Proceedings of the Sixth International Conference on Research in Intelligent and Computing pp. 13–16
DOI: 10.15439/2021R21
ISSN 2300-5963 ACSIS, Vol. 27

Mobile robots interacting with obstacles control based on artificial intelligence

Tran Duc Chuyen
University of Economics-Technology for Industries
Hanoi, Vietnam

Roan Van Hoa
University of Economics-Technology for Industries
Hanoi, Vietnam

Nguyen Duc Dien
University of Economics-Technology for Industries
Hanoi, Vietnam

Tung Lam Nguyen
Hanoi University of Science and Technology
Hanoi, Vietnam
lam.nguyentung@hust.edu.vn

***Abstract*—In this paper, research on the applications of artificial intelligence in implementing Deep Deterministic Policy Gradient (DDPG) on Gazebo model and the reality of mobile robot has been studied and applied. The goal of the experimental studies is to navigate the mobile robot to learn the best possible action to move in real-world environments when facing fixed and mobile obstacles. When the robot moves in an environment with obstacles, the robot will automatically control to avoid these obstacles. Then, the more time that can be maintained within a specific limit, the more rewards are accumulated and therefore better results will be achieved. The authors performed various tests with many transform parameters and proved that the DDPG algorithm is more efficient than algorithms like Q-learning, Machine learning, deep Q-network, etc. Then execute SLAM to recognize the robot positions, and virtual maps are precisely built and displayed in Rviz. The research results will be the basis for the design and construction of control algorithms for mobile robots and industrial robots applied in programming techniques and industrial factory automation control.**

***Index Terms*—Mobile robots, artificial intelligence, DDPG algorithm, autonomous navigation, reinforcement learning.**

I. Introduction

NOWADAYS Artificial Intelligence (AI), Internet of Things (IoT), and robot controls are receiving a lot of attention. Robot technology has changed since the first introduction of robots in 1917. Today, machines are present in our lives, supporting us in everyday life, [1] - [5]. One of these new technologies is artificial intelligence that has come to life as well as robotics and machine tools technology, so robots can now properly process and manage information, and automatically perform certain tasks without human assistance, replacing humans in industrial factories. However, the ability to perceive the environment (feel) and make decisions (to take action) is a very difficult task for the computerized machines. Therefore, the field of Artificial Intelligence (AI) is needed for mobile robots to solve such problems, [3, 4, 5, 6].

In this paper, the authors present a robot control problem based on an intelligent and modern Deep Deterministic Policy Gradient (DDPG). The designed help the robot to navigate in a complex environment with obstacles. Experimental studies are per-formed on automated navigation for the mobile robot. More specifically, the author introduces the neural network structure to generalize and approximate the values of all states based on the DDPG artificial intelligence algorithm. This is a policy-based, online learning intensive learning algorithm, and is backboned by the Actor - Critic intelligent network architecture. The tests are conducted on the Gazebo emulator using a high-profile computer with a mobile robot, with its open-source extension to perform automated navigation tasks for mobile robots, and then carried out to experiments. The research results will be the basis for the research and application of mobile robots in practical production, in industrial plants, [6, 8, 10, 12].

II. Mobile robot model

Consider a mobile robot as in Fig. 1. Two coordinate frames are used to describe the motion of the mobile robot, the global coordinate frame *(X, Y)* which is earth-fixed and the other is the local coordinate frame (X_l, Y_l) which is attached to the mobile robot. The angle between the two coordinate frames is denoted as θ. The robot's motion will be defined for the navigation stack. As the global coordinate chosen in Fig. 1, it is clear that the robot's velocity contains three components: the linear velocity along OX and OY axes, and the angular speed, [6].

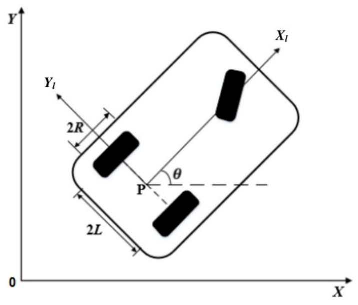

Fig. 1. The model mobile robot

To identify the position of the multi-directional mobile robot, the P-coordinate is selected on the robot's frame as its control center position reference point. Reference point P is positioned by the coordinates (*x, y*) in the global frame of the entire control environment for the robot. In order to formulate the motion of the mobile robot as component movements, it is necessary to define the motion mapping along the axes of the spherical frame with the motion along the coordinate of the local frame. This mapping is represented as the following expression:

$$\begin{bmatrix}\dot{x}_l\\ \dot{y}_l\\ \dot{\theta}_l\end{bmatrix}=\begin{bmatrix}cos\theta & sin\theta & 0\\ -sin\theta & cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} \tag{1}$$

