SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348 – 8549 /doi:10.14445/23488549/IJECE-V8I1P101
Volume 8 Issue 1, 1-5, January 2021

Autonomous Navigation for Mobile Robot Based on Reinforcement Learning

Roan Van Hoa*[#], Dinh Thi Hang[#], Tran Quoc Dat[#], Tran Dong[#], Tran Thi Huong[#]

[#]*University of Economics - Technology for Industries, Viet Nam*

Received Date: 28 November 2020
Revised Date: 04 January 2021
Accepted Date: 09 January 2021

Abstract *- Reinforcement learning is a subset of machine learning that deals with learning decisions from the environment's rewards. Classic reinforcement learning algorithms are usually applied to small sets of states and actions. However, in real applications, the state spaces are large, bringing the problems of generalization and the curse of dimensionality. In this paper, we integrate the neural network into reinforcement learning methods to generalize the value of all the states. Simulation results on the Gazebo framework show the feasibility of the proposed method. The Robot can complete navigation tasks safely in an unpredicted dynamic environment and becomes a truly intelligent system with strong self-learning and adaptive abilities.*

Keywords - *Artificial intelligence, autonomous navigation, mobile robots, reinforcement learning.*

I. INTRODUCTION

A mobile robot's automatic navigation can be divided into three subsystems: information perception, behavioral decision making, and manipulation control. Path planning is the basis of mobile robot navigation and control [1, 2]. The goal of route planning for the mobile Robot is to find the path from the current location to the target location. The path should be as short as possible; the road's smoothness must meet the dynamics of the mobile Robot and the safe path without collision [3]. Depending on the amount of environmental information known in the road planning process, road planning can be divided into global road planning and local road planning [4]. There are many ways to plan your way. According to specific algorithms and strategies, path planning algorithms can be divided into four categories: pattern matching, artificial lead field, map construction, and artificial intelligence [5, 13, 14]. Each type of route planning algorithm has an optimal application scenario and limitations. Current route planning of mobile robots is mainly based on their surroundings. In addition to the limitations of traditional path planning, robots cannot complete their learning and judgment in complex environments, a bottleneck in developing research in the field [6]. Therefore, it is especially important to develop a road planning method with low dependence on the environment, quickly adapting to the surrounding environment.

Endowing robots with human-like abilities to perform specific skills smoothly and naturally is one of the important goals of robotics to cover a large range of real-life missions such as delivery, search or rescue missions, security surveillance. Such missions require different levels of autonomy in the navigation to react to different dynamic factors such as environmental conditions changes. One of the most common approaches is to train mobile robots via multiple expert demonstrations. These demonstrations provide robots prior experiences from examples, defined as a sequence of state-action pairs recorded during the teacher's demonstration of the desired robot behavior. However, in some cases, such as exploring hazardous environments, no prior experience or demonstrations are available to mobile robots.

In this paper, we present a reinforcement learning (RL) based self-learning algorithm in unknown environments. Experiments are conducted on autonomous navigation tasks for mobile robots. More specifically, we introduce a neural network structure to generalize and approximate all the states of RL algorithms' value. The experiments are conducted on the Gazebo simulator using its open-source extension Gym to perform autonomous navigation tasks [7].

The rest of this paper is organized as follows. Section 2 gives an overview of the related works. Section 3 describes the general methodology and model-free RL-based methods. Then, section 4 introduces some experimentation results followed by a discussion of the proposed method's efficiency. Finally, Section 5 concludes our paper and presents our future works.

II. RELATED WORK

Autonomous navigation is a fundamental and critical research area of mobile robotics. Different reinforcement learning algorithms have been applied to this problem, such as the fuzzy Q-learning presented in [8] or the dueling network architectures [9].

Humanoid robot navigation is described in [10] by using a

Footnotes: 8-point Times New Roman font;
Manuscript received July 1, 2012; revised August1, 2012; accepted September 1, 2012.
Copyright credit, project number, corresponding author, etc.

supervised RL approach combined with Gaussian distributed state activation. The Robot successfully performed a backward docking movement used for autonomous recharging. In [11], a neural network reinforcement learning method was studied for visual control of a robot manipulator. A direct mapping from the image space to the actuator command was developed using two different RL algorithms, Q-learning and SARSA, combined with neural networks. A database of representative learning samples was also employed to speed up the convergence of the algorithms. This hybrid system could provide the high accuracy of a manipulator positioning in a situation of low-resolution images.

