A Reinforcement Learning Method of Obstacle Avoidance for Industrial Mobile Vehicles in Unknown Environments Using Neural Network

Chen XIA*, A. EL KAMEL
LAGIS (UMR CNRS 8219), Ecole Centrale de Lille, Villeneuve d'Ascq, 59650, France
(donaldgreg.x@gmail.com)

Abstract - **This paper presents a reinforcement learning method for a mobile vehicle to navigate autonomously in an unknown environment. Q-learning algorithm is a model-free reinforcement learning technique and is applied to realize the robot self-learning ability. The state-action Q-values are traditionally stored in a Q table, which will decrease the learning speed when large storage memory is needed. The neural network has a strong ability to deal with large-scale state spaces. Therefore, the neural network is introduced to work with Q-learning to ensure the self-learning efficiency of avoiding obstacles for industrial vehicles in unpredictable environments. Experiment results show that an autonomous mobile vehicle using the proposed method can successfully navigate to the target place without colliding with obstacles, and hence prove the self-learning ability of navigation in an unknown environment.**

Keywords - **Neural network, obstacle avoidance, Q-learning, reinforcement learning, unpredicted environments**

I. INTRODUCTION

Traditional fixed industrial robots always have a place in many various industries, while mobile robots have the capability to move around in their working environment. The mobility allows more flexibility in a great number of industrial applications, such as warehouse transportation and distribution. Therefore, increased interest in mobile robotics is spread across all industries, especially in hazardous applications.

Industrial mobile robots should have the autonomous navigation capacity in its working environment, which aims to find a collision free path from a starting point to a target point [1, 2], such as transporting materials from one position to another in a warehouse. Global path planning techniques can be applied when a complete knowledge of the environment is acquired. While robots are working in unknown environments, local path planning techniques, which rely on sensory information of the mobile robots, has successfully proven adequate in achieving the task of obstacle avoidance, such as fuzzy logic control [3], potential field method [4], genetic algorithms [5].

However, classic path planning methods have a common constraint that control strategy needs to be well designed by programmers. When the environment changes or the robot encounters situations that are not considered beforehand by designers, they may make no reaction but execute the predefined coping strategy. This may lead to dangerous or even fatal consequences, such as collision, crash. In this way, such robot is still a kind of machine that perfectly executes what it is taught without good adaptability. What we expect is a truly intelligent autonomous robotic system. Therefore, mobile robots with self-learning ability become a hot research topic.

Robot learning is normally realized by the interaction between the robot and the surrounding environment. Reinforcement learning is a machine learning technique based on trial-and-error mechanisms, improving the performance by getting feedback from the environment [6, 7]. This paper enhances the learning ability of industrial mobile vehicles based on accumulated interacting experiences. Reinforcement learning provides so many learning methods, and Q-learning is one of them [7-9]. However, Q-learning is usually applied to discrete sets of states and actions. In real applications, the state spaces are continuous and large-scale spaces will bring the problems of the generation and the curse of dimensionality. Since neural network (NN) has a good generalization performance and can approximate any functions in any accuracy, it is natural that the implantation of neural network is one of the effective approach for solving the problem of discrete Q-learning generation [8, 10, 11].

Some similar researches use exactly the same environment both to train the robot's learning ability and to test the navigation skills [12, 13, 14]. However, what we really expect is that the robot learns the obstacle avoidance behaviors in some environments and can navigate independently in a completely new unknown environment, which will prove the feasibility and stability of the proposed navigation strategy.

This paper presents an approach of obstacle avoidance learning for mobile vehicles in an unknown environment by developing a neural network based Q-learning architecture (NNQL).

The rest of the paper is organized as follows. Q-learning design is described in Section II. Section III introduces the neural network and presents the architecture of the algorithm of neural network based Q-learning for mobile robot navigation. In Section IV, the experiment is conducted to show the simulation results of the proposed method. Section V draws the final conclusions.

