SSRG International Journal of Electrical and Electronics Engineering Volume 8 Issue 4, 19-24, April 2021
ISSN: 2348 – 8379 / https://doi.org/10.14445/23488379/IJEEE-V8I4P104 © 2021 Seventh Sense Research Group®

Building Intelligent Navigation System for Mobile Robots Based on the SARSA Algorithm

Nguyen Thi Thu Huong

Department of Electrical Engineering, Faculty of Control Engineering, Le Quy Don Technical University

Received Date: 07 March 2021
Revised Date: 14April 2021
Accepted Date: 26 April 2021

Abstract - *This article presents the construction of an intelligent automatic navigation system for mobile robots in a flat environment with defined and unknown obstacles. The studies using programming tools are the operating system for mobile robots (Robot Operating System - ROS). From updated information on maps, operating environment, robot control position, and obstacles (Simultaneous Localization and Mapping (SLAM)) to calculate the motion trajectory of the mobile robot. The navigation system calculates the global and local trajectory for the robot based on the application of SARSA algorithm. The results of simulation studies in the Gazebo environment and the experimental run on the real Turtlebot3 mobile robot showed the practical efficiency of automatic navigation for this mobile robot.*

Keywords - *Artificial intelligence, Mobile robot, Robotic, Reinforcement learning, SARSA algorithm.*

I. INTRODUCTION

The term robotics and robot control are nowadays becoming common and are step by step closely linked to people's daily lives such as service robots (robot vacuuming, cleaning the house, cooking, preparing, etc.), industrial robots (robots in production lines), medical robots, robots in the military field, etc. Robot engineering is a multi-disciplinary field including mechanics, electricity, electronics, and automatic control and information technology. Each field, when researched, plays an important role in the process of researching, designing, and manufacturing robots [1] - [5]. The action environment to control a robot is often represented as a finite state (Markop Decision Process - MDP), and the reinforcement learning algorithms for this context have much to do with the Dynamic planning techniques. The transition probabilities and the gain probabilities in MDP are often random but static during the course of the robot control problem. Unlike supervised learning, in reinforcement learning, there are no correct input/output pairs, and near-optimal actions are not explicitly assessed as true or false. Furthermore, online performance is of interest here, which involves finding a balance between discovery (unmapped territory) and exploitation (existing knowledge). There are two commonly used methods to solve decision problems: searching in strategic space and searching in the space of value functions, also known as "strategic iteration" and "value iteration". . These two methods are characteristic reinforcement learning algorithms. Besides, in recent studies, scientists propose a combined method between the two above methods, which is the Actor-Critic learning method [3, 7, 10, 13, 16].

Previously, we worked on traditional Controllers like PID, Fuzzy PD, PD+I, PI, LQR, and many more classic controllers [1, 2, 14, 21, 22]. The biggest problem with those methods is that they need to be tuned manually. So, it will cause many errors, the system works inaccurately, making the quality of the controller not high. Therefore, it will cause many errors, the system works inaccurately, making the quality of the controller not high. The working system is not optimal, not good. Many times optimum values aren't achieved at all. The biggest benefit of reinforcement learning algorithms as controllers is that the model tunes itself to reach the optimum values such as Q-learning algorithm, Deep Q-Network algorithm, DDPG algorithm, SARSA (State Action Reward State Action) algorithm, etc. The SARSA algorithm is an algorithm that finds the optimal cumulative value for an action, which is very similar to the Q-learning algorithm. The obvious difference between them is that Q-learning belongs to an off-policy algorithm group while the SARSA algorithm belongs to an on-policy group [3, 8, 11].

With the SARSA algorithm, learning is done by a number of agents that only choose random actions to explore the environment in the first moves. Then selected actions in the next steps are all in compliance with the agreed policy. By the way, the learning agents will learn longer than other algorithms. However, it will be safer, better learning quality. For mobile robots, the automatic navigation in fixed and mobile obstacle environments, the use of the SARSA algorithm is appropriate, bringing many benefits in the intelligent control process for the robot [9, 24, 26].

