SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348 – 8549 /doi:10.14445/23488549/IJECE-V8I1P103

Volume 8 Issue 1, 12-17, January 2021
© 2021 Seventh Sense Research Group®

Research and Application Deep Q-Network Algorithm for Automatic Navigation for Omnidirectional Mobile Robots

Tran Thi Huong[#], Pham Thi Thu Ha[#], Pham Van Bang[#]

[#]Faculty of Electronic Engineering Technology, University of Economics - Technology for Industries, Viet Nam

Abstract *- In this paper, the setting up of the Deep Q-Network (DQN) algorithm in the mobile robot's Gazebo simulation environment has been calculated, designed, and controlled. Building experiments aim to make the robot model learn the best actions to control and navigate in an environment with many obstacles. For the mobile robot to move in an obstacle environment, the robot will then automatically control to avoid these obstacles. After that, the robot can remain within a specific limit, the more rewards it accumulates. The authors have performed various tests with many parameters and demonstrated the performance curves on the simulation. The research results will be the basis for designing and establishing control algorithms for current and future mobile robots for application in programming techniques and automation control in industrial control.***

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

I. INTRODUCTION

The automatic navigation of a mobile robot can be divided into three basic problems: information perception, behavioral decision making, and control manipulation. Setting up a route tracking plan is the basis of the mobile robot navigation and control problem [1] - [3]. Planning the entire mobile robot control environment is to find the way from the current location to the target location. The path must be as short as possible, the smoothness of the road should meet the mobile robot's dynamics, and the path must be safe, no collisions occur [2]. Based on specific algorithms and strategies, the control planning algorithms for robots can be divided into four categories: first, pattern matching, second artificial lead field, construction setting mapping, and fourth artificial intelligence [3, 4]. Each type of motion map planning algorithm for a robot has an optimal application scenario and eliminates the limitations of algorithm control errors. The current map planning of a mobile robot is mainly based on its surroundings. In addition to the limitations of traditional route planning, robots cannot complete their learning and judgment in complex environments, which is an issue in developing research in the field [9, 16]. Therefore, it is especially important to develop a method of mapping the robot so that the omnidirectional mobile robot is less dependent on the environment and can quickly adapt to its surroundings.

Creating robots with human-like abilities to execute specific skills smoothly and naturally is one of the mobile robot's key goals for performing various real-life tasks such as a delivery row, search or rescue, security surveillance in the army. Such tasks require varying degrees of autonomy in navigation in response to various motivational factors such as changing environmental conditions. One of the most popular approaches is to train mobile robots through numerous expert demos. These demonstrations provide the robot with previous experiences from the examples, defined as a series of state-action pairs recorded during the teacher's demonstration of the desired behavior boots. However, in some cases, such as exploring dangerous environments, mobile robots have no experience or require pre-demonstration [6, 8, 9, 10, 11, 13].

Although the DQN algorithm and its extensions have enjoyed great success in the console environment of reward-seeking games, at the beginning of the teaching process, the current DQN algorithms required millions of policy-based random patterns to teach robots any optimal policy, and the right variety of patterns will lead to slower training speeds. However, in many cases, the teaching sampling can be different or time-consuming. Therefore, some researchers try to represent a real-world basis by using synthetic models [1, 6, 8, 25] to improve sample performance. When the general model is well trained, the DQN algorithm can be trained without interacting with the actual environment. In [1, 2, 4, 7, 18, 20], teaching methods for robots give optimal policies for when to use teaching patterns.

In this paper, the authors present a DQN based self-learning algorithm in the cases of an unknown environment. Experiments are conducted on the autonomous navigation tasks for mobile robots. More specifically, we introduce a neural network structure to generalize and approximate the value of all the states of DQN algorithms. The experiments are conducted on the Gazebo simulator using its open source extension Gym to perform the autonomous navigation tasks for omnidirectional mobile robots.

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

II. BUILDING CONTROL MODELS FOR OMNIDIRECTIONAL MOBILE ROBOTS

A. The controller design for the robot

The omnidirectional mobile robot model is set up on the same basis as the four-wheeled Asimo robot with dimensions: height 65cm, width 42cm square. The robot mission is for the mobile omnidirectional robot. Figure 1 shows the control system block diagram of an omnidirectional mobile robot.