II. Q-LEARNING DESIGN

Reinforcement learning is aimed at learning a mapping from the states of the environment to the robot behaviors. The robot doesn't need previous knowledge about the surrounding environment. It learns about the environment via interacting with it [9]. The structure of reinforcement learning in a mobile robot navigation problem is shown as Fig. 1. The learning system perceives the environment and receives a state that describes it.

Then the system chooses an action from the action space to execute and then moves to a new state. Then, the system receives an immediate evaluation reward as well as perceives the new state. The purpose of the learning system is to find a control policy that maximizes the expected amount of reward during the learning period [7].

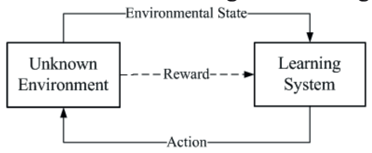

Fig.1. The structure of reinforcement learning

A. Models for mobile vehicles and the environment

The mobile robot is assumed that has three wheels, one in front and two in back. The vehicle is equipped with eight sensors around to perceive the environment, and each sensor is responsible for one region of a range of 45º, as shown in Fig. 2. The total sensor detection range of θ is the interval $[0,2\pi]$.

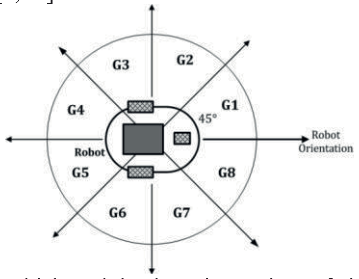

Fig.2. The vehicle and the detection regions of eight sensors

The working environment of the vehicle consists of its target and the obstacles, as shown in Fig.3. The initial vehicle location and the goal are predefined, where the robot will try to reach the goal with free collision path in spite of the presence of obstacles in the environment.

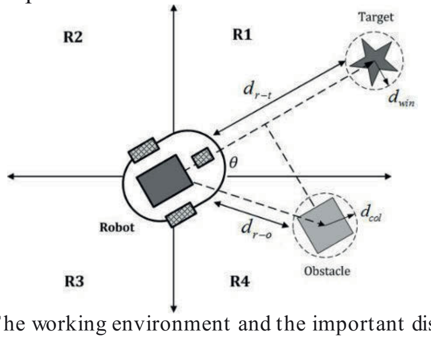

Fig.3. The working environment and the important distances

d_{r-t} is the distance between the robot and the target and d_{r-o} is the distance between the robot and obstacles.

Several basic assumptions concerning the vehicle and the navigation environment are made here [9]:

- The position p, the velocity v, of the robot are known at each time instant.
- The shapes of the obstacles in the robot's detection range are known for the robot at each time instant.
- The position of the target p_{tar} and the region of the target R_g are known at each time instant.

B. States and actions spaces of Q-learning

Q-learning is aimed at learning a mapping from the state input to the action output. In a navigation problem, the robot perceives the state from the environment by means of its sensors, and this state of environment is used by a reasoning process to determine the action to execute in the given state.

The state spaces can be completely defined by the robot sensor information which detects the relative or approximate distances and directions between him and the surrounding obstacles or the target. Hence, a state of environment can be expressed in a vector with eight components:

$$s_t = [d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8] \tag{1}$$

s_t is the state at instant t. $d_i, i = 1,2,\ldots,8$ are the reading distances of each sensor. If no obstacle is found, then $d_i = 0$.

The action spaces are defined by five moving actions of the robot: move forward, turn left at 30º, turn left at 60º, turn right at 30º and turn right at 60º. The five actions are based on the vehicle orientation.

C. The reward function

The reward function measures an immediate feedback for the action taken at a given state. It evaluates how good or how bad the taken action is at a specific situation. Before giving the reward function, one environment state is classified into four different properties, called the state property:

- Safe State (SS): a state where the robot has a low or no possibility of collision with surrounding obstacles.
- Non-Safe State (NS): a state where the robot has a high possibility of collision with some obstacles in the environment.
- Winning State (WS): one of the terminate states when the robot reaches its goal.
- Failure State (FS): one of the terminate states when the robot collides with obstacles.