III. NEURAL NETWORK-BASED REINFORCEMENT LEARNING METHODS

A. *Model-Free Reinforcement Learning*

Reinforcement learning can be formulated as a Markov Decision Process (MDP). Model-based RL algorithms can be used if we know the state transition function $T\left(s,a,s'\right)$. In contrast, we focus more on model-free RL that the environment model T is not known in advance, especially in our work of autonomous navigation tasks.

In this paper, we focus on robot control problems. A mobile robot acts in a stochastic environment by sequentially choosing actions over time steps to maximize a cumulative reward. We model the problem as a Markov Decision Process: a state-space S, an action space A, a transition dynamics distribution $P\left(s_{t+1}|s_t,a_t\right)$ satisfying the Markov property $P\left(s_{t+1}|s_1,a_1,s_2,a_2,...,s_t,a_t\right)=P\left(s_{t+1}|s_t,a_t\right)$, any trajectory $s_1,a_1,s_2,a_2,...,s_T,a_T$ in state-action space, and a reward function $r:S\times A\to R$. A stochastic policy $\pi\left(s_t,a_t\right)=P\left(a_t|s_t\right)$ is used to select actions and produce a trajectory of states, actions, and rewards $S\times A\times R$.

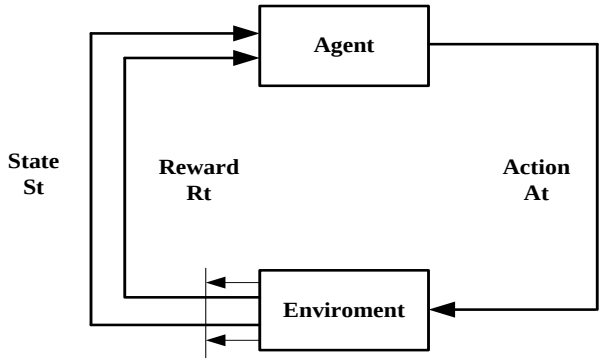

Figure 1: The schematic diagram of the reinforcement learning model

We study model-free RL methods that the Robot drives an optimal policy without explicitly learning the environment's model. More specifically, we focus on Q-learning algorithms, one of the most major model-free reinforcement learning algorithms.

B. *Q-Learning*

Q-Learning algorithm is an important off-policy model-free reinforcement learning algorithm for temporal difference learning. The update of state-action values in Q-learning is defined by

$$Q\left(s_t,a_t\right):=Q\left(s_t,a_t\right)+\alpha\left[r_{t+1}+\gamma\max_a Q\left(s_{t+1},a_t\right)-Q\left(s_t,a_t\right)\right] (1)$$

Where α is the learning rate, with a value between 0 and 1. If the learning rate is set to 0, the Q-values are never updated. If this value is set to a value near 1, learning can occur quickly, γ is the discount factor, and models the importance of future rewards worth less than immediate rewards.