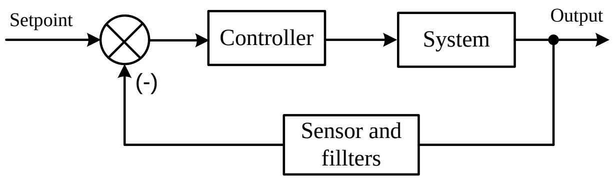


Fig 1: Control system block diagram

The input signal is a setpoint signal. The robot's Smart Sensor measures the frame's winding angle, pitch, and tilt angle every second and sends them to the microprocessor to command control based on algorithms. Intelligent control such as adaptive control, sustainable control, neural network control and artificial intelligence, etc. Then the controller will calculate the optimal action-value to respond to the unit controlled by a closed control loop, and then it will make the control of the robot stable; figure 1 shows the robot's control system, [1, 8].

B. Reinforcement learning methods as controllers

Previously, we worked on traditional controllers like PID, Fuzzy PD, PD, PI & LQR algorithm. The biggest problem with those methods is that they need to be tuned manually. So, reaching optimal values of controllers depends on many trials and errors. Many times optimum values aren’t achieved at all. The biggest benefit of reinforcement learning algorithms as controllers is that it tunes itself to reach the optimum values. The following two sections discuss the Q Learning algorithm and Deep Q Network algorithm.

With Q learning algorithm: According to document [3], “this algorithm gives actors the ability to learn how to act optimally in Markovian domains by experiencing the consequences of actions without asking them to build control environment maps for an omnidirectional mobile robot.” In the Markovian domain, the function Q - algorithmically generated model - computes the expected utility for a given finite state s and any possible finite action a. for agents - which are omnidirectional mobile robots, in this case - allowing to choose the optimal action with the highest value of Q (s, a), this action selection rule is also called priority policy. Initially, the function values Q (s, a) are assumed to be 0. Then through each training step, the values are updated according to the following equation, [1, 3]:

$$Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right) + \alpha\left[R_t + \right.$$
$$\left. + \gamma \max_{a\in A} Q\left(s_{t+1}, a_t\right) - Q\left(s_t, a_t\right)\right] \quad (1)$$

$$Q\left(s_t, a_t\right) \leftarrow R_t + \gamma . \max_{a\in A} Q\left(s_{t+1}, a\right) \quad (2)$$

In the DQN algorithm, when training the neural network, there exists an arithmetic coefficient (α) in the backpropagation process, so it is reasonable to omit the learning coefficient (α) in formula (2). By doing this, the calculation of updating values also becomes simpler then we have:

$$Q\left(s_t, a_t\right) \leftarrow R_t + \alpha .(r + max Q(s_{t+1}, a)) \quad (3)$$

The Q matrix had 20 columns, each column representing a state and ten rows representing every action. Initially, the Q-values were assumed to be 0, and some random actions were specified for every state in the policy π . We trained or 1500 episodes, each episode having 2000 iterations. At the beginning of each episode, the simulation refreshed. Whenever the robot’s state exceeded the limit, it was penalized by assigning a reward to -100. The Q Table is updated at each step according to equation (3). This algorithm shows the full algorithm, [23 - 25].

III. RESEARCH AND APPLICATION OF DQN ALGORITHM

A. DQN algorithm learning method

DQN inherits all properties of Q-learning. It is a model in the form of a free model, learns online, and belongs to off-policy algorithms.

As mentioned in document [1, 11, 17]. The Q-learning and State Action Reward State Action SARSA (State Action Reward State Action) algorithms have memory problems when storing the evaluation function as a two-dimensional array Q(*s*, *a*). When the state space and the action space are very large, about hundreds or thousands, this storage will have memory problems, not to mention that the computational cost of updating the value will increase exponentially.

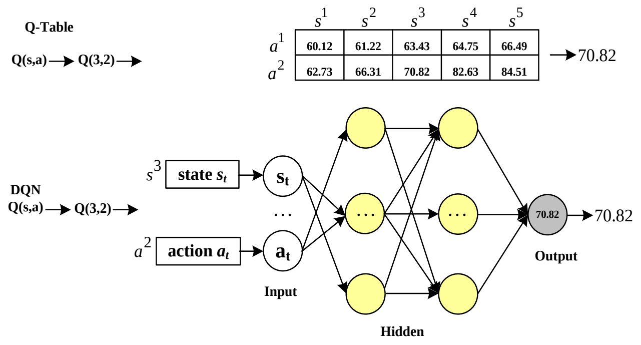


Figure 2: The from Q-Table to DQN algorithm

In addition, the Q-learning algorithm still has another major weakness: the inability to estimate values for unknown states $s_i \notin S\{s_1, s_2, ...s_T\}$, thus, inability to predict, leading to lack of generalization. To solve this problem, the DQN algorithm can remove the two-dimensional Q-Table array instead of building a neural network to approximate this Q-Table algorithm table shown in figure 2 below the illustrative example.

