SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549 / https://doi.org/10.14445/23488549/IJECE-V10I3P103

Volume 10 Issue 3, 15-22, March 2023

Original Article

Research and Development of Omnidirectional Mobile Robot Tracking Control Based on Artificial Intelligence

Chau Thanh Phuong

Faculty of Electronic Engineering Technology, University of Economics - Technology for Industries, Viet Nam.

Corresponding Author : ctphuong@uneti.edu.vn

Received: 04 February 2023 Revised: 07 March 2023 Accepted: 17 March 2023 Published: 28 March 2023

Abstract *- This paper presents the research and construction of a motion tracing control system for omnidirectional mobile robots based on reinforcement learning techniques in automatic control. The process of controlling a mobile robot in a flat environment with definite and unknown obstacles, taking into account the nonlinear factor of interference. Research and application of programming tools are operating systems for mobile robots (Robot Operating System - ROS). From updated information on maps, operating environment, robot control position, and obstacle identification (SLAM) to calculate the movement trajectory of a three-wheeled omnidirectional mobile robot. The positioning system calculates the orbital tracking for the robot based on the Q-learning algorithm. The results of simulation research in the Gazebo environment and running tests on real Turtlebot mobile robots have shown the practical effectiveness of the research problem of tracking motion tracking and intelligent navigation for mobile robots.*

Keywords *- Three-wheeled mobile robot, Self-propelled robot, Automatic system, ROS, Artificial intelligence, Q-learning algorithm, Reinforcement learning.*

1. Introduction

Mobile robots have recently been used in mission-critical tasks and various activities. Due to the intelligent capabilities we humans have equipped them with: from new control algorithms, more optimized drive systems, control motors (servo motors), etc., making the operation of mobile robots more and more accurate [1]. These robots can be used as a stand-alone base or with rigid arms with multiple degrees of freedom and flexibility based on the nature of the action execution task [1-4]. In actual environmental conditions, the robot always has uncertain non-snow factors, which is undeniable; in this case, that factor is considered a nonlinear factor in control; needs to be overcome [5, 6]. Therefore, the research and development; of modeling the robot under ideal conditions, when environmental factors are not considered: noise factors, factors, wheel slip, obstacle course, etc., results include insufficient precision and large errors. Different from traditional wheeled robots (standard wheels), mobile robots using omnidirectional wheels have additional advantages, such as the ability to change position and orientation flexibly because they have the ability to move forward and rotate simultaneously or independently. Usually, the wheel is arranged along the axis of the robot. However, for the omnidirectional mobile robot, the wheels are arranged on the sides, and one wheel guides the robot to take advantage of the degrees of freedom of the omnidirectional wheel. In the motion control technique of mobile robots, the problem of orbital tracking and fast impact is the most necessary requirement. Therefore, using a common controller such as PID control, fuzzy control, PD control, linear control, PI control, LQR control, etc., is unsuitable for the system. Mobile robot systems always have these nonlinear factors that cannot be overcome [4, 5, 6]. Therefore, this is a very important issue that needs to be considered even when using intelligent control algorithms, such as artificial intelligence, sustainable adaptive optimal control, etc., for robots [7-9].

The study of the process of controlling mobile robots in many different fields and tasks has clearly demonstrated the importance of the robot transmission system model. From a number of studies that have been done to model the control system of mobile robots, as shown in document [7], some models of transmission systems for industrial robots in general and automatic robot models have been studied operating mobile robots in particular. Document [8], the study on programming control, navigation, tracing trajectory in flat space, and problem space for self-propelled robots and mobile robots does not consider nonlinear variable factors such as those above. However, in these works, they only stop at the design and simulation of the system without clearly assessing the nonlinear factor of the drive system for the robot. In general, the problem of controlling mobile robots for industry, transportation, medicine, etc., is being studied by domestic and international scientists [8-11].

In the fields of control engineering and information

technology, reinforcement learning is a subfield of machine learning that studies how an agent in an environment should choose which actions to take in order to maximize its effectiveness and maximize a certain reward in the long run. Reinforcement learning algorithms try to find a strategy that maps states of the environment to the actions the agent should take in those states [4, 5]. The working environment for robot control is often represented as a finite state Markop decision process (MDP), and reinforcement learning algorithms for this context are heavily involved in engineering techniques and dynamic planning. The transition probabilities and gain probabilities in the MDP are usually random but static during the robot control problem [12-17].