The reward is given to the robot instantly when it moves from one state to a new state after executing one action. The reward function r is defined as follows:

- Moving from a Non-Safe State to a Safe State: $r = 0.3$.
- Moving from a Safe State to a Non-Safe State: $r = -0.2$.
- Moving from a Non-Safe State to a Non-Safe State but getting closer to the obstacles: $r = -0.4$.
- Moving from a Non-Safe State to a Non-Safe State and getting away from the obstacles: $r = 0.4$.
- Moving to a Winning State: $r = 1$.
- Moving to a Failure State: $r = -0.6$.

D. The Q-value function

The Q-value function expresses the mapping policy from the perceived environment state to the executing action. One state-action Q-value $Q(s_t, a_t)$ corresponds with one specific state and one action in this state. All the Q-values should be initially set to zeros. Then the

Q-values will be updated while training the robot and this can be interpreted as the robot is learning.

At each time instant t, the robot observes its current state of environment s_t, then selects an action a_t. A state-action Q-value $Q(s_t, a_t)$ is now produced. After performing the action a_t, the robot observes the subsequent state s_{t+1}, and also receives an immediate reward r_t. Last, the current $Q(s_t, a_t)$ is updated to its optimal Q-value $Q^*(s_t, a_t)$ according to the following Q-value function [7, 15]:

$$Q^*(s_t, a_t) = Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a_i \in A} Q(s_{t+1}, a_i) - Q(s_t, a_t) \right] \quad (2)$$

α is the learning rate, set between 0 and 1. Setting it to 0 means that the Q-values are never updated, hence nothing is learned. Setting a high value means that learning can occur quickly.

γ is the discount factor with a range of 0 and 1. If γ is close to 0, the robot will tend to consider immediate reward. On the contrary, if γ approaches 1, the robot will take more future reward into account.

E. Action selection strategy

After obtaining the environment information, the vehicle selects an appropriate action to execute according to the action selection mechanism. During learning, it is important to try different actions as much as possible, but the robot has also to solve the dilemma between "exploration" and "exploitation". A good way is to select actions applying the Boltzmann probability distribution.

$$P(a|s) = \frac{e^{Q(s,a)/T}}{\sum_{b \in A} e^{Q(s,b)/T}} \quad (3)$$

T is a temperature parameter between 0 and 1 that determines the stochastic probability of action selection.

Then, during the robot navigation process, in order to exploit the most the policy, the robot takes greedy action selection according to the following equation:

$$a^*(s) = arg \max_{b \in A} Q(s, b) \quad (4)$$

III. THE ALGORITHM OF NEURAL NETWORK Q-LEARNING

Traditional Q-learning is designed for the discrete set of states and actions. However, in the mobile robot navigation tasks where the state spaces are continuous due to the sensory inputs, a large-scale memory space is needed to store all the state-action pairs and the learning speed will decrease. In order to solve this dimensionality problem, the neural network is introduced, since it provides a good generalization performance as a universal function approximator.

A. Architecture of neural network Q-learning

In the proposed NNQL, a three-layer neural network replaces the traditional Q-table and approximates the Q-value function, as shown in Fig.4.

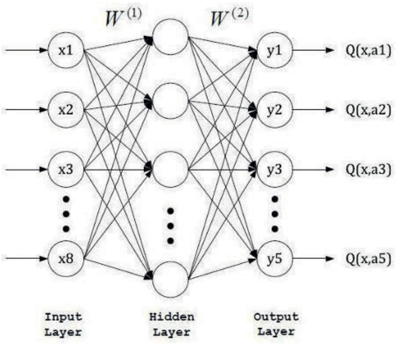

Fig.4. Three-layer neural network architecture

The inputs are the eight reading distances from the robot sensors, representing the perception of environment. The outputs correspond with the five action Q-values. The weight W_1 is used to connect the input layer and the hidden layer, and similarly, the weight W_2 links the hidden layer and the output layer. For the hidden units, the sigmoid function is used.