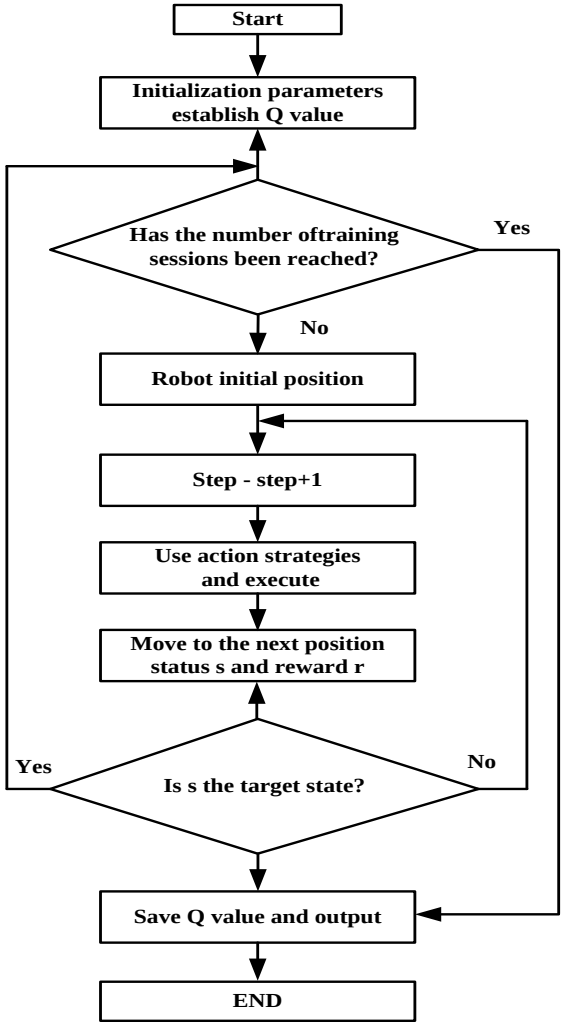

Figure 2: Schematic diagram of Q-Learning path planning method

C. *Neural Network-based Q-Learning*

This paper introduces a new model-free reinforcement learning method by integrating a neural network to Q-learning. This hybrid method takes advantage

of the strengths of both Q-learning and neural networks. Because the reinforcement learning can be modeled as an MDP, we need first to define the state space *S* and action space *A*.

a) State action space

Six discrete robot actions define the action space A. Among them, five are basic moving actions: move forward (F), turn left at 30° (L30), turn left at 60° (L60), turn right at 30° (R30), turn right at 60° (R60) and one emergency action: move backward (B). A turn at 90° is not defined because a sharp turn will bring great danger to a real vehicle. In sum,

$$A = \{F, L30, L60, R30, R60, B\} \qquad (2)$$

The state-space S is composed of a very large but finite number of states, defined by four groups of features, and is expressed in:

$$S = [D_t \quad V_t \quad I_t \quad U_t]^T \qquad (3)$$

Where $D_t \in R^{7\times1}$ are sensory degrees of danger? It $U_t \in R^{7\times1}$ is a vector to indicate if a wall is detected (U = 1) or an object obstacle (U = 0) for the seven sensors in D_t. $V_t \in \{1, 2, ..., 9\}$ is the target region and $D_t \in \{0,1\}$ is the indicator that determines if the Robot has detected the target.

b) Reward function

The reward function measures the immediate feedback for the action taken at a given state. It evaluates how good or how bad the performed action is. Before giving the reward function, one environment state at each time instant is classified into five state properties: Safe State (SS), Cozy State (CS), Dangerous State (DS), Winning State (WS), and Failure State (FS). The Reward Function can be defined as Table 1.

TABLE 1: REWARD FUNCTION

State Transition	Extra Criteria	r
Other states Winning State		1
Safe State → Cozy State		0
Cozy State → Safe State		0
Dangerous State → Cozy State		0.6
Cozy State → Dangerous State		0
Dangerous State → Failure State		-1
Dangerous State → Dangerous State (approaching obstacles)	$d_{warn} \le d^t_{\min} \le d^{t-1}_{\min} - 2$	-0.3
	$d^t_{\min} = d^{t-1}_{\min} - 1 \le d_{warn}$	-0.3
	$d^t_{\min} = d^{t-1}_{\min} - 2 \le d_{warn}$	-0.6
	$d^t_{\min} \le d^{t-1}_{\min} - 3$	-1
Nonsafe State → Nonsafe State (evading obstacles)	$d_{warn} \le d^t_{\min}$	0.7
	$d^t_{\min} \ge d_{warn}$	0.3

Component d_{r-o} and d_{r-t} define the distances between the Robot and obstacles, and the target, component d_{bou} defines the boundary distance (sensor detection range) of SS and other states, the component d_{col} defines the radius of the collision region around the obstacle, component d_{win} defines the radius of the winning region around the target, component d_{cozy} defines the cozy distance, component d_{warn} is a warning distance that the Robot is approaching too close to an obstacle, component $d^{t-1}_{\min}$ and components $d^t_{\min}$ are the minimum distances between the Robot and the surrounding obstacles.

c) State- action value iteration

The Q-value function expresses the mapping policy from the perceived state of the environment to the executing action. One Q-value $Q(s_t, a_t)$ corresponds with one specific state and one action in this state. In our method, we propose predicting all state Q-values using a three-layer neural network, as shown in Fig. 3.