Based on the above analysis, the paper builds an orbital tracking controller for a three-wheeled omnidirectional mobile robot based on Q-learning reinforcement learning techniques combined with the MPC model applied in automatic control. In which the controller is designed when affected by some nonlinear components of the model and unknown perturbation factors, simulation results and egg testing using Matlab Simulink software and other supporting tools.

2. The Dynamic Model for Omnidirectional Mobile Robot Robot

The study and analysis of robot dynamics is a complex mechanical system with many masses and possibly many degrees of freedom. Each degree of freedom performs one motion and is controlled by an electric drive. Furthermore, a robot is a control object containing many interrelated motors. To build a control system model for the robot, we consider a mobile robot, as shown in Figure 1, with three wheels, two wheels on both sides: left rudder, right rudder and front wheel (multi-wheel) direction) can make the mobile robot balance and not cause any movement restrictions for the mobile robot [3, 4], [6], [9, 10, 18].

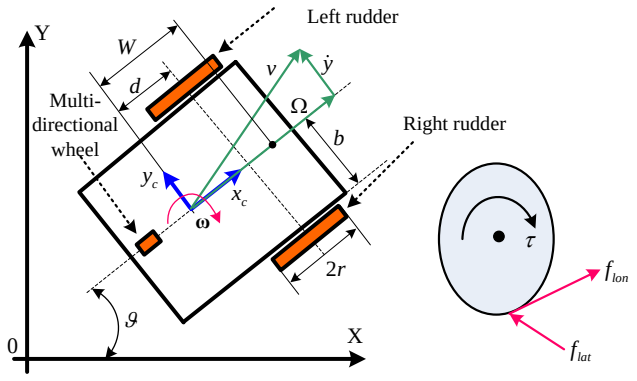

Fig. 1 The kinetic model for controlling a mobile robot

In which figure 1 depicts a wheeled mobile robot with two active wheels x_c and y_c is the robot's position in the plane, ϑ is the robot orientation, φ_r is the angle of the right wheel, φ_l is the angle of the wheeled vehicle on the left, b is half the width of the robot, d is the distance from the center of gravity to the wheel axle, and r is the wheel radius. For this robot, the free motion of the movable wheel is not considered in the kinematic model, as shown below.

Then we call the general coordinator of the system $q = [x_c\ y_c \vartheta\ \phi_r\ \phi_l]^T$, the dynamic equation of the transmission system, when taking into account the wheel slip phenomenon is set up as follows [1], [4]:

$$M(q)\ddot{q} + c(q,\dot{q}) = N\tau(t) - A^T(q)\lambda + F(q,\dot{q}) \quad (1)$$

Where, $[M(q)]_{5\times5}$ is the inertial matrix, $[c(q,\dot{q})]_{5\times1}$ is the Coriolis and centrifugal force matrix, $[\tau]_{2\times1}$ is the input vector of the system, $[N]_{5\times5}$ is the matrix of the input coefficients of the system, λ is the factor vector Lagrange and $[F(q,\dot{q})]_{5\times1}$ are traction vectors.

The constraints of the system in the process of tracking the robot's motion trajectory, which, when considering the nonlinear factor, are written in the following form:

$$\begin{aligned} \dot{x}_c\, cos(\vartheta) + \dot{y}_c\, sin(\vartheta) + b\dot{\vartheta} &= r\dot{\phi}_r - \dot{\zeta}_r \\ \dot{x}_c\, cos(\vartheta) + \dot{y}_c\, sin(\vartheta) - b\dot{\vartheta} &= r\dot{\phi}_l - \dot{\zeta}_l \\ -\dot{x}_c\, sin(\vartheta) + \dot{y}_c\, cos(\vartheta) - d\dot{\vartheta} &= \dot{\gamma} \end{aligned} \quad (2)$$

where ζ_r is the longitudinal slip of the right wheel, ζ_l is the longitudinal slip of the left wheel, and γ is the lateral slip.

$$A(q) = \begin{bmatrix} cos(\vartheta) & sin(\vartheta) & b & -r & 0 \\ cos(\vartheta) & sin(\vartheta) & -b & 0 & -r \\ -sin(\vartheta) & cos(\vartheta) & -d & 0 & 0 \end{bmatrix} \quad (3)$$

$$S(q) = \begin{bmatrix} \frac{r(b\,cos(\vartheta)-d\,sin(\vartheta))}{2b} & \frac{r(b\,cos(\vartheta)+d\,sin(\vartheta))}{2b} \\ \frac{r(d\,cos(\vartheta)+b\,sin(\vartheta))}{2b} & \frac{r(-d\,cos(\vartheta)+b\,sin(\vartheta))}{2b} \\ \frac{1}{2b} & -\frac{1}{2b} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (4)$$

The constraint matrix of the system $A(q)$ is inferred based on $A(q)\dot{q} = 0$ the system constraints when taking into account the nonlinear component, and the empty space matrix of the constraints is obtained in the form shown in equations (3) and (4).