The feedforward neural network (FFNN) takes the responsibility to calculate the activations of the output units from the input units. Each output unit's activation represents the Q-value for the corresponding action in the given state. In FFNN, the weights W_1 and W_2 always keep unchanged.

B. Backpropagation neural network

The backpropagation neural network (BPNN) is designed to train the neural network by updating the weights W_1 and W_2. Only a neural network with fully trained weights can be used in robot navigation tasks. The weight changes are based on the network's error, the difference between its output for a given input and a target value, what the network is expected to output. The optimal Q-value in Eq. (2) is treated as the target for the output unit corresponding to the selected action. Gradient descent is used to optimize the network error [15].

C. The training process of neural network Q-learning

The neural network based Q-learning can be divided into two processes. The first one is the training process to train the robot learning ability, and the second one is the navigation process to use the trained policy to finish a navigation task.

Training the mobile robot is done by exposing it to different episodes of environments. The greater the number of episodes used to train the robot, the better will be the performance of the robot navigation. Each episode starts by perceiving the current state of the environment. The surrounding obstacle locations are supplied to the robot through its sensors. Once the current state is checked, if it is a Safe State the robot changes its orientation toward the target location, and moves one step forward trying to reach the target in the shortest path. If the current state is Non-Safe State, the robot inputs the current state into the FFNN and outputs all the possible Q-values. According to the Boltzmann action selection mechanism, the robot takes an action and moves to a new state. Then, the robot checks the resulting new state, gets the immediate reward and updates the Q-values accordingly. Then the updated Q-values are sent back to the neural network and the weights of the neural network get updated by using backpropagation algorithm.

Each episode has limited steps. The robot needs to reach the target within the steps. If the robot runs out of the steps and does not reach the target, or if the robot collides with an obstacle or reach the target, the episode is terminated and a new episode is started.

D. The navigation process using the learned algorithm

After training the robot, the resulting policy can be used by the robot for future navigation tasks in various environments.

The robot starts its navigation by finding its current environment state. If it is a Safe State the robot changes its orientation towards the target and moves one step forward. It continues moving until entering a Non-Safe region where adopts the trained policy. The robot uses the FFNN to generate all possible Q-values. The robot takes the action that has the biggest Q-value. After that, the robot finds its new current state and repeats the process until the robot reaches its goal or collide an obstacle.

IV. SIMULATION AND RESULTS

In order to evaluate the proposed method, the simulation experiments are carried out in MATLAB. The mobile vehicle is represented by a rectangle-shaped robot. It is equipped with 8 sensors to observe the environment, as shown in Fig. 1. The environment map has a size of 100 m × 100 m. The obstacles are randomly scattered in the environment and the robot has no prior knowledge of their numbers, sizes and positions. The initial position of the robot is placed at (20, 20) and the target position, a red circle in the map, is found at (90, 90). The velocity of the robot is fixed at 2m/s. The mission of the robot is to start from the initial position and to find an optimal path to arrive at the target position without any collision with any obstacles. If no obstacles are detected, the robot is designed to move directly to the target.

The neural network has three layers: 8 in the input layer, 6 in the hidden layer and 5 in the output. The input is eight reading distances from the robot sensors. The range of sensor detection is 10 m. The output is five action Q-values. The experiment parameters are selected as follows:

- Learning rate: $\alpha = 0.1$,
- Discount factor: $\gamma = 0.15$,
- Maximum temperature: $T_{max} = 0.8$,
- Minimum temperature: $T_{min} = 0.01$.

A. The training process simulation

1000 episodes are set in the learning process and each episode has a maximum of 500 moving steps. All episodes have different configurations of random obstacle positions. A new episode will be started in the following three situations:

- The robot finds a collision free path to the target;
- The robot collides with an obstacle or the map borders;
- The robot runs out the moving steps.