II. THE CONTROL MODEL FOR MOBILE ROBOT

A robot is a complex mechanical system with many masses and many degrees of freedom. Each degree of freedom performs a movement and is controlled by an electric drive system. Furthermore, the robot is a controlled object containing many interrelated motors. To build the control system model for a robot, we consider a mobile robot, as shown in Figure 1. Two coordinate frames can be used to describe the movement of a mobile robot. One is the global coordinate frame (X, Y) defined in the world, and the other is the local coordinate frame (Vl and Vr) defined on the mobile robot. The angle between the two coordinate frames is denoted by θ. The robot's motion will be defined for the navigation stack. As the overall coordinate is chosen in Figure 1, it is clear that the robot's velocity contains three components: the linear velocity along the OX axis and the angular speed along the OY axis.

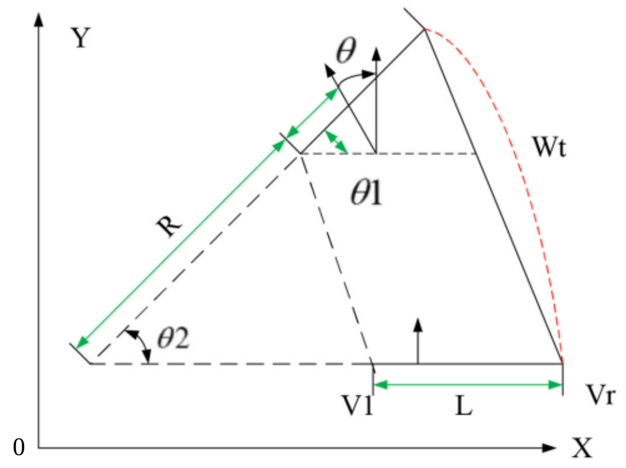

Fig. 1 The path planning motion model of mobile robots

The path planning task explored in this study relies on a two-wheeled mobile robot and a semi-front for arbitrary movement. This mobile robot can control the speed of its two rudders to achieve arbitrary motion trajectories such as linear motion, spin, or circle. Figure 1 shows the robot's posture at adjacent intervals. Based on the established kinetic model, this mobile robot can travel on a flat path and can avoid fixed and moving obstacles moving [1, 3, 9].

The world coordinate system pose of the mobile robot at time t is set to $W_t = [x_t, y_t, \theta_t]^T$ if the world coordinate pose of the mobile robot at a time with $(t + \Delta t)$ is $W_{t+\Delta t} = \left[x_{t+\Delta t}, y_{t+\Delta t}\right]^T$ the distance between the left and right driving wheels is L, the speeds of the left and right driving wheels are v_l and v_r, and the robot linear speed and angular speed are respectively v and ω, the speed v of the mobile robot in the ideal motion state is:

$$v = \frac{v_l + v_r}{2} \tag{1}$$

The angular velocity of the robot is:

$$\omega = \frac{v_l + v_r}{L} \tag{2}$$

The instantaneous curvature radius R is:

$$R = \frac{v}{\omega} \tag{3}$$

As shown in Figure 1, $\theta_1 = \theta_2 = \theta$, after Δt, the heading angle of the robot changes as follows:

$$\theta_{t+\Delta t} = \theta_t + \theta \tag{4}$$

The motion from position $W_t = [x_t, y_t, \theta_t]^T$ to $W_{t+\Delta t} = \left[x_{t+\Delta t}, y_{t+\Delta t}, \theta_{t+\Delta t}\right]^T$ can be regarded as a circular arc with radius R. If the arc is used to approximate the actual trajectory of the mobile robot, the geometric relationship should be:

$$\begin{bmatrix} x_{t+\Delta t} \\ y_{t+\Delta t} \\ \theta_{t+\Delta t} \end{bmatrix} = \begin{bmatrix} x_t + R(\sin(\theta_t+\theta) - \sin\theta_t) \\ y_t + R(\cos(\theta_t+\theta) - \cos\theta_t) \\ \theta_t + \theta \end{bmatrix}, \theta \neq 0 \tag{5}$$

Combining the above equations, the motion equation of the differential mobile robot can be obtained as:

$$\begin{bmatrix} x_{t+\Delta t} \\ y_{t+\Delta t} \\ \theta_{t+\Delta t} \end{bmatrix} = \begin{bmatrix} x_t + \frac{L(v_r+v_l)}{2(v_r-v_l)}(\sin(\theta_t+\theta) - \sin\theta_t) \\ y_t + \frac{L(v_r+v_l)}{2(v_r-v_l)}(\cos(\theta_t+\theta) - \cos\theta_t) \\ \theta_t + \theta \end{bmatrix}, \theta \neq 0 \tag{6}$$