By considering the limit of wheel slip $\dot{y}_l = 0$ implies that the wheel cannot slide orthogonal to the wheel plane, then it is straightforward to have:

$$-\dot{x}sin\theta+\dot{y}cos\theta=0 \tag{2}$$

Defining forward velocity $\dot{x}_l$ of the multi-directional mobile robot as v and rotation speed θ as *w*, the kinematics of the multi-directional mobile robot becomes:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix}=\begin{bmatrix}cos\theta\, 0\\ sin\theta\, 0\\ 0 \quad 1\end{bmatrix}\begin{bmatrix}v\\ w\end{bmatrix} \tag{3}$$

The forward velocity and rotation speed of the robot have the following relation:

$$v=\frac{1}{2}(v_l+v_r), \qquad w=\frac{v_l-v_r}{2L} \tag{4}$$

If we consider the linear velocities at the wheels and the angular velocities of the two qualified wheels, then we can get: $v_l = w_l R$ and $v_r = w_r R$. The angular velocities of the two standard wheels can be represented according to the forward velocity and rotation speed of the multi-directional mobile robot as:

$$w_l=\frac{v+wL}{R}, \quad w_r=\frac{v-wL}{R} \tag{5}$$

Considering the acceleration of the multi-directional mobile robot in its local working coordinate limit, then the robot dynamic can be written as [12]:

$$\begin{bmatrix}\dot{v}\\ \dot{w}\end{bmatrix}=\begin{bmatrix}\frac{1}{Rm} & \frac{1}{Rm}\\ \frac{L}{RI} & \frac{L}{RI}\end{bmatrix}\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} \tag{6}$$

where *m* and *I* are the robot mass and inertia, respectively; *R* is the radius of the two fixed wheels; *L* is the half of the distance between the two fixed standard wheels; $\tau = [\tau_1\ \tau_2]^T$ is the input torque vector exerted to the two fixed standard wheels, [1, 4, 8, 29]. The goal is to teach multi-directional mobile robots to track and follow certain trajectories within the right spaces and work environments with $e_{x,k} \approx 0,\ e_{y,k} \approx 0,\ e_{\theta,k} \approx 0$.

III. Application of artificial intelligence to mobile robot control

A. DDPG algorithm learning method

DDPG algorithm is built similar to the idea of Double Deep Q-Network. It is a model with reinforcement learning, online learning and an off-policy algorithm group with an Actor - critic network structure, [6, 7].

$$\max_a Q_{\theta Q}(s,a)=Q_{\theta Q}\left(s, arg\,max\, Q_{\theta Q}(s,a)\right) \tag{7}$$

If we build a neural network to choose the optimal action for a particular state, $\mu_{\theta\mu}(s) arg \max_a Q_{\theta Q}(s,a)$, then optimize component $Q_{\theta Q}$ according to the network parameters θ_μ just created, then we have:

$$\theta_\mu \leftarrow arg \max_{\theta\mu} Q_{\theta Q}\left(s, \mu_{\theta\mu}(s)\right) \tag{8}$$

This optimization considers the change of $Q_{\theta Q}$ according to the variable θ_μ, or in other words, the evaluation of the action. This can be calculated using a string rule like the following expression:

$$\frac{dQ_{\theta Q}}{d\theta_\mu}=\frac{dQ_{\theta Q}}{d\mu}\cdot\frac{d\mu}{d\theta_\mu} \tag{9}$$

So building a function that approximates the value of action by a $\mu_{\theta\mu}(s)$ neural network here is the main difference of DDPG. Thus, in each DDPG algorithm structure, there are always two components, one is Actor $\mu_{\theta\mu}(s)$ and the other is Critic $Q_{\theta Q}(s,a)$ lattice. The relationship between the two aforementioned components and their connection to the enhanced learning environment of this algorithm when controlling the multi-directional mobile robot in a random open environment with many fixed and obstacle obstacles mobile. At that time, the DDPG algorithm always meets the requirements of the control quality as well as the working quality of the robot.