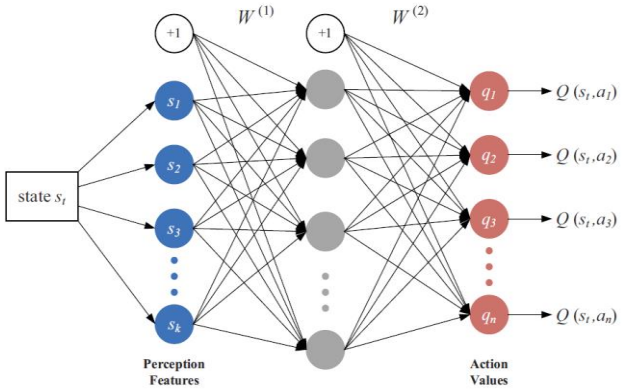

Figure 3: A three-layer neural network architecture

The action value iteration is realized by updating the neural network by means of its weights. Unlike the classic neural network, the neural network in the reinforcement learning does not have label outputs. Q-learning is a process of value iteration, and the optimal value after each iteration serves as the target value for neural network training. The update rule is

$$Q_{k+1}(s_t, a_t) := Q_k(s_t, a_t) + \alpha\left[r_t + \gamma \max_a Q_k(s_{t+1}, a_t) - Q_k(s_t, a_t)\right] \qquad (4)$$

Where a component $Q_k(s_t, a_t)$ is the Q-value in the k^{th} iteration. We also employ the popular stochastic gradient descent (SGD) to train the neural network online. The goal is to minimize the cross-entropy cost function.

d) Robot Navigation Using Neural Network-based Q-learning

After training the Robot via the integrated neural network, the resulting policy is still stochastic but near-deterministic used by the Robot for future navigation tasks in various environments. The Robot starts its navigation through the environment by finding its current state. If it is a Safe State, the Robot does not need to follow the policy but changes its orientation towards the target and moves one step forward. It continues moving until entering a Non-Safe region where the Robot needs to adopt the trained control policy. The Robot uses the Neural Network to generate all possible state-action Q-values. The Robot greedily takes the action that has the biggest Q-value. After that, the Robot finds its new state and repeats action selection until the Robot reaches its goal or collides with an obstacle.

IV. EXPERIMENTAL RESULTS

Our objective is to demonstrate the preliminary results obtained by the offline learning phase in Gazebo-based simulation [7] and the OpenAI Gym extension [12]. We define a navigation mission as the problem of visiting a set of waypoints. The simulation environment in Gazebo is shown in Fig. 4.

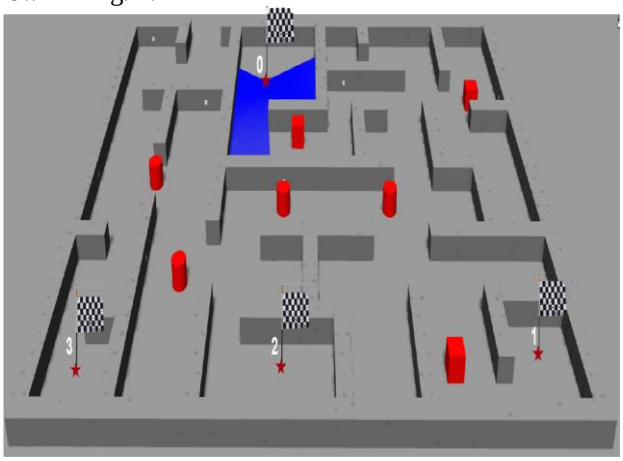

Figure 4: Example of a simulation environment in Gazebo with seven dynamic obstacles: single box or cylinder in red models dynamic obstacles, blue area represent laser scan. The flags represent the example of navigation waypoints

The dynamic obstacles can be randomly generated in the environment. A learning episode of Q-learning is a completed mission. We deployed 400 learning episodes in simulation. The first 200 episodes are to visit a set of waypoints with the total optimal trajectory length of 204m and the last 200ones for a different set of waypoints with the total path length of 166 m.

Fig.5 depicts the total reward obtained over 400 learning episodes. Despite many fluctuations due to the variable complexity of environments and the efficiency of existing navigation algorithms, the trend line also indicates the total reward growth of the total reward during the learning phase.