The goal is to teach omnidirectional mobile robots to follow a control program based on recognition and data processing through identification devices such as smart cameras; Intelligent sensors, and control algorithms so that the robot follows certain trajectories in the space and working environment in accordance with parameters $e_{x,k} \approx 0,\ e_{y,k} \approx 0,\ e_{\theta,k} \approx 0$, with k is constant. Then the mobile robot will perform awareness in the working environment to make the automatic navigation smoothly (avoiding fixed obstacles and moving obstacles) perfectly to the destination [4, 6, 7, 20, 23, 26].

III. RESEARCH AND APPLICATION OF SARSA ALGORITHM FOR MOBILE ROBOT

The reinforcement learning method with the SARSA algorithm is developed to serve the intelligent calculation of science and technology in general and, in particular, cybernetics (automatic robot control). Where "action value equals present value plus the sum of optimal future values." With SARSA algorithm on-policy group in particular and reinforcement learning in general, everything is divided into "State – st" and "Action – at" with time denoted by a series of time steps. (t = 0, 1, 2, .vv ...). For a continuous work environment such as controlling a mobile robot, the first thing to do is to quantify the state space to get an update $S = \{S_1, S_2, ... S_m\}$ and quantize the action space to set

$A = \{a_1, a_2, \ldots a_n\}$, and the result is the medium generates reward $r_t = r(s_t, a) \in R$, for better understanding, we have an interactive learning environment diagram as shown in figure 2, [3, 9, 19].

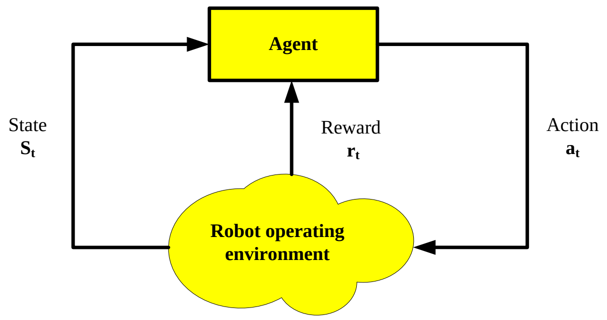

Fig. 2 The robot learning environment interactive diagram

The SARSA algorithm is an on-policy algorithm for Learning. The major difference between SARSA and Q-Learning is that the maximum reward for the next state is not necessarily used for updating the Q-values. Instead, a new action, and therefore reward, is selected using the same policy that determined the original action. The name SARSA actually comes from the fact that the updates are done $Q(s, a, r, s', a')$, where: s, a is the original state and action, r is the reward observed in the following state, and s', a' are the new state-action pair. The Q-value update rule is defined by:

$$Q(s_t, a_t) := Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (7)$$

It is straightforward to design an on-policy control algorithm based on the SARSA prediction method.

```
Algorithm: On-policy SARSA algorithm
Initialize Q(s,a) arbitrarily;
repeat (for each episode):
    Initialize $s$;
    Choose $a$ from $s$ using policy derived from $Q$;
    repeat (for each step of episode):
        Take action $a$, observe $r$, $s'$;
        Choose $a'$ from $s'$ using policy derived from $Q$;
        $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$;
        $s \leftarrow s'; a \leftarrow a'$;
    until $s$ is terminal
until all episodes end.
```

As in all on-policy methods, we continually estimate Q^π for the behavior policy π, and at the same time change π toward greediness with respect to Q^π. The general algorithm is given as: as you can see, there are two action selection steps needed for determining the next state-action pair along with the first. The parameters α and γ have the same meaning as they do in Q-Learning.

