Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

A Global Path Planning Algorithm for Robots Using Reinforcement Learning

Penggang Gao, Zihan Liu, Zongkai Wu, Donglin Wang
Department of Engineering
Westlake University
Hangzhou, Zhejiang Province, China
gaopenggang, liuzihan, wuzongkai, wangdonglin@westlake.edu.cn

Abstract— **Path planning is the key technology for autonomous mobile robots. In view of the shortage of paths found by traditional best first search (BFS) and rapidly-exploring random trees (RRT) algorithm which are not short and smooth enough for robot navigation, a new global planning algorithm combined with reinforcement learning is presented for robots. In our algorithm, a path graph is established firstly, in which the paths collided with the obstacles are removed directly. Then a collision-free path will be found by Q-Learning from starting point to the goal. The experiment results illustrate that it can generate shorter and smoother paths, compared with the BFS and RRT algorithm.**

Index Terms— **Global path planning, Random Sampling, Reinforcement Learning**

I. Introduction

Path planning [1-2] which is aiming to find a path for mobile robots is one of the main research contents of motion planning [3]. Recently autonomous vehicles are getting more and more interest in industry and academia [4]. In the traditional navigation scheme for the self-driving car, path planning is one of key technologies in completing navigation tasks. For instance, some goods or materials can be transported automatically from one place to another by intelligent vehicles, which input new energy into the logistics field. Besides, the path planning function is significant for indoor mobile robots. Taking service robots as an example, path planning is the basic requirement when they fulfill cleaning, guest guiding, dining service tasks.

Path planning can be classified into global path planning and local path planning. A rough path is provided by global path planning module for robots. The path length is an important evaluation standard for the global path. Generally speaking, robots cost less energy when following the shorter paths. For local path planning, the path needs to meet dynamics, kinematics and orientation constraints. More importantly, the local path needs to avoid all obstacles, including mobile obstacles.

Path planning can be classified into traditional algorithms and intelligence algorithms based on machine learning as well. In the traditional algorithm, there are two kinds of main methods including algorithms based on sampling [11] and algorithms based on optimization [25]. Recently, many researchers try to solve path planning problems using machine learning. Deep learning can be used to find a path. Inputting labeling information and environment data like vision information, a path can be obtained for robots in an unseen environment after training [5]. The combination of reinforcement learning and path planning is also a promising research direction. With current state and environment, a reasonable path can be selected using reinforcement learning [17-18]. The advantage of deep reinforcement learning is that we do not need to annotate large amounts of data [6]. We only need to specify the planning objectives like avoiding collision, arriving the end point, finding the shortest path and so on. Then the robot will start to try in the environment, and network is updated iteratively.

II. Related work

With the development of many years, a great number of traditional global planning algorithms has been presented. Best first search (BFS) [7] is a heuristic search algorithm that selects the closest path node to the goal. It runs relatively fast, but it can not guarantee that the shortest path will be found. Dijkstra [8] has developed a method that selects the node closest to the initial node, which is able to find the shortest path. However, it has a large calculation burden comparing with BFS and A* [10] algorithm. The probabilistic roadmap (PRM) [9] is a graph-based search method and can be used to find the shortest path on the map which converts continuous space into discrete space, combined with a search algorithms such as A* that is combining BFS and Dijkstra algorithm. PRM is used widely, because it can find a short path in real-time. The rapidly-exploring random trees (RRT) [11] algorithm which performs well in high-dimensional space can find a path in real-time, while it has difficulty to get a competitive solution in the path length and smoothness compared with Dijkstra or PRM algorithm.