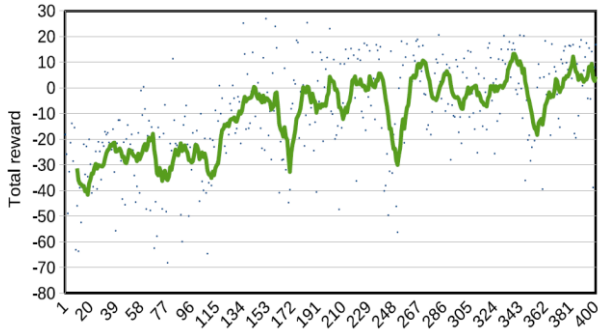

Figure 5: Total reward of the learning phase of 400 episodes with the trend line in green

V. CONCLUSIONS

In this paper, we presented the robot self-learning strategy without prior experience under explicit feedback. We explored the mobile robot navigation problem 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. We designed a new reward expression and introduced the neural network architecture to store and train the large-scale Q-values and generalize the learning performance to large-scale state and action spaces. Experiments are conducted on autonomous navigation tasks for mobile robots. The simulation results show the stability and feasibility of the hybrid method.

ACKNOWLEDGMENT

This study was supported by the Faculty of Electrical Engineering, University of Economics - Technology for Industries, Viet Nam; http://www.uneti.edu.vn/.

REFERENCES

[1] Ghosh, S., Panigrahi, P. K., and Parhi, D. R, Analysis of FPA and BA meta-heuristic controllers for optimal path planning of mobile robot in the cluttered environment, IET Sci. Measure. Technol. 11,(2017) 817- 828. doi: 10.1049/iet-smt.2016.0273.

[2] Orozco-Rosas, U., Montiel, O., and Sepúlveda, R, Mobile robot path planning using membrane evolutionary artificial potential field, Appl. Soft Comp. 77 (2019) 236–251. doi: 10.1016/j.asoc.2019.01.036.

[3] Han, J., and Seo, Y., Mobile robot path planning with surrounding point set and path improvement, Appl. Soft Comp. 57, (2017)35–47. doi: 10.1016/j.asoc.2017.03.035.

[4] Li, G., and Chou, W., Path planning for mobile robot using self-adaptive learning particle swarm optimization, Sci. China Inform. Sci. 61, 052204–052213. doi: 10.1007/s11432-016-9115-2.

[5] Zhao, Y., Zheng, Z., and Liu, Y. (2018), Survey on computational intelligence-based UAV path planning, Knowledge-Based Syst. 158, (2018) 54–64. doi: 10.1016/j.knosys.2018.05.033.

[6] Bakdi, A., Hentout, A., Boutami, H., Maoudj, A., Hachour, O., and Bouzouia, B., Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control, Robot. Autonomous Syst. 89,((2017) 95–109. doi:10.1016/j.robot.2016.12.008.

[7] I. Zamora, N. G. Lopez, V. M. Vilches, and A. H. Cordero, Extending the open-air gym for robotics: A toolkit for reinforcement learning using ros and gazebo, arXiv preprint

arXiv:1608.05742, 2016.

[8] A. D. Pambudi, T. Agustinah, and R. Effendi, Reinforcement Point and Fuzzy Input Design of Fuzzy Q-Learning for Mobile Robot Navigation System, 2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT), 2019.

[9] X. Ruan, D. Ren, X. Zhu, and J. Huang, Mobile Robot Navigation based on Deep Reinforcement Learning, 2019 Chinese Control And Decision Conference (CCDC), 2019.

[10] N. Navarro-Guerrero, C. Weber, P. Schroeter, and S. Wermter, Real-world reinforcement learning for an autonomous humanoid robot, Robotics and Autonomous Systems, 2012.

[11] Z. Miljković, M. Mitić, M. Lazarević, and B. Babić, Neural network reinforcement learning for visual control of robot manipulators, Expert Systems with Applications, 40 (2013) 1721–1736.

[12] G. Brockman, V. Cheung, L. Pettersson, J.Schneider, J. Schulman, J. Tang, and W. Zaremba, Openai gym, arXiv preprint in arXiv:1606.01540, 2016.

[13] Roan Van Hoa, L. K. Lai, Le Thi Hoan, Mobile Robot Navigation Using Deep Reinforcement Learning in Unknown Environments, SSRG International Journal of Electrical and Electronics Engineering (SSRG-IJEEE), , (2020) 7(8) 15-20.

[14] Pham Ngoc Sam, Tran Duc Chuyen, Research and Designing a Positioning System, Timeline Chemical Mapping for Multi-Direction Mobile Robot, SSRG International Journal of Electronics and Communication Engineering, 7 (11), (2020) 7-12.