The training process will be terminated when all the learning episodes are finished. Some training episodes are shown in Fig.5.

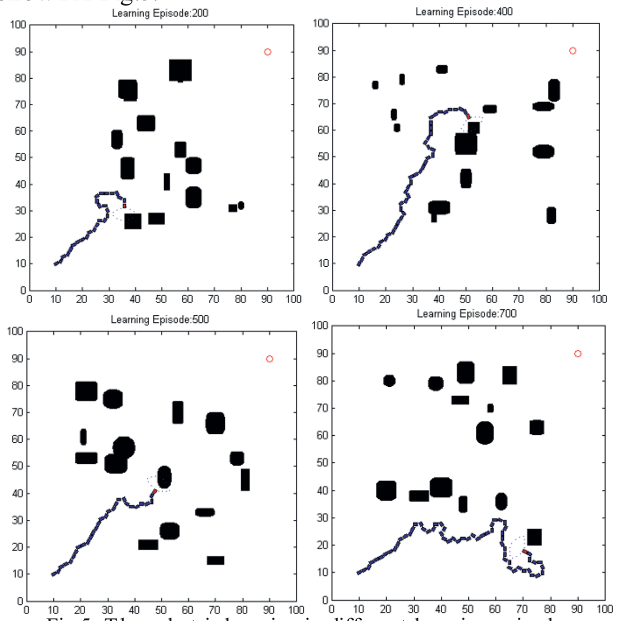

Fig.5. The robot is learning in different learning episodes

Since the proposed algorithm is a trial-and-error method, it is reasonable that a great number of episodes failed due to a collision. In these episodes, the robot learned to navigate in different environments and we can see that the robot tried different actions and the rewards evaluated this action decision, which can help the robot to correct the action selection in the future. During all episodes, the proposed algorithm adopted BPNN to train the weights W_1 and W_2. When the learning process is completed, the weights have been well trained and can be used directly in the following process of robot navigation.

B. Robot navigation process simulation

In the navigation process, the weights W_1 and W_2 have converged to their optimal values. The robot has learned how to behave in front of obstacles. The FFNN is now only adopted, and the robot chooses the best fit action. Then it is the time for the mobile robot to demonstrate its intelligence of independent navigation in an unpredictable environment.

Fig.6 shows that the robot was exposed to a new unknown environment and it succeeded in getting to the target position in a collision free path.

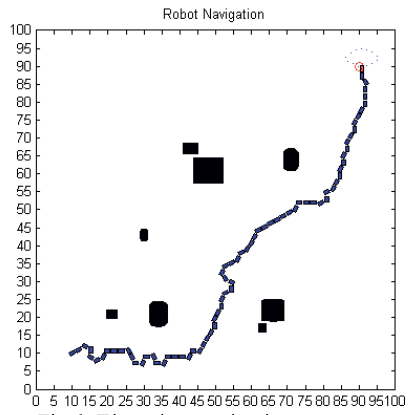

Fig.6. The robot navigation process

Then, in order to test the stability of the proposed method, the robot executed other navigation missions using the same weights and the policy, as shown in Fig.7.

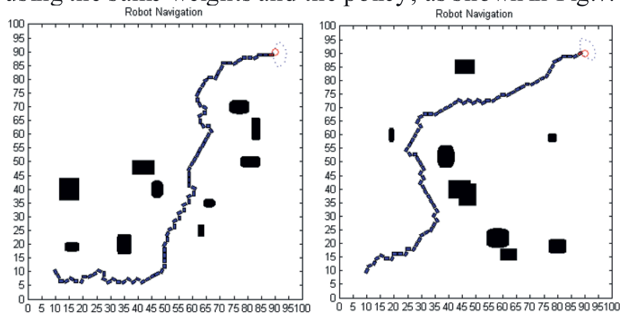

Fig.7. Other navigation processes

The robot navigation environments changed in the above two missions, and the robot completed both tasks successfully. The paths that the robot chose may not be the optimal ones due to a lack of complete knowledge of environment map, but they are still acceptable and perfect enough to meet our expectation. Therefore, the above experiments have proven the feasibility and the stability of the proposed NNQL algorithm.