The same Q – learning algorithm, DQN training process is also based on temporary differential method, DQN Agent updates network parameter θ of Q rating (S, A) at each step of the network training, to Execute action a, receiving the new algorithm R, then it will significantly improve the performance of the control model for the mobile robot when using the control programming using the DQN algorithm.

B. The robot navigation using DQN

The DQN algorithm was originally introduced as an AI agent that can play videogames at a level that can rival human players [3, 14]. The DQN algorithm has been able to complete different games with the same algorithm. In videogames, each new screen is a new state where the agent can take action. Since there are so many possible states and actions, it is impossible to explore them or use conventional algorithms to solve the game.

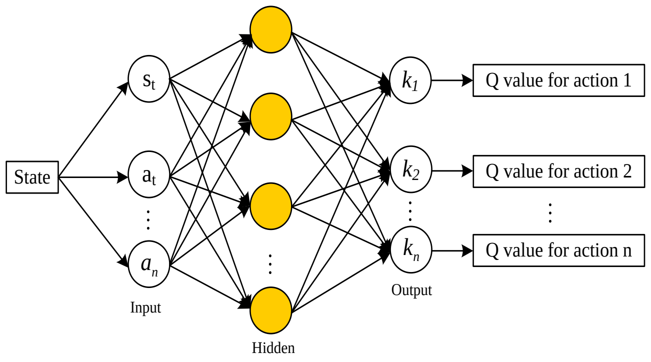


Figure 3: The schematic diagram of the reinforcement learning model

A deep Q - network algorithm starts by exploring a game and gradually learning its mechanics; the more the agent plays this game, the more it learns and can achieve higher scores. In this work, the DQN will learn the connection between the change of geometrical properties and their effect on final results through simulations and then use that knowledge to design structures that produce the optical responses that we desire. First, the environment is set up; this includes the initial structure design and the simulation environment. Second, the agent's actions to change the structure are decided, and finally, the reward system is defined. The DQN algorithm that connects all these parts is shown in figure 3.

The decision to take in a given state is decided by an updated neural network based on what it has learned. To improve the performance of the DQN, an auxiliary model is used alongside. This network is used to select the agent's action, while the main DQN network predicts the Q-value of the state-action pair. This prevents the overestimation that is a problem in general DQN. At each iteration, two models are trained, and the weights of the target model are gained from the combination of the main model weights and the target model weights. This method helped the overestimation caused by using just one model. The auxiliary network is updated periodically with the parameters of the DQN algorithm.

The implementation of the DQN algorithm on our Robot model: is similar to the Q-Learning algorithm. However, there are some exceptions. At first, a model was initialized instead of Initializing the Q matrix. Instead of choosing the action based on policy π, Q values were calculated according to the model in the greedy policy. At the end of every episode, the model was trained using random mini-batches of experience. At first, an architecture with two hidden Relu layers of 20 units was selected, whereas the last layer was a Linear Dense layer with ten units. With the γ of 0.999 and α of (0.65, 0.7, 0.8). The DQN algorithm deployed on the robot model shows that to evaluate the algorithm's quality and efficiency when the estimation function is used in the whole process, the objective function helps to create a separation for improvement. Stability improvement. Furthermore, the neural network training process is kept fixed for computation and is updated gradually, and the network training process is somewhat similar to supervised learning.

The performance of the deployed DQN algorithm is very satisfactory. One of the main reasons this algorithm offers omnidirectional robot control is to develop an algorithm that can be used to control autonomous robots and industrial robots in industrial factories. , in industrial environments such as the 5S environment, Japan's 6S environment applies to all factories. Compared to other algorithms, the comparison of DQN algorithms shows the optimization of RL, Q-learning, etc., was also successful. Several tests for process control have been given and clearly show how well the DQN algorithms work under different circumstances. Therefore, to build a complete DQN algorithm, it is always necessary to meet the needs: from selecting a control action to a multicast mobile robot, executing the action; receive rewards, store and sample to train the algorithm, then go to the calculation of the target function, thereby updating the model parameters by minimizing the loss function on all selected samples, teach, followed by choosing a method to update neural network parameters and target function, and finally updating the control coefficients with uncertainty, [1, 3, 10, 15, 21].