In fact, we can understand more clearly that the DDPG algorithm is improved from other algorithms to have the ability to compute continuous action space problems for multi-directional mobile robots, when Then we go to update the target function as follows: For an input sample set of $(s_i, a_i, R_i, s_i^{'})$, then the formula updates the target function value as follows:

$$y_i = R_i + \gamma Q'(s_i^{'}, \mu'(s_i^{'})) \tag{10}$$

Then we compute the loss function for the sample M value $(s_i, a_i, R_i, s_i^{'})$ to train the written network:

$$J=\frac{1}{M}\sum_{i=1}^{M}(y_i-Q(s_i,a_i))^2 \tag{11}$$

According to the string rule in expression (9), the gradient is calculated as follows:

$$\nabla_{\theta\mu} J=\frac{1}{M}\sum_{i=1}^{M}G_{ai}G_{\mu i} \tag{12}$$

In which, $G_{ai} = \nabla_a Q(s_i, a)$ is the output gradient of the Critic network according to variable a, estimated by the Actor network $a = \mu(s_i)$. And $G_{\mu i} = \nabla_{\theta\mu}\mu(s_i)$ is the gradient of the Actor network input according to the model parameter θ_μ.

The DDPG algorithm with neural network training and training always ensures the requirements for accurate and reliable control, DDPG Agent updates the network parameter θ of the Q rating (*S, a*) at each step of the process. network trainer, to do action a, receive new algorithm R, then it will significantly improve the performance of the control model for multi-directional mobile robot when using DDPG algorithm control programming. Moreover, DDPG always explores the space for action constantly and this is also a great challenge for scientists, [6, 13, 14].

B. The robot navigation using DDPG algorithm

The DDPG is a member of the actor-critic algorithm, which contains four neural networks: Current critic network $Q(s,a|\theta^Q)$, current actor network $\mu(s|\theta^\mu)$), target critic network $Q'(s,a|\theta^{Q'})$, and target actor network $\mu'(s|\theta^{\mu'})$, where $\theta^Q, \theta^\mu, \theta^{Q'}, \theta^{\mu'}$ are the network weights. The ingredient Q' and μ' are copy of Q and μ respectively in the structures. Both $\theta^{Q'}$ and $\theta^{\mu'}$ are partially updated from the current networks at each timestep. The current critic network is updated by minimizing the loss function. Then, the gradient function is continuous, ensuring that the robot's agent action when controlled in an obstacle environment and now the algorithm is updated in a continuous space. The specific process of the DDPG algorithm for navigating multi-directional mobile robots is described in detail as follows, [6, 7].

The performance of the DDPG algorithm deployed is very positive on the multi-directional mobile robot control model. One of the reasons authors chose to study this algorithm for the primary control of multi-directional robots was to develop something industrially controllable. Comparing the DDPG algorithms with other algorithms has also been successful for the goal of mobile navigation for robots. Some tests for each form have been given and it is clear that the DDPG algorithm works better than other algorithms like Q-learning, RL, etc.

Algorithm: Deep Deterministic Policy Gradient algorithm for the mobile robot

```
1: Randomly initialize critic network $Q(s,a|\theta^Q)$, actor network $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
2: Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$
3: Initialize replay buffer $R$
4:   for episode = 1 to $M$ do
5:     Initialize a random process $N$
6:     Receive initial state $s_1$ from environment $E$
7:     for $t = 1$ to $T$ do
8:       Select action $a_t = \mu(s_t|\theta^\mu) + N_t$ according to the current actor network
9:       Execute action $a_t$ in the environment $E$, and receive reward $r_t$ and new state $s_{t+1}$
10:      Store transition $(s_t, a_t, r_t, s_{t+1})$ in buffer $R$
11:      Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$
12:      Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
13:      Update the critic by minimizing the loss: $L = \frac{1}{N}\sum_i [y_i - Q(s_i, a_i|\theta^Q)]^2$
14:      Update the actor policy using the sampled gradient:
               $\nabla_{\theta^\mu}\mu|_{s_i} \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|\theta^Q)|_{s=s_i,a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s=s_i}$
15:      Update the target networks:
               $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}, \theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
16:    end for
17: end for
```

Therefore, to build a complete DDPG algorithm, it is always necessary to meet the needs of selecting a control action to a robot, executing the action, receiving rewards, storing and sampling to train the algorithm, calculating of the target function. Subsequently updating the model parameters by minimizing the loss function on all selected samples, followed by selecting the method to update the target neural network parameters, and finally updating the environmental discovery coefficient during the control process [3, 5, 6, 15, 16].