Reinforcement learning injects fresh energy into robot planning research, and many research results have been achieved in recent years. In [12], Panov A I et al designs several Q-network to complete path planning tasks on the grid maps. Zhang Q et al modifies the coefficient of Bellman equation [13] to achieve the purpose of quick convergence to the right Q function when using Q-Learning to complete the path planing tasks in a grid map. The classical Q-Learning algorithm is improved by Konar A presuming that the distance from the current state to both the next state and the goal can be obtained [14]. It takes less time, and the number of right angles will decrease. Zucker M et al improves the traditional reinforcement learning to guide robots to find a path in maze game [15]. Feldmeier J uses a hierarchical structure [16] to combine reinforcement learning with a dimensional emotional mode and find a path in the maze game through his method. Because these algorithms are based on a grid map, they are hard to be used in real intelligent vehicles. An algorithm [17] is presented by Aranibar D B et al to complete motion planning task using Q-Learning. The generated path satisfies the non-holonomic constraints, and the orientation error is small when the robot arrives at the end point. Igarashi H et al formulates the local path planning problem as a discrete optimization problem [18] where Q-Learning is used to learning weight parameters.

Some researches also use deep reinforcement learning for path planning. Deep learning is good at perception and reinforcement learning performs well in decision-making. The combination of two learning method can get amazing results. Lei X et al [19] applies deep Q-network (DQN) [20] proposed by DeepMind in 2016 to dynamic path planning of unknown environment. The convolution neural network (CNN) [21] extracts the feature of environment with the input data, including the angle, distance information of lidar data, then Q-Learning is to make policies according to the feature. In [22], artificial neural network and Q-Learning was combined by Dubey A D et al to complete planning tasks for robots. In this algorithm, the training data was generated by Q-Learning, then these data was inputted to an artificial neural network to learning a path. Bae H et al [23] also uses CNN and DQN to enable multiple robots to complete the path planing tasks at the same time in the simulation maze environment.

In this paper, we use random sampling to substitute the grid map. It can generate more suitable states for Q-Learning, allowing the algorithm to be used for the real intelligent vehicles, and it generates shorter and smoother path. On the other hand, our algorithm has a small number of states and actions for Q-Learning, which means that it is not necessary to replace the Q-table with a neural network.

III. Establishment of Path Graph

To build the path graph, accepted-rejected sampling [24] is used in our algorithm, which can convert a continuous space problem into a discrete space problem and it is a gorgeous method to learn about the environment for robots. These points are generated randomly on the map. A point will be deleted if it is located in the obstacle region. The process of random sampling continues until the expected number of points is reached. These collision-free points will be the states of Q-Learning.

Sampling random method can save computation as well, because it has fewer states for reinforcement learning compared with the methods based on the grid map. Another advantage is that random sampling is a good way to avoid obstacles. It has less computing burden than the obstacle avoidance algorithm based on the optimization method [25]. It is also more reliable than the force-based method like the artificial potential field [26]. In this kind of force-based obstacle avoidance algorithm based on force, the end point generates gravity force, and obstacles generates repulsive force. The end point produces great power when the robot is far from the end point, so that robots may collide with obstacles in this case. Besides, the robots can not reach the goal when the obstacle is near the end point, because the obstacles produce great repulsion. The last point is that robots will never reach the goal and keep oscillating near obstacles, when gravity and repulsion are equal in magnitude and are opposite in direction.

After random sampling, each point will be connected with other points, and the paths collided with the obstacles are removed directly. If we do not remove these paths and enable the algorithm to learn how to avoid obstacles, the results of finding a collision-free path by Q-Learning in our experiments will be unsatisfactory. It is more reliable to remove these paths directly.

Given the query point, the k nearest instance points from the collision-free paths need to be found for constructing Q-table. There are some algorithms to find k nearest path points. Considering the real-time performance of the algorithm, the method shown in **Algorithm 1** is used. During the process, euclidean distance is used as the distance measure.

In some cases, some query points have a small number of path points from the collision-free paths, which is less than the expected number. In this case, P will be returned directly. In the worst case, $O(n)$ time complexity can also be achieved. At last, the quick sort algorithm [27] proposed by British computer scientist Tony Hoare is used in our algorithm.

In complex environments, there are just a few collision-free paths for some path points, even some points have no connection with other points. In either case, the number of actions is less than the expected action number for reinforcement learning. To solve this problem, repeated actions are introduced for these states. The action is randomly selected in the existing actions until the number of actions corresponding to the states is the same as the expected action number. The isolated path points are removed directly, and the states corresponding to these points will not appear in the Q-table. The final path graph is shown in the Fig.1. Then the Q-Learning algorithm can be carried out.