Algorithm: DQN Algorithm as applied in the system

```
Initialize Robot;
Initialize model $M$;
Initialize Penalty Reward $pen$;
for number of episodes do
    Reset simulation ;
    Wait for 1 second ;
    Pause simulation ;
    Read the pitch angle $\phi$ of the robot ;
    $state \leftarrow \phi$ ;
    start simulation ;
    for number of iterations do
        Generate a random number $rand$;
        if $rand \leq \delta$ then
            take random action ;
        end
        else
            $Q \leftarrow M(state)$;
            $action \leftarrow action\ formax(Q)$;
        end
        $state_{new} \leftarrow \phi$;
        Pause simulation;
        if absolute value of $state_{new} \geq limit$ then
            if $reward_{total} \leq Target$ then
                $reward \leftarrow pen$;
                $experience \leftarrow$
                 $(state, reward, action, state_{new})$;
                Add Experience to Memory;
            end
            Break ;
            else
                Print Passed ;
                Break ;
            end
        end
        else
            $reward \leftarrow 1$;
            $experience \leftarrow$
              $(state, reward, action, state_{new})$;
            Add Experience to Memory;
            $state \leftarrow state_{new}$
        end
    end
    Take radom minibatch of Experience;
    if $reward == pen$ then
        $Q_{pred} \leftarrow reward$;
    end
    else
        $Q_{pred} \leftarrow$
        $reward + \gamma max(Q(state_{new}, action))$
    end
    ;
    Train the model according to loss
    $abs(Q_{pred}(state, action) - Q_{pred}(state, action))$;
end
```

IV. EXPERIMENTAL RESULTS

In this section, some simulations are conducted based on the powerful simulation environment tool in Gazebo. Figure 5 shows that the map built on Gazebo is a map made with strict walls and a mobile robot that can be manipulated to move around those environments to get the necessary data. Used to construct the map. The red lines are laser scan signals generated from pine sensors, smart cameras, and the robot's current position is updated (the robot is denoted in green) using geometric measurements. Primarily a visualization tool that can provide live updates of maps generated from the SLAM algorithm. Furthermore, a vehicle's motion trajectory in the map can also be visualized and generated in an obstacle-containing environment, as shown in figure 4.

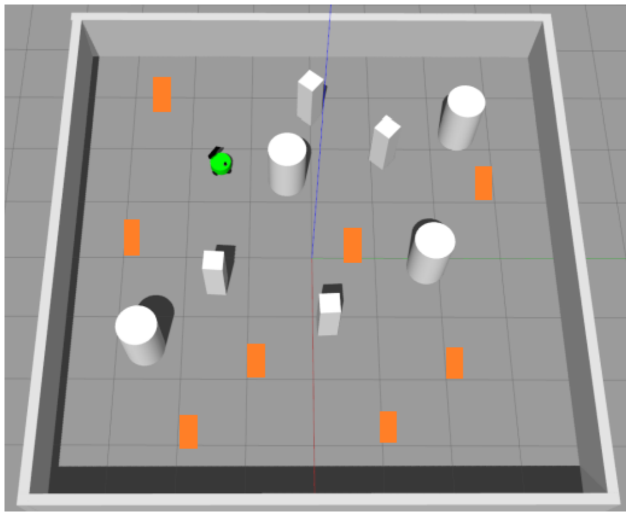
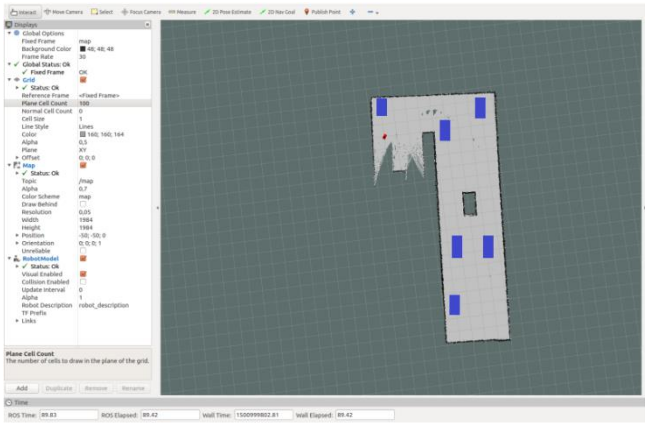


Figure 4: The example of a simulation environment in Gazebo with contains a robot (green)

The general idea behind the positioning node is to save the trajectory data of the vehicle. The vehicle's position is obtained by looking up a transform from the fixed world frame to the base-link frame, which represents the actual position of the vehicle. The estimated position is generated by the SLAM algorithm. The topic was introduced for testing purposes to compare the vehicle's actual position with an estimated position to validate each SLAM algorithm. The data is saved to the text file is the x and y coordinates and a time stamp. The values are stored into a vector with a frequency of 10 Hz; basically, the same as the frequency of the/car-position topic referred to in the controller.