Suppose *{f₁, ..., fₙ}* are numerical features of the state and the action. Thus, $f_i(s, a)$ provides the value for the i-th feature for state s and action a. These features are typically binary, with domain [0, 1], but they can also be other numerical features. These features will be used to represent the Q-function.

$$Q_\omega(s,a) = \omega_0 + \omega_1 f_1(s,a) + \ldots + \omega_n f_n(s,a) \quad (8)$$

For some tuple of weights, $\omega = [\omega_0, \omega_1, \ldots, \omega_n]$. Assume that there is an extra feature f_0 whose value is always 1 so that ω_0 does not have to be a special case. An experience in SARSA of the form {s, a, r, s', a'} (the agent was in state s, did action a, and received reward r and ended up in state s,' in which it decided to do action a' provides the new estimate of $r + \gamma Q(s', a')$ to update Q(s, a). This experience can be used as a data point for linear regression.

Let $\delta = r + \gamma Q(s', a') - Q(s, a)$ Weight ωi is updated by:

$$\omega_i \leftarrow \omega_i + \alpha\delta f_i(s,a) \quad (9)$$

Similarly, we can use linear function approximation on the Q-learning algorithm. The transition is of the form {s, a, r, s'}, and the difference.

$$\delta = \left[r + \gamma \max_{a'} Q(s', a')\right] - Q(s,a) \quad (10)$$

The update rules for Q-values and weights are:

$$\begin{aligned} &\delta = Q(s,a) \leftarrow Q(s,a) + \alpha\delta; \\ &\omega_i \leftarrow \omega_i + \alpha\delta f_i(s,a) \end{aligned} \quad (11)$$

This update can then be incorporated into SARSA, giving the algorithm:

```
Algorithm: SARSA with linear function approximation
Input:
    $f = [f_1, ..., f_n]$: a set of features;
    $\gamma \in [0,1]$: discount factor;
    $\alpha > 0$: step size for gradient descent.
Initialize weights $\omega = [\omega_0, \omega_1, ..., \omega_n]$;
Observe current state $s$;
Select action $a$ (stochastic policy $\pi(a|s)$);
repeat
    Carry out action $a$ (transition probability $Pr(s'|s,a)$);
    Observe state $s'$ and receive reward $r$;
    Select action $a'$ using a policy based on $Q_\omega$ (stochastic policy $\pi(a'|s')$);
    Let $\delta = r + \gamma Q(s',a') - Q(s,a)$;
    for $i = 0$ to $n$ do
        Update weights: $\omega_i \leftarrow \omega_i + \alpha\delta f_i(s,a)$;
    end for
    $s \leftarrow s'$, $a \leftarrow a'$;
until Termination.
Output: An optimal weights $\omega*$.
```

In robotics, it's often usually to use linear or non-linear feature representations to handle the large-scale state-action spaces. At that time, the SARSA algorithm implements action agents for intelligent automatic navigation for mobile robots to perform trajectories to avoid dynamic as well as static obstacles during the movement of the mobile robot. The robot also calculates the shortest trajectory for the robot to move to the destination with the fastest path [2, 3, 9].

VI. RESEARCH RESULTS AND COMMENTS

With the algorithm researched and proposed in the second and third parts as above, we conduct research on the robot Turtlebot3 mobile robot with a structure including a two-wheeled robot to control the two sides and a multidirectional wheel in the section. The robot's head, the control circuit using a Raspberry Pi 3 embedded computer, has a control signal receiver unit from Jetson TX2, an Astra smart camera, and an intelligent sensor that then transmits commands to a smart microcontroller, etc. To record images from the environment as well as to measure the distance between omnidirectional mobile robots and unknown obstacles, mobile robots equipped with Rplidar 3D are placed on top to scan 360 degrees from values to obstructions, the robot's surrounding perceptual environment.

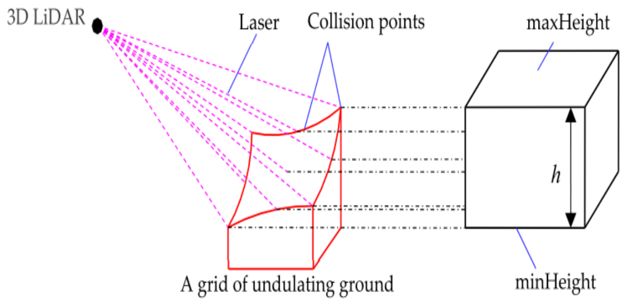

Fig. 3 Schematic diagram of 3D LiDAR terrain detection

When terrain reconstruction is performed in 3D space, the undulating ground and obstacles can be detected by 3D LiDAR. As shown in Figure 3, the irregular square is a block of undulating ground divided in a grid, The laser cast from a 3D LiDAR can detect all positions on the undulating ground, and all height information is updated to the variables of the node. As shown in the square on the right, the height of the top surface (maxHeight) of the square is the value of the node, and the height of the bottom surface (minHeight) is the value of the node.