Algorithm 1 Find the k nearest path points

```
Input:
    the query point: $P_Q$;
    N path points: $P = \{P_1, P_2...P_i...P_n\}$;
Output:
    the $k$ nearest path points $P_k$ or $P$;
1: Initialize parameter $k$;
2: Initialize the distance $D = \{D_1, D_2...D_i...D_n\}$ between
   the $P_Q$ and $P_i$;
3: Initialize $S_a = [\ \ ]$,$S_b = [\ \ ]$;
4: if $n <= k$ then
5:     return $P$;
6: else
7:     Calculate $D$;
8:     Randomly select a number $D_r$ in $D$;
9:     for each $D_i \in D$ do
10:        if $D_r < D_i$ then
11:            add $D_r$ to $S_a$;
12:        else
13:            add $D_r$ to $S_b$;
14:        end if
15:    end for
16:    if the size of $S_a >= k$ then
17:        Quickly sort $S_a$ in ascending order
18:        add the points to $P_k$ according to the first $k$ members in $S_a$
19:        return $P_k$;
20:    else
21:        Quickly Sort $S_a$ and $S_b$ in ascending order
22:        Add the path points to $P_k$ according to $S_a$;
23:        Add other path points to $P_k$ according to $S_b$ until
           the number of $P_k$ is up to $k$;
24:        return $P_k$;
25:    end if
26: end if
```

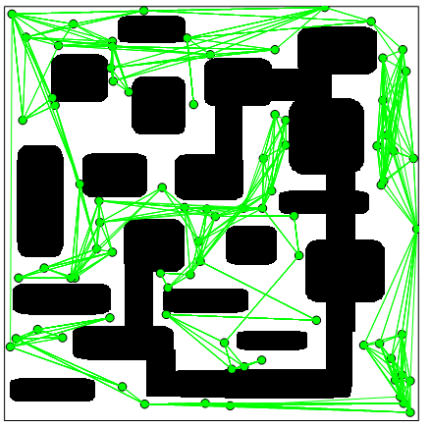

Fig. 1: An example of path graph under our algorithm

Algorithm 2 Q-Learning for finding the path

```
1: Initialize parameters;
2: Initialize Q-table;
3: for each episode do
4:     Set the starting state and the end state according to the
       positions of starting and end points
5:     while true do
6:         $step+ = 1$
7:         Choose action according to $\varepsilon$-greedy policy
8:         Get $r, s'$ and judge whether to reach the end point
9:         Get $Q(s, a)$ from Q table
10:        if state $\neq$ end state then
11:            $Q(s', a) = r + \gamma * maxQ(s', a)$
12:        else
13:            $Q(s', a) = r$;
14:            $done = true$
15:        end if
16:        $Q(s, a) \leftarrow Q(s, a) + \alpha * (Q(s', a) - Q(s, a))$
17:        if $done == true$ then
18:            $Step = 0$;
19:            Break;
20:        end if
21:        if $step > max\_step$ then
22:            $Step = 0$;
23:            Break;
24:        end if
25:    end while
26: end for
```

IV. Search for the path by Q-Learning

Global path planning can be considered as the Markov decision processes. The robots need to select the next path point, which makes path short and smooth, according to the current location and other information. After training, reinforcement learning can make rational action according to current state and environment. Then the best path will be remembered by this learning algorithm.

Q-Learning is used to find a collision-free path in our work, and Q-table is the key to this algorithm. In our algorithm, path points are regarded as states, and selecting next path points are taken as actions. In complex environments, the number of the paths corresponding to some path points is less than the expected action number, so the repeated states are used. We randomly select the states from the states found by **Algorithm 1** until the number of actions is up to the expected. Then Q-table can be built according to the number of the states and the actions.

The algorithm for finding the path using Q-Learning is shown in **Algorithm 2**. Because of ε-greedy policy, the actions which have the biggest Q value will not be selected all the time, and other actions will be chosen randomly. The max iteration number that is the max_step in the **Algorithm**

2 is set in each episode. Otherwise, it is hard to estimate whether the collision-free path can be found.