The kinematics of a mobile robot in nonlinear tracing control is written in the following form:

$$\dot{x}_c = \Omega\, cos(\vartheta) - \Delta\, sin(\vartheta)$$

$$\dot{y}_c = \Omega\, sin(\vartheta) + \Delta\, cos(\vartheta)$$

$$\dot{\vartheta} = \omega \quad (5)$$

In there:

$$\omega = \frac{r\dot{\phi}_r - r\dot{\phi}_l}{2b} - \frac{r\dot{\zeta}_r - r\dot{\zeta}_l}{2b}; \Omega = \frac{r\dot{\phi}_r + r\dot{\phi}_l}{2} - \frac{r\dot{\zeta}_r + r\dot{\zeta}_l}{2};$$

$$\Delta = d\left(\frac{r\dot{\phi}_r - r\dot{\phi}_l}{2b} - \frac{r\dot{\zeta}_r - r\dot{\zeta}_l}{2b}\right) + \dot{\gamma} \quad (6)$$

Here we rewrite (5) in matrix form, then $\dot{q}$ becomes equation (7) as follows:

$$\dot{q} = H(q)(R - \dot{\zeta}) + \psi \quad (7)$$

$$\text{In there: } R = [v \quad \omega]^T \quad (8)$$

$$\psi = [-\dot{\gamma}\, sin(\vartheta) \quad \dot{\gamma}\, cos(\vartheta) \quad 0 \quad \dot{\zeta}_r \quad \dot{\zeta}_l]^T \quad (9)$$

$$\dot{\zeta} = \left[\frac{r(\zeta_r + \zeta_l)}{2} \quad \frac{r(\zeta_r + \zeta_l)}{2b}\right]^T \quad (10)$$

$$H(q) = \begin{bmatrix} cos(\vartheta) & -d\, sin(\vartheta) \\ sin(\vartheta) & d\, cos(\vartheta) \\ 0 & 0 \\ \frac{1}{r} & \frac{b}{r} \\ \frac{1}{r} & -\frac{b}{r} \end{bmatrix} \quad (11)$$

Taking the derivative of (7) and substituting it into expression (1), we get (12) as follows:

$$M(q)\left[\dot{H}(q)(R - \dot{\zeta}) + H(q)(\dot{R} - \ddot{\zeta}) + \dot{\psi}\right] + \\ + c(q, \dot{q}) = N\tau - A^T(q)\lambda \quad (12)$$

From the equation $S^T(q)A^T(q) = 0$, we work with multiplying $S^T(q)$ into both sides of equation (12), and we ignore the term $A^T(q)\lambda$, and then we get (13) as follows:

$$\dot{R} = (S^T(q)M(q)H(q))^{(-1)} \\ \times \left[-S^T(q)M(q)\dot{H}(q)(R - \dot{\zeta}) + \right. \\ \left. + S^T(q)N\tau - S^T(q)M(q)\dot{\psi} - S^T c(q, \dot{q})\right] + \ddot{\zeta} \quad (13)$$

Considering $v = \begin{bmatrix}\dot{\phi}_r \\ \dot{\phi}_l\end{bmatrix}$, since the state space equation of the system is written in terms of $\dot{x} = \begin{bmatrix}\dot{q} \\ \dot{v}\end{bmatrix}$, it needs to be calculated and transformed, then we get equation (14) as follows:

$$R(t) = \begin{bmatrix} v \\ \omega \end{bmatrix} = Pv, P = \frac{r}{2}\begin{bmatrix} 1 & 1 \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix} \quad (14)$$

$$\dot{x} = \begin{bmatrix}\dot{q} \\ \dot{v}\end{bmatrix} = \begin{bmatrix} H(q)(R - \dot{\zeta}) + \psi(q, \dot{\eta}) \\ (S^T MHP)^{(-1)}\left[-S^T M\dot{H}(R - \dot{\zeta}) - S^T M\dot{\psi} - S^T c\right] + P^{(-1)}\ddot{\zeta} \end{bmatrix} + \\ + \begin{bmatrix} 0 \\ (S^T MHP)^{(-1)} S^T N\tau \end{bmatrix} \quad (15)$$