In this section, some simulations are performed based on a powerful and environmental simulation engine in Gazebo. As shown in figure 4 shows the map built on Gazebo is a map created with strict walls, and a mobile robot can be controlled to move around fixed obstacles or obstacles. Mobile. This issue, in order to give the robot context for robot movements, is used to build action maps when the robot is active. Dark blue dotted lines show the robot's path when avoiding obstacles created by smart tree sensor, smart camera, and updated robot's current position (mobile robot is denoted by blue tree leaves) using geometric dimensions. This is a visual tool that can provide live updates of maps generated from the SLAM algorithm to control the robot. Furthermore, the omnidirectional mobile robot's trajectory in the map can also navigate automatically, and in this environment, obstacles can always be created, as shown in figure 4 for the robot to move.

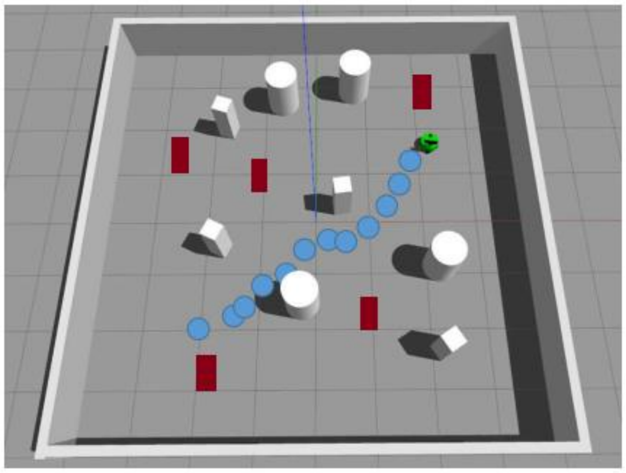

Fig. 4 Build visual maps and robot simulation models in Gazebo

An environment contains a robot (green) with a depth camera and randomly placed obstacles (white and brown) in Gazebo, as figure 4. The point cloud data generated by the depth camera with 125 degrees as figure 5, and point cloud data generated by the depth camera with 130 degrees as figure 6. Here is primarily a visualization tool that can provide live updates of maps generated from the SLAM and SARSA augmented learning algorithms. Furthermore, the robot's trajectory in the map can also be displayed in the real-world environment where the training and teaching, and identification process so that the robot knows during obstacle avoidance in a smart and perfect way.

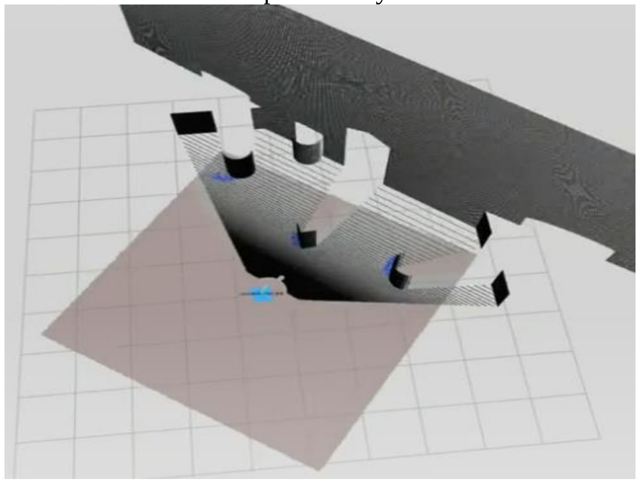

Fig. 5 The point cloud data is generated by the depth camera at an angle of 125 degrees

