September. 2017.

[30] Y. Zheng, B. Li, D. An, and N. Li, "A multi-agent path planning algorithm based on hierarchical reinforcement learning and artificial potential field, " in *International Conference on Natural Computation*, 2016, pp. 363–369.

[31] D. Luviano Cruz and W. Yu, "Multi-agent path planning in unknown environment with reinforcement learning and neural network, " in *IEEE International Conference on Systems, Man and Cybernetics*, San Diego, USA, 2014, pp. 3458–3463.

[32] Y. Zhao, Z. Zheng, X. Zhang, and Y. Liu, "Q learning algorithm based UAV path learning and obstacle avoidence approach, " in *Chinese Control Conference (international)(CCC)*, Dalian, China, 2017, pp. 3397–3402.

[33] B. Huang, G. Cao, and M. Guo, "Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance, " in *International Conference on Machine Learning and Cybernetics*, Guangzhou, China, 2005, pp. 85–89.

Lan Jiang is currently pursuing the M.S. degree with the School of Information Science, Zhejiang Sci-Tech University, Hangzhou, China. Her current research interests include machine learning and data analysis.

Hongyun Huang received the M.S degree in software engineering from Zhejiang Sci-Tech University in 2015, and the bachelor degree in business administration from Shanghai Jiao Tong University in 2009. She is currently a Data Analyzer at the Center of Multi-Media Big Data of Library, Zhejiang Sci-Tech university. Her current research interests include data analysis, system modeling, AI computing. She has authored and coauthored over 6 papers on the above areas.

Zuohua Ding (M'11) received the M.S. degree in computer science and the Ph.D. degree in mathematics, both from the University of South Florida, Tampa, FL, USA, in 1996 and 1998, respectively. From 1998 to 2001, he was a Senior Software Engineer with Advanced Fiber Communication, Petaluma, CA, USA. He has been a Research Professor with the National Institute for Systems Test and Productivity, Vail, CO, USA, since 2001. He is currently a Professor and the Director with the Laboratory of Intelligent Computing and Software Engineering, Zhejiang Sci-Tech University, Hangzhou, China. His current research interests include system modeling, program analysis, service computing, software reliability prediction, and Petri nets. He has authored and coauthored over 70 papers.