In figure 5, to create an operating environment to control and navigate the robot, the authors map intending to create environments containing obstacles, then programming the robot. Move around and avoid obstacles (the robot in figure 5 is the red color spot) with devices such as cameras and smart sensors. Here is primarily a visualization tool that can provide live updates of maps generated from the SLAM algorithm. Furthermore, the vehicle's trajectory in the map can also be visualized.



Global Options	
Fixed Frame	map
Background Color	48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
✓ Status: Ok	
Reference Frame	<Fixed Frame>
Plane Cell Count	100
Normal Cell Count	0
Cell Size	1
Line Style	Lines
Color	160; 160; 164
Alpha	0,5
Plane	XY
Offset	0; 0; 0
Map	✓
✓ Status: Ok	
Topic	/map
Alpha	0,7
Color Scheme	map
Draw Behind	☐
Resolution	0,05
Width	1984
Height	1984
Position	-50; -50; 0
Orientation	0; 0; 0; 1
Unreliable	☐
RobotModel	✓
✓ Status: Ok	
Visual Enabled	✓
Collision Enabled	☐
Update Interval	0
Alpha	1
Robot Description	robot_description
TF Prefix	
Links	

Plane Cell Count
The number of cells to draw in the plane of the grid.

ROS Time: 89.83 ROS Elapsed: 89.42 Wall Time: 1500999802.81 Wall Elapsed: 89.42

Figure 5: The main window shows the map generated by a SLAM algorithm

The dynamic obstacles can be randomly generated in the environment. A learning episode of the DQN algorithm is a completed mission. The authors deployed 4500 learning episodes in simulation. The first 2250 episodes are to visit a set of waypoints with the total optimal trajectory length of 255m and the last 2250 ones for a different set of waypoints with the total path length of 195m. Figure 6 depicts the total reward obtained over 4500 learning episodes. Despite many fluctuations due to the variable complexity of environments and the efficiencyof existing navigation algorithms, the trend line also indicates the total reward growth of the total reward during the learning phase [12, 16, 18].

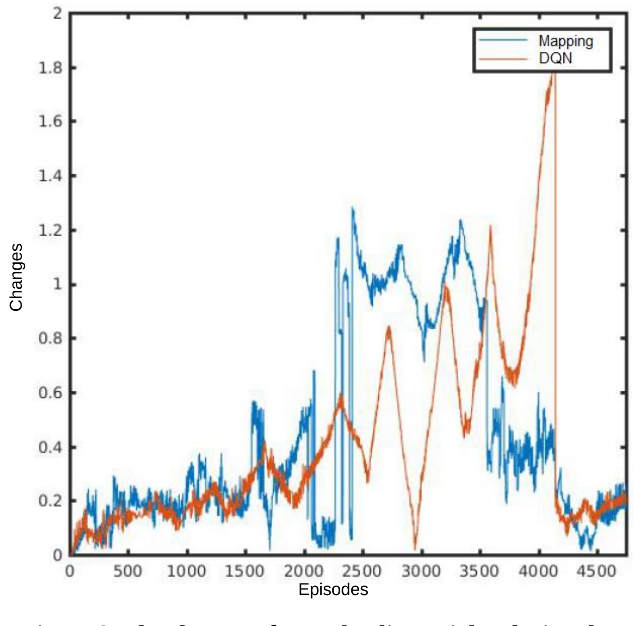


Figure 6: The changes of neural policy weights during the learning process of 4500 episodes

The neural network DQN can perceive: the environment and perform feature extraction to realize the fitting from the environment to the state action function. This has been mentioned in the literature. In [7] proposed an adaptive DQN algorithm strategy and applied it to text recognition. These results showed that the DQN algorithm is significantly better than other algorithms, which also indicated the DQN algorithm's advantages in image recognition [2].

Compared with the deep learning algorithm Q-learning, the DQN algorithm is better than Q-learning in terms of value accuracy and strategy, consistent with previous reports [3]. The hierarchical reinforcement learning technology is utilized to achieve the mapping from state to action and meets mobile robots' mobile needs. The data have also proven that the robot path planning method based on deep reinforcement learning is an effective end-to-end mobile robot path planning method, which has also been confirmed in a study by [17]. The above results illustrate the feasibility of the proposed method in the path planning of mobile robots.