V. CONCLUSION

This paper explores the mobile vehicles navigation problem in the context of industrial applications by combining reinforcement learning and neural network. Q-learning is applied to enhance the self-learning ability of a mobile robot through trial-and-error interactions with an unknown environment. The neural network is integrated to store and train the large-scale Q-values. Therefore, an intelligent control strategy using neural network based Q-learning is implemented in the paper. The experiment results show the feasibility and the stability of the proposed method. The mobile vehicle can complete navigation tasks safely in an unpredictable environment and becomes a truly intelligent system with strong self-learning and adaptive abilities.

REFERENCES

[1] D. Filliat and J.-A. Meyer, "Map-based navigation in mobile robots: I. A review of localization strategies," *Cognitive Systems Research*, vol. 4, no. 4, pp. 243–282, Dec. 2003.

[2] J.-A. Meyer and D. Filliat, "Map-based navigation in mobile robots: II. A review of map-learning and path-planning strategies," *Cognitive Systems Research*, vol. 4, no. 4, pp. 283–317, Dec. 2003.

[3] S. M. Raguraman, D. Tamilselvi, and N. Shivakumar, "Mobile robot navigation using Fuzzy logic controller," in *2009 International Conference on Control, Automation, Communication and Energy Conservation, 2009. INCACEC 2009*, 2009, pp. 1–5.

[4] S. S. Ge and Y. J. Cui, "Dynamic Motion Planning for Mobile Robots Using Potential Field Method," *Autonomous Robots*, vol. 13, pp. 207–222, 2002.

[5] Y. Hu and S. X. Yang, "A knowledge based genetic algorithm for path planning of a mobile robot," in *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*, 2004, vol. 5, pp. 4350–4355 Vol.5.

[6] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd edition. Prentice Hall, 2010.

[7] M. E. Harmon and S. S. Harmon, *Reinforcement Learning: A Tutorial.* 1996.

[8] K. Macek, I. Petrovic, and N. Peric, "A reinforcement learning approach to obstacle avoidance of mobile robots," in *7th International Workshop on Advanced Motion Control, 2002*, 2002, pp. 462–466.

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

[10] B.-Q. Huang, G.-Y. Cao, and M. Guo, "Reinforcement Learning Neural Network to the Problem of Autonomous Mobile Robot Obstacle Avoidance," in *Proceedings of 2005 International Conference on Machine Learning and Cybernetics, 2005*, 2005, vol. 1, pp. 85–89.

[11] G.-S. Yang, E.-K. Chen, and C.-W. An, "Mobile robot navigation using neural Q-learning," in *Proceedings of 2004 International Conference on Machine Learning and Cybernetics, 2004*, 2004, vol. 1, pp. 48–52 vol.1.

[12] C. Li, J. Zhang, and Y. Li, "Application of Artificial Neural Network Based on Q-learning for Mobile Robot Path Planning," in *2006 IEEE International Conference on Information Acquisition*, 2006, pp. 978–982.

[13] J. Qiao, Z. Hou, and X. Ruan, "Q-learning Based on Neural Network in Learning Action Selection of Mobile Robot," in *2007 IEEE International Conference on Automation and Logistics*, 2007, pp. 263–267.

[14] J. Qiao, Z. Hou, and X. Ruan, "Application of reinforcement learning based on neural network to dynamic obstacle avoidance," in *International Conference on Information and Automation, 2008. ICIA 2008*, 2008, pp. 784–788.

[15] R. Rojas, The Backpropagation Algorithm in Neural Networks: A Systematic Introduction. Springer, 1996, pp. 151–182.