V. Experiment

For evaluating the effectiveness of our algorithm, various experimental scenarios are chosen for testing the algorithm, including some complex experimental scenarios. We mainly analyze the experimental results by comparing our algorithm with BFS algorithm and RRT algorithm.

A. Experimental environment

Both the physical and the simulation experiments are conducted in this work. The simulation experiment is performed in Matlab R2016a. In the test scenario, the black area represents obstacles , and the white area is accessible, as shown in Fig.3. The length and width of the map are both 500. The starting coordinates is (10, 10), and ending coordinates is (490, 490). Totally, we set six test scenarios, which aims to show the effect of our algorithm in different complex scenario.

Our algorithm also has been tested in Autolabor Pro robot platform, equipped with 16-line lidar. This algorithm can be executed after obtaining points cloud map from mapping module which mainly relies on the simultaneous localization and mapping and knowing the accessible area as well as obstacle area. It is shown in Fig.2.

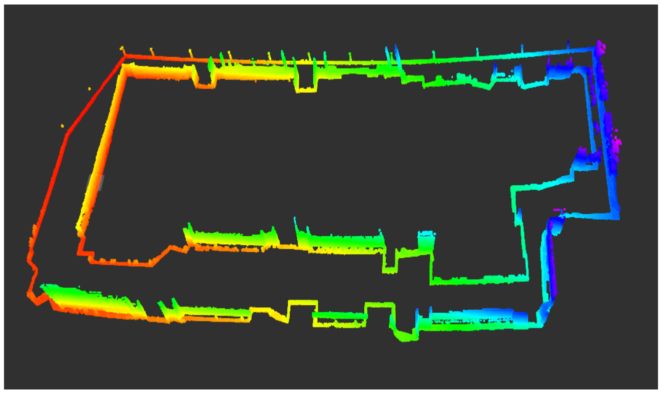

Fig. 2: Point cloud map. The length is more than 400 meters and the width is more than 100 meters

Parameters are set as follows. During these processes, the reward when the robot arrives at the next point is set as -2, in order to make the path short. The robot gets the reward 30 when it reaches the goal. For other parameters, ε and γ are set as 0.9, and α is 0.1. The number of max iteration is 100. At the training stage, the number of the episode is 5000. 80 random points are generated on the map and every state has 10 actions.

B. Evaluation criterion

In this work, two evaluation standards are used, including the path length and path smoothness.

- **Path Length (PL)**. Its mathematical expression is as follows

$$\begin{aligned} Dist(l) &= \sum_{i=1}^{n} d(P_i, P_{i+1}), \\ l &= \{P_1, P_2 ... P_i ... P_n\}. \end{aligned} \tag{1}$$

- **Path Smoothness (PS)**. There are two adjacent line segments. The deflection angle is the angle of the extension of one line to the other. A smaller sum of all deflection angles in the path means a smoother path. Robots consume less energy when tracking a smooth path. Its mathematical expression is as follow

$$\begin{aligned} Smoo(l) &= \sum_{i=2}^{n-1} |\theta_i|, \\ l &= \{P_1, P_2 ... P_i ... P_n\}. \end{aligned} \tag{2}$$

C. Experimental results and analysis

Our algorithm has been tested in 6 scenarios. The PL is shown in Table I, and the PS is shown in Table II. The average length of our algorithm reduced by 219.7584 comparing with BFS algorithm and reduced by 89.112 comparing with RRT algorithm. In our evaluation of Table 2, the less PS, the smoother the path. The PS of RRT algorithm is three times larger than that of our algorithm, and PS of BFS algorithm is 2.4 times that of our algorithm.