V. CONCLUSIONS

In this paper, we presented the robot self-learning strategy without prior experience under explicit feedback. The authors explored the mobile robot navigation problem by combining reinforcement learning and neural network. The DQN algorithm is applied to enhance the self-learning ability of a mobile robot through trial - and - error interactions with an unknown environment. Authors designed 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 the autonomous navigation tasks for mobile robots. The simulation results show that the stability and applicability of the method using the DQN algorithm are very effective; these new studies can be completely applied to the control and navigation for mobile robots in industrial factories in Vietnam as well as in the world, furthermore, than previous studies [4, 7, 14, 16], the research result of the paper is better. Then DQN algorithm is simpler, making the navigation of robot operation monitored through visual tools, like cameras and intelligent sensors for precise control over obstacles.

ACKNOWLEDGMENT

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

REFERENCES

[1] Nguyen Thanh Tuan, Base Deep learning, The Legrand Orange Book. Version 2, last update, August 2020.

[2] Charu C. Aggarwal, Neural Networks and Deep Learning, Springer International Publishing AG, part of Springer Nature, 2018.

[3] Vu Thi Thuy Nga, Ong Xuan Loc, Trinh Hai Nam, Enhanced learning in automatic control with Matlab Simulink, Hanoi Polytechnic Publishing House, 2020.

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

[5] Han, J., and Seo, Y., Mobile robot path planning with surrounding point set and path improvement, Appl. Soft Comp. 57, 35–47.

[6] V. Matt and N. Aran, Deep reinforcement learning approach to autonomous driving, ed: arXiv, 2017.

[7] 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, 2018.

[8] Andrea Bacciotti**,** Stability and Control of Linear Systems**,** Publishing Ltd; Springer Nature Switzerland AG, 2019.

[9] 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.

[10] 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.

[11] Gu, S.; Holly, E.; Lillicrap, T.; Levine, S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Marina Bay Sands, Singapore, 3389–3396, 29 (2017).

[12] 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.

[13] Do Quang Hiep, Ngo Manh Tien, Nguyen Manh Cuong, Pham Tien Dung, Tran Van Manh, Nguyen Tien Kiem, Nguyen Duc Duy, An Approach to Design Navigation System for Omnidirectional Mobile Robot Based on ROS, (IJMERR); 11(9) (2020) 1502-1508.

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

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

[16] Saleem, Y.; Yau, K.L.A.; Mohamad, H.; Ramli, N.; Rehmani, M.H.; Ni, Q. Clustering and Reinforcement Learning-Based Routing for Cognitive Radio Networks. IEEE Wirel. Commun. 2017.

[17] 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.

[18] Pham Ngoc Sam, Tran Duc Chuyen, Research and Designing a Positioning System, Timeline Chemical Mapping for Multi-Direction Mobile Robot, 7(11) (2020) 7-12, Publishing by SSRG - IJECE Journal.

[19] Shota Ohnishi, Eiji Uchibe, Yotaro Yamaguchi, Kosuke Nakanishi, Yuji Yasui, and Shin Ishii, constrained Deep Q-Learning Gradually Approaching Ordinary Q-Learning, 13 (2019) 7-12, Publishing by Frontiers in Neurorobotics Journal, December.

[20] 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), 7(8) (2020) 15-20.

[21] Song Han, Luo Li and Xinbin Li Deep Q-Network-Based Cooperative Transmission Joint Strategy Optimization Algorithm for Energy Harvesting-Powered Underwater Acoustic Sensor Networks, Sensors (Basel) 2020 Nov; 20(22): 6519. Published online 2020 November 14.

[22] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. Nature 2015.

[23] Fu X, Du J, Guo Y, Liu M, Dong T, et al. A machine learning framework for stock selection. arXiv, cited 2018 August 2018.

[24] Ganggang Guo, Yulei Rao, Feida Zhu, Fang Xu, Innovative deep matching algorithm for stock portfolio selection using deep stock profiles, PLoS One. 15(11) (2020), Published online November.

[25] Wu, Y.; Tan, H.; Peng, J.; Zhang, H.; He, H. Deep reinforcement learning of energy management with continuous control strategy and traffic information for a series-parallel plug-in hybrid electric bus. Appl. 247, (2019) 454-466, Energy.