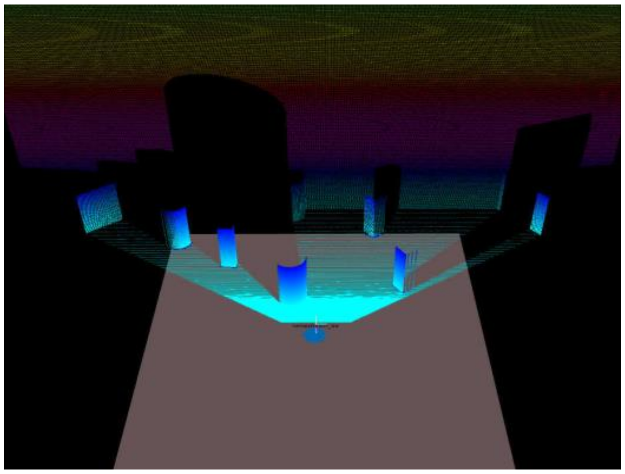

Fig. 6 The point cloud data is generated by the depth camera at an angle of 130 degrees

In addition, we can build a grid map system created from point cloud data to observe and identify obstacles, helping the robot to move optimally during the control process.

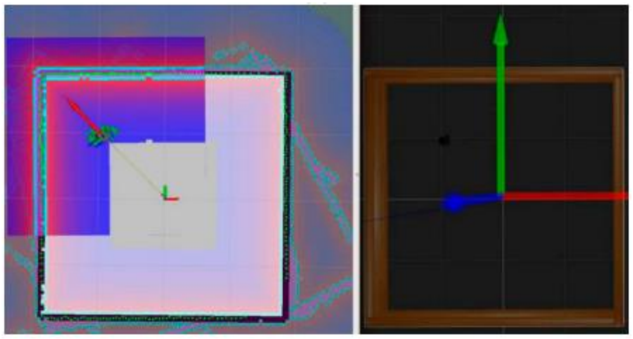

Fig. 7 Turtlebot3 relocalization and navigation in a square environment

In figure 7, the robot was set to the correct orientation and position near the upper left corner of the newly built map, the same as the real position shown on the right side of the figure. With human assistance, this initialed localization method made the particles quickly converge to a concentrated region and obtained the correct position. Then, combined with the robot odometer and laser data, robot localization and navigation experiments consistently maintained the correct trajectory. Robot relocalization with a manually set correct position.

These results showed that the SARSA algorithm is significantly better than other algorithms, which also indicated the advantages of the RL algorithm in image recognition [11, 21, 24]. Compared with the deep learning algorithm Q-learning, the SARSA algorithm is better than Q-learning in terms of value accuracy and strategy, which is also consistent with previous reports [10, 12, 15, 25]. The reinforcement learning technology is utilized to achieve the mapping from state to action and meet the mobile needs of mobile robots. The data have also proven that the robot path planning method based on reinforcement learning is an effective end-to-end mobile robot path planning method; this goal is to avoid obstacles in the defined and unspecified environment. The above results illustrate the feasibility of the proposed method in the path planning of mobile robots.

V. CONCLUSION

In this article, the author presents a mobile robot self-learning strategy without relying on prior experience under clear feedback. The author has studied the mobile robot navigation problem based on the enhanced learning method with the SARSA algorithm. The SARSA algorithm is applied to improve the self-learning, computation, and perception of mobile robots through interactions with an unknown environment. Tests were performed on automated intelligent navigation tasks for mobile robots. The simulation results in Gazebo and the fact that the stability and applicability of the SARSA algorithm are very effective, these new studies can completely apply to the control and navigation of mobile robots. In industrial factories in Vietnam as well as in the world, further, than previous studies [10, 12, 23], the research results of the paper are better. Then, the SARSA algorithm is simpler, making it possible to improve the quality of the robot's operation through intuitive tools, such as cameras and smart sensors to navigate with the goal of avoiding obstacles. Obstacles so that the robot reaches the destination without any obstacles.

ACKNOWLEDGMENT

This study was supported by the Department of Electrical Engineering, Faculty of Control Engineering, Le Quy Don Technical University (MTA - Military Technical Academy). No. 236, Hoang Quoc Viet Road, Bac Tu Liem district Ha Noi capital of Viet Nam. https://www.mta.edu.vn/.

REFERENCES

[1] Nguyen Doan Phuoc., The Advanced control theory, Science and Technics Publishing House, In Viet Nam, (2015).