TABLE I: Comparison of PL among three algorithms

Test Scenario	BFS	RRT	Our algorithm
1	1.2035×10^3	939.3274	842.7834
2	839.9526	856.2605	773.8429
3	1.2666×10^3	951.1101	897.3120
4	865.5471	807.9231	707.5923
5	1.1542×10^3	1.0747×10^3	924.0190
6	1.3598×10^3	1.2764×10^3	1.2255×10^3

TABLE II: Comparison of PS among three algorithms

Test Scenario	BFS	RRT	Our algorithm
1	7.3573	5.9689	3.4346
2	4.4680	7.8619	1.5593
3	8.9582	11.6482	5.2422
4	5.3286	9.5898	1.4244
5	8.9421	15.4581	2.1080
6	17.5055	15.5125	8.1485

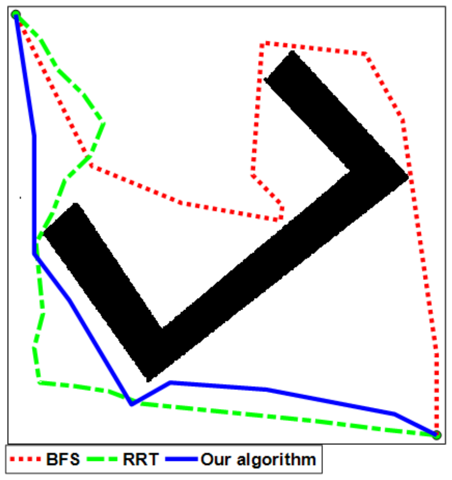

Fig. 3: Comparison of three algorithms in test scenario 1

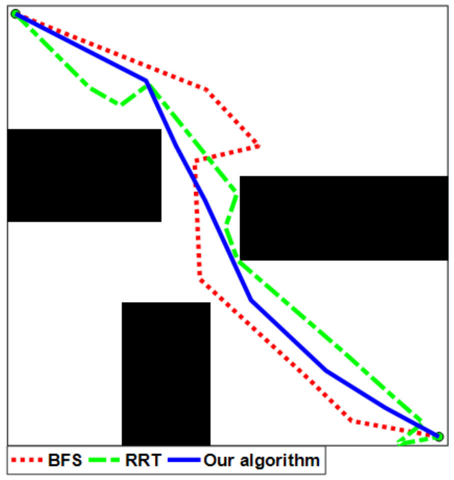

Fig. 6: Comparison of three algorithms in test scenario 4

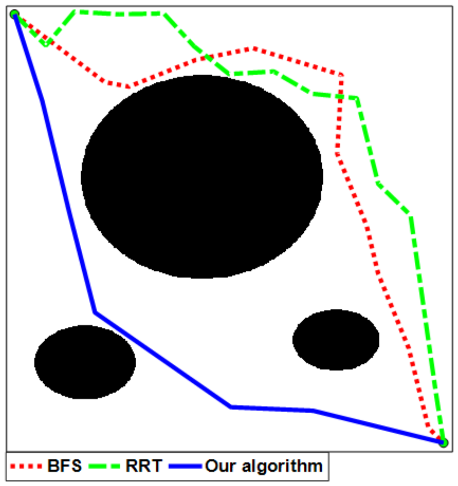

Fig. 4: Comparison of three algorithms in test scenario 2

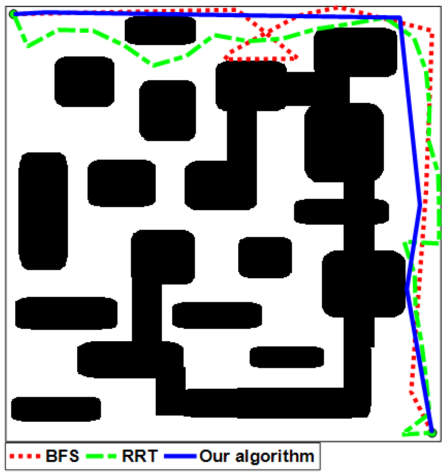

Fig. 7: Comparison of three algorithms in test scenario 5

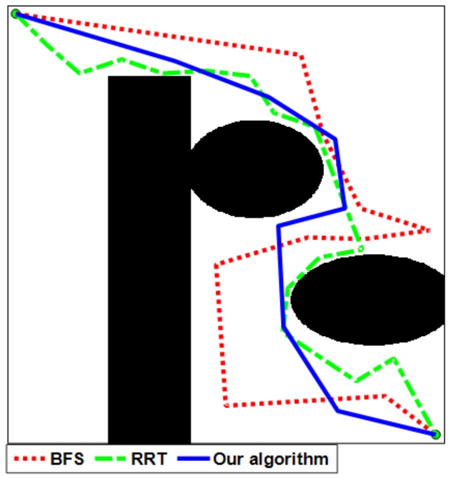

Fig. 5: Comparison of three algorithms in test scenario 3

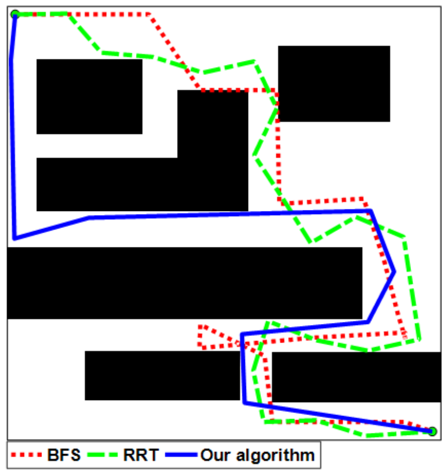

Fig. 8: Comparison of three algorithms in test scenario 6

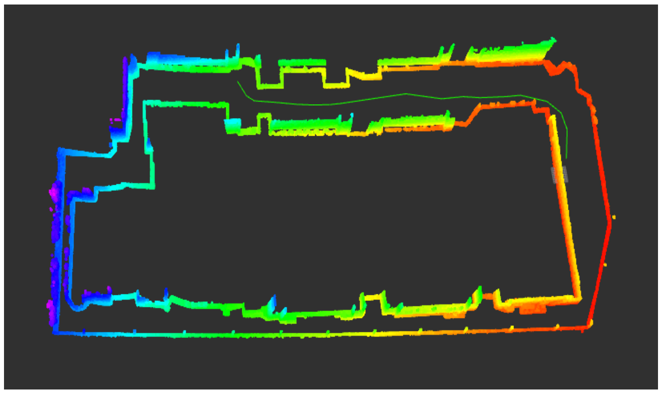

Fig. 9: The test results in real robots. The green line is the path found by our algorithm

Our algorithm can find a path which avoids passing into the inside of the concave obstacle. On the contrary, the BFS algorithm is greedy, in which the robot will move directly towards the goal and return when the current road is blocked, as shown in Fig.3. Unreasonable paths in which robots will move to the starting point shown in Fig.5, Fig.7 and Fig.8 are found by BFS, due to the similar reason.