IV. EXPERIMENTAL RESULTS

A. The research TurtleBot mobile robot

Here the authors go to study the model of mobile robot: with the actual hardware architecture of this robot is shown as shown in figure 3, in which each hardware module will perform a number of tasks, in the sequence of activities of this mobile robot: such as finding a path, crossing obstacles, etc.

In the tests, the authors perform several tasks in the sequence of the mobile robot's operations, such as the Ubuntu-powered Raspberry Pi 3 Model B +. The Raspberry Pi embedded computer directly processes information from a range of sensors including the Astra smart camera and the smart sensor then transmits commands to a smart microcontroller. To record images from the environment as well as measure the distance between mobile robots and unknown obstacles, mobile robots are equipped with cameras and smart sensors, in which the smart camera can do 360 degrees laser scanning and ranges within 12m generate map data to be used for the mapping process. The smart microprocessor control circuit receives control signals from Raspberry Pi 3 Model B+.

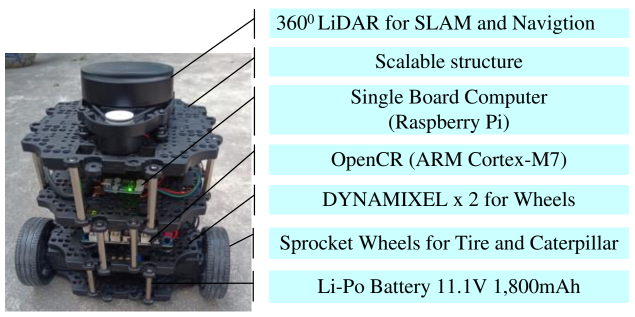
360^0 LiDAR for SLAM and Navigtion
Scalable structure
Single Board Computer (Raspberry Pi)
OpenCR (ARM Cortex-M7)
DYNAMIXEL x 2 for Wheels
Sprocket Wheels for Tire and Caterpillar
Li-Po Battery 11.1V 1,800mAh

Fig. 3. Pictures of the actual TurtleBot robot

B. The experimental results

In this section, the actual mobile robot TurtleBot is tested in a real environment, which is the environment used for real world testing consisting of flowerpots as obstacles.

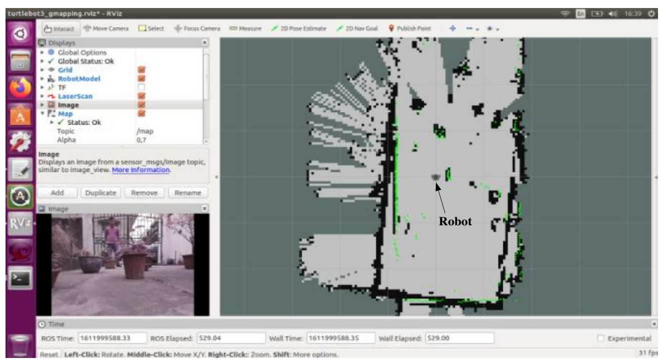

Fig. 5. The TurtleBot robot is in the process of creating concurrent mapping

In order to set up an operating environment for the purpose of controlling and navigating the robot, the authors have created a map with the goal of creating an obstacle environment of flowerpots with many different colors, then programmed for the controller, for the mobile robot as depicted in Fig. 5 and 6. The environment here includes obstacles created by different flowerpots, the robot's path will be taught in advance via computer, Wifi network, and the actual TurtleBot Robot. based on DDPG artificial intelligence algorithm and SLAM algorithm. This is primarily a visualization tool that can provide live updates of maps generated from the SLAM and DDPG algorithms. Furthermore, the vehicle'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 the obstacle avoidance process intelligently and perfectly.