Now we consider the problem of longitudinal and lateral traction forces related to the robot drive system as follows [7, 20, 27]: the drag force is a function of the slip ratio (sr), and the slip angle (sa) is determined under forms $sr = \dot{\zeta} / max(|r\dot{\phi}|, |r\dot{\phi} - \dot{\zeta}|)$ and $sa = arctan(\dot{\gamma} / |r\dot{\phi} - \dot{\zeta}|)$. By considering that both components (sr) and (sa) are small in this paper, then the longitudinal and transverse tensile forces are estimated linearly in the form of equations (16) and (17), where $\delta > 0$ and $\varsigma < 0$, according to [15], [27].

$$f_{doc} = \delta \frac{\dot{\zeta}}{|r\dot{\phi} - \dot{\zeta}|} \quad (16)$$

$$f_{ngang} = \varsigma \frac{\dot{\gamma}}{|r\dot{\phi} - \dot{\zeta}|} \quad (17)$$

In expressions (16), (17) and in figure 1, f_{doc} is f_{lon}; f_{ngang} is f_{lat}, which clearly shows the nonlinear factor in the process considering the kinematics and dynamics of the robot. The process of automatic movement and navigation or avoiding obstacles in the robot's path always follows the motion trajectory and generates forces at the wheels; these forces are usually generated when encountering obstacles, undulating roads, etc.

3. Research and Application of Q - Learning Algorithms for Robot

3.1. The Q - Learning Algorithms

The reinforcement learning method with a Q-learning algorithm is a branch of machine learning developed to serve intelligent computation for the field of science and technology in general and in terms of cybernetics in particular; Robot control techniques are being researched and applied to develop algorithms. This is a model to study reinforcement learning from offline to online control, which is the enhanced dynamic programming method IDP (Incremental Dynamic Programming), [3, 4], [22]. To design optimal learning rules for precise traction control; Online approximation of the nonlinear control problem.

With Q-learning in particular and reinforcement learning in general, everything is divided into “state - st” and “action – at” with time represented by a series of time steps (t = 0, 1, 2, 3 etc.). For a continuous working environment such as controlling a self-propelled robot, the first thing to do is to quantize the state space to update $S = \{S_1, S_2, \dots S_m\}$ and

quantize the action space to set $A = \{a_1, a_2, \dots a_n\}$, resulting in a The school generates rewards $r_t = r(s_t, a) \in R$, to understand better we have the learning environment interaction diagram as shown in Figure 2.

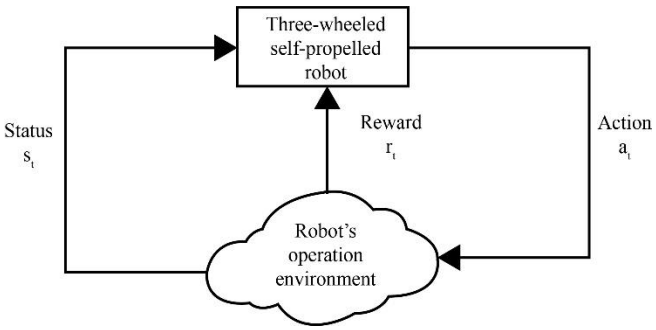

Fig. 2 The diagram of interaction with the learning environment of an omnidirectional mobile robot

Then, the way Q-learning works is to compute and store the value of Q on a particular action and state, Q(s, a). All information and experience accumulated from previous calculations will be coded into an evaluation table.