Compared with our algorithm, the RRT algorithm will find a longer and zigzag path as shown in Fig.4 and Fig.8, because of its randomness. In test scenario 4 and 5, the paths detour near the end point and are shown in Fig.6 and Fig.7. In our method, since a global path graph is obtained, robots can get better understanding of environment, which help find a shorter and smoother path. Mostly, the PS of the shorter path is smaller.

Physical experiment is conducted in the outdoor environment. After obtaining the large scale point cloud map, lidar points located on the ground wwould be removed by the ground segmentation algorithm, shown in Fig.2. Knowing the end point, a path can be found by our path algorithm for robots, which is shown in the Fig.9.

VI. Conclusion and Future Work

The Q-Learning, a classical reinforcement learning algorithm, is used to find a global path for robots in this research. Shorter and smoother paths can be obtained by our algorithm compared with BFS and RRT algorithm. As a part of future work, we consider modifying the update equation of Q-Learning to obtain a path which is far from the obstacles area. When the path is close to obstacles, the agent can get a negative reward. If this reward is added, the robot will be guaranteed from dangerous.

VII. Acknowledgements

This work is support by the Natural Science Foundation of Zhejiang province, China (Grant No.LZ19F020001).

References

[1] Nardi L, Stachniss C. Uncertainty-Aware Path Planning for Navigation on Road Networks Using Augmented MDPs[C]//Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA). 2019.