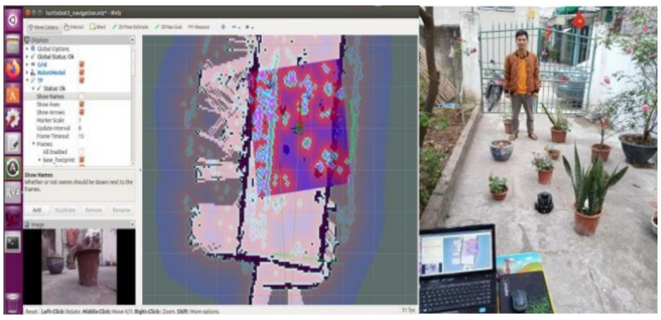

Fig. 6. Results of navigating Robot TurtleBot done on Rviz

From the simulation results and experimental results, compared with some other algorithms: Q-learning, DDPG algorithm is better than DQN, Q-learning in terms of value accuracy and control strategy are presented in [13, 16], this is also consistent with the DDPG algorithm that the authors have proposed in the paper. Accelerated learning technology and rapid action processing in large environments, can be used to achieve action status maps and meet the mobility needs of mobile robots. The data also demonstrated that the robot's path planning method based on DDPG method is better than previous studies, as shown in [13]. The above results prove the strength of the algorithm, the optimal problem of the proposed method in planning the path of the mobile robot that allows interaction with objects based on artificial intelligence.

V. Conclusions

The paper has presented the construction and control formulation of the mobile robot TurtleBot control structure. It is shown that the good effect of Slam when using Gmapping stack to construct 2D map. This article also performs a successful route planning for mobile robot when interacting with objects that are static and unknown obstacles due to constant update of local path in the environment. Tasks are based on data generated from smart cameras and smart sensors. Furthermore, robot activity can be monitored through visual tools.

References

[1] M. N. Cirstea, A. Dinu, J.G. Khor, M. McCormick, "*Neural and Fuzzy Logic Control of Drives and Power Systems*", Linacre House, Jordan Hill, Oxford OX2 8DP , First published 2002.

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

[3] Nils J. Nilsson, "*The quest for artificial interlligence a history of ideas and achievements*", Web Version Print version published by Cambridge University Press, Publishing September 13, 2010, http://www.cambridge.org/us/0521122937.

[4] Mohit Sewak, "*Deep Reinforcement Learning*", Springer Nature Singapore Pte Ltd. 2019.

[5] Latombe, J.C. "*Robot Motion Planning*"; Kluwer Academic Publishers: Norwell, MA, USA, 1992.

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

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

[8] Tran Hoai Linh, "*Neural network and its application in signal processing*", Hanoi Polytechnic Publishing House, 2015.

[9] 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); pp: 1502-1508, Volume 11; Issue 9; 2020.

[10] Roan Van Hoa, Tran Duc Chuyen, Nguyen Tung Lam, Nguyen Duc Dien, Tran Ngoc Son, Vu Thi To Linh, "*Reinforcement Learning based Method for Autonomous Navigation of Mobile Robots in Unknown Environments"*, Proceedings of the 2020 International Conference on Advanced Mechatronic Systems, Hanoi, Vietnam, December 10 - 13, 2020.

[11] Evan Prianto, MyeongSeop Kim, Jae-Han Park, Ji-Hun Bae, and Jung-Su Kim, "*Path Planning for Multi-Arm Manipulators Using Deep Reinforcement Learning: Soft Actor–Critic with Hindsight Experience Replay*", Sensors, Published: 19 October 2020.

[12] Deepak Ramachandran, Rakesh Gupta, "*Smoothed Sarsa: Reinforcement Learning for Robot Delivery Tasks"*, 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, May 12-17, (2009).

[13] M. U. KHAN, Mobile Robot Navigation Using Reinforcement Learning in Unknown Environments, BALKAN JOURNAL OF ELECTRICAL & COMPUTER ENGINEERING, Vol. 7, No. 3, July 2019 (2019).

[14] G. A. Cardona, C. Bravo, W. Quesada, D. Ruiz, M. Obeng, X. Wu, and J. M. Calderon, *Autonomous Navigation for Exploration of Unknown Environments and Collision Avoidance in Mobile Robots Using Reinforcement Learning*, Conference Paper, April 2019, DOI: 10.1109/ SoutheastCon42311.2019.9020521, (2020).

[15] Luis V. Calderita, Araceli Vega, Sergio Barroso-Ramírez, Pablo Bustos and Pedro Núñez, *Designing a Cyber-Physical System for Ambient Assisted Living: A Use-Case Analysis for Social Robot Navigation in Caregiving Centers*, pp 2-24, Sensor. (2020).

[16] 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, 2019.