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

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

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

[6] 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 .

[7] 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.

[8] A. Folkers, M. Rick, and C. Buskens., Controlling an autonomous vehicle with deep reinforcement learning., IEEE Intell. Veh. Symp. Proc., (2019) 2025–2031.

[9] Cuong Nguyen Manh, Tien Ngo Manh, Dung Pham Tien, Van Nguyen Thi Thanh, Manh Tran Van, Duyen Ha Thi Thanh, and Duy Nguyen Duc "Autonomous Navigation for Omnidirectional Robot Based on Deep Reinforcement Learning," IJMERR, 9(8)(2020) 1134-1139.

[10] L. a. S. H. Lin., Modeling and Adaptive Control of an Omni-Mecanum-Wheeled Robot., Intelligent Control and Automation, 4 (2013) 166-179.

[11] Pinto, L.; Gupta, A. Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, (2016) 16–21 3406-3413.

[12] Bicchi, A.; Kumar, V. Robotic grasping and contact: A review. In Proceedings of the 2000 ICRA, millennium Conference, IEEE International Conference on Robotics and Automation, Symposia Proceedings (Cat. No. 00CH37065), San Francisco, CA, USA, 24–28 1(2000) 348-353.

[13] Fang, B.; Jia, S.; Guo, D.; Xu, M.; Wen, S.; Sun, F. Survey of imitation learning for robotic manipulation. Int. J. Intell. Robot. Appl. 3(2019) 362–369.

[14] Zhang, F.; Leitner, J.; Milford, M.; Upcroft, B.; Corke, P. Towards vision-based deep reinforcement learning for robotic motion control. arXiv 2015.

[15] Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. Deep Reinforcement Learning: A Brief Survey. IEEE Signal Process. Mag. 34(2017) 26–38.

[16] Luong, N.C.; Hoang, D.T.; Gong, S.; Niyato, D.; Wang, P.; Liang, Y.C.; Kim, D.I. Applications of deep reinforcement learning in communications and networking: A survey. IEEE Commun. Surv. Tutorials 21(2019) 3133-3174.

[17] Cao, J.; Liu, W.; Liu, Y.; Yang, J. Generalize Robot Learning From Demonstration to Variant Scenarios with Evolutionary Policy Gradient. Front. Neurorobotics. (2020).

[18] Krishnan, S.; Garg, A.; Liaw, R.; Thananjeyan, B.; Miller, L.; Pokorny, F.T.; Goldberg, K. SWIRL: A sequential windowed inverse reinforcement learning algorithm for robot tasks with delayed rewards. Int. J. Robot. Res. 38(2019) 126 -145.

[19] Hwangbo, J.; Lee, J.; Dosovitskiy, A.; Bellicoso, D.; Tsounis, V.; Koltun, V.; Hutter, M. Learning agile and dynamic motor skills for legged robots. Sci. Robot. (2019).

[20] Golemo, F.; Taiga, A.A.; Courville, A.; Oudeyer, P.Y. Sim-to-real transfer with neural-augmented robot simulation. In Proceedings of the Conference on Robot Learning, New York, NY, USA, (2018) 29–31 817- 828.

[21] Mees, O.; Merklinger, M.; Kalweit, G.; Burgard, W. Adversarial skill networks: Unsupervised robot skill learning from video. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 30 May–5 June (2020) 4188–4194.

[22] Zhao, T.; Deng, M.; Li, Z.; Hu, Y. Cooperative Manipulation for a Mobile Dual-Arm Robot Using Sequences of Dynamic Movement Primitives. IEEE Trans. Cogn. Dev. Syst. 12(2020) 18–29.

[23] Deng, M.; Li, Z.; Kang, Y.; Chen, C.L.P.; the Chu, X. A Learning-Based Hierarchical Control Scheme for an Exoskeleton Robot in Human-Robot Cooperative Manipulation. IEEE Trans. Cybern. 50(2020) 112–125.

[24] R. K. e. a. Megalingam., ROS based autonomous indoor navigation simulation using SLAM algorithm., Int. J. Pure Appl. Math, (2018), 199-205.

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

[26] https://www.mathworks.com/help/reinfocermentlearning/ug/ddpg - Agent. html, (2020).