[2] Xue Y, Sun J Q. Solving the path planning problem in mobile robotics with the multi-objective evolutionary algorithm[J]. Applied Sciences, 2018, 8(9): 1425.

[3] Latombe J C. Robot motion planning[M]. Springer Science & Business Media, 2012.

[4] Litman T. Autonomous vehicle implementation predictions[M]. Victoria, Canada: Victoria Transport Policy Institute, 2017.

[5] Wu P, Cao Y, He Y, et al. Vision-based robot path planning with deep learning[C]//International Conference on Computer Vision Systems. Springer, Cham, 2017: 101-111.

[6] Tai L, Liu M. Towards cognitive exploration through deep reinforcement learning for mobile robots[J]. arXiv preprint arXiv:1610.01733, 2016.

[7] Dechter R, Pearl J. Generalized best-first search strategies and the optimality of A[J]. Journal of the ACM (JACM), 1985, 32(3): 505-536.

[8] Dijkstra E W. A note on two problems in connexion with graphs[J]. Numerische mathematik, 1959, 1(1): 269-271.

[9] Boor V, Overmars M H, Van d S A F. The Gaussian sampling strategy for probabilistic roadmap planners[J]. 1999.

[10] Nannicini G, Delling D, Liberti L, et al. Bidirectional A* search for time-dependent fast paths[C]// International Workshop on Experimental and Efficient Algorithms. 2008.

[11] LaValle S M. Rapidly-exploring random trees: A new tool for path planning[J]. 1998.

[12] Panov A I, Yakovlev K S, Suvorov R. Grid Path Planning with Deep Reinforcement Learning: Preliminary Results[J]. Procedia Computer Science, 2018, 123: 347-353.

[13] Zhang Q, Li M, Wang X, et al. Reinforcement Learning in Robot Path Optimization[J]. JSW, 2012, 7(3): 657-662.

[14] Konar A, Goswami I, Singh S J, et al. A Deterministic Improved Q-Learning for Path Planning of a Mobile Robot[J]. IEEE Trans. Systems, Man, and Cybernetics: Systems, 2013, 43(5): 1141-1153.

[15] Zucker M, Bagnell J A. Reinforcement planning: RL for optimal planners[C]//Robotics and Automation (ICRA), 2012 IEEE International Conference on. IEEE, 2012: 1850-1855.

[16] Feldmaier J, Diepold K. Path-finding using reinforcement learning and affective states[C]//The 23rd IEEE International Symposium on Robot and Human Interactive Communication. 2014: 543-548.

[17] Aranibar D B, Alsina P J. Reinforcement learning-based path planning for autonomous robots[C]//EnRI-XXIV Congresso da Sociedade Brasileira de Computae ao. 2004: 10.

[18] Igarashi H. Path planning of a mobile robot by optimization and reinforcement learning[J]. Artificial Life and Robotics, 2002, 6(1-2): 59-65.

[19] Lei X, Zhang Z, Dong P. Dynamic Path Planning of Unknown Environment Based on Deep Reinforcement Learning[J]. Journal of Robotics, 2018, 2018.

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

[21] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.

[22] Dubey A D, Mishra R B, Jha A K. Path planning of mobile robot using reinforcement based artificial neural network[J]. International Journal of Advances in Engineering & Technology, 2013, 6(2): 780.

[23] Bae H, Kim G, Kim J, et al. Multi-Robot Path Planning Method Using Reinforcement Learning[J]. Applied Sciences, 2019, 9(15): 3057.

[24] Casella G, Robert C P, Wells M T. Generalized accept-reject sampling schemes[M]//A Festschrift for Herman Rubin. Institute of Mathematical Statistics, 2004: 342-347.

[25] Rusmann C, Hoffmann F, Bertram T. Integrated online trajectory planning and optimization in distinctive topologies[J]. Robotics and Autonomous Systems, 2017, 88: 142-153.

[26] Khatib O. Real-time obstacle avoidance for manipulators and mobile robots[M]//Autonomous robot vehicles. Springer, New York, NY, 1986: 396-404.

[27] Hoare C A R. Quicksort[J]. The Computer Journal, 1962, 5(1): 10-16.