We calculate the total reward obtained after time t as R_t returned as follows:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \quad (18)$$

where, $0 \le \gamma < 1$ is the deduction factor for the rewards. The smaller the value of γ, the more focused the reward is while performing the action. Then, the action value function (function Q) is defined as follows:

$$Q^n(s, a) = E_\pi\{R_t | s_t = s, a_t = a\} \quad (19)$$

where, $E_\pi\{\dots\}$ represents the expectation under the stochastic policy in the action space. The function $Q^n(s, a)$ represents the total expected discount reward when we choose action a under state s and then choose action under policy π. The function Q is described as a recursive formula as follows:

$$Q^\pi(s, a) = \sum_{s' \in S} Pr(s'|s, a)\, r.((s, a, s') + \gamma \sum_{a' \in A} \pi(a'|s') Q^\pi(s', a')) \quad (20)$$

where S and A are the state and action set, respectively. From this formula, we can determine that the function Q according to the optimal policy π^*, that is, the optimal Q function, satisfies the following equation, which is called the Bellman optimal equation:

$$Q^*(s, a) = E_{s'}\{r_t + \gamma \max_{a'} Q^*(s', a')\} \quad (21)$$

In the Q-learning algorithm, by iteratively updating the function Q used in expression (21) based on experimental data, the function Q randomly converges to $Q^*(s, a)$. Thus, the optimal policy can be defined as an ambitious policy of $Q^*: a^* = argmax_a(s, a)$. In practice, the robot's learning agent on the move must explore the action environment because the Q function is unreliable and needs to choose an action to be used broadly as a stochastic policy; it is then allowed to choose a probabilistic action for an input state s. More specifically, policy μ will engage in choosing an action that maximizes the function Q in state s with probability a of *1- μ, μ* [0, 1] and allows a random action to be selected with the remaining probability. When states and actions are discrete and distinct, a simple way to represent the function Q is to use as a table of values for all pairs of states, acting as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha.\left((r + \gamma \max_{a'} Q(s', a') - Q(s', a')\right) \quad (22)$$

In which $0 < \alpha \le 1$ is the learning rate and the larger the learning rate, the faster the new data will be updated. With this algorithm, the table Q converges to the optimal function Q under the convergence condition of the random approximation. On the other hand, since this is based on random approximation, an appropriate amount of data is required for all (s, a) pairs. In the tabular Q - Learning method, when the number of elements in the state or action space is very large or the state or action space is continuous, we usually represent the function Q as a parameter function Q (s, a; θ) using the parameters θ and then update the parameters according to the gradient expression as follows:

$$\theta \leftarrow \theta + \alpha\left(target_Q - Q(s, a; \theta)\right)\nabla_\theta Q(s, a; \theta) \quad (23)$$

Here, "$target_Q$" is the target value based on the optimal Bellman equation (17), and it is calculated as follows:

$$target_Q r(s, a; s') + \gamma \max_{a'} Q(s', a'; \theta) \quad (24)$$

Function Q is updated in its consistent sequence. The Q-learning algorithm is a method based on value functions and from the value function to give an optimal policy, in which the approximate value of the function Q is regressed to the target value, which depends on it is him. This means the true value changes automatically when the learning rule is updated. Therefore, when a non-linear function, such as a neural network, is used to approximate p, this learning becomes unstable due to kinetic changes in the target and in the worst case, the Q function will diverge [2, 6, 8], [23-26].

3.2. The Intelligent Tracking Control for Mobile Robots using the Q-learning Algorithm

In the Q-learning algorithm, the values of the navigation control positions for the robot are usually updated by the instantaneous differential method, using the difference

between an iteration to estimate and calculate the Q-value function according to the parameters expression (22) above. When encountering navigational problems on the way, let the robot move with many different states and move actions (left, right), avoid obstacles (moving obstacles, fixed obstacles)), etc. Then we choose $\alpha = 0.1$; $\gamma = 0.95$; At this time, the robot moves in many different situations; at this time, the process of updating table Q is performed.

At the beginning of algorithm training, the subject will go once or twice to the right. However, as soon as the action to the left is selected, this action will continue to be selected on subsequent moves because it always gets rewarded for performing this left action. In this process, the robot always follows the correct motion trajectory according to the dynamics process. The goal of the model during navigation for the robot is to keep it within an allowable limit, that is, ±5 degrees. At first, the robot model, Q matrix, and policy π will be initialized—some important points to navigation during migration, like non-finite states. In the limited range, there can be hundreds and thousands of elevation angles, and thousands of columns are impossible when updating the Q - learning algorithm. So, we have sorted the state values into 20 state angles from -10 degrees to 10 degrees. For the action value, we have chosen ten different velocities, and they are [-250, - 100, - 50, - 25, - 10, 10, 25, 50, 100, 250] ms-1. The Q matrix has 20 columns, each representing a state and ten rows representing every action. Initially, Q values are assumed to be 0, and some random action is assigned to every state in policy π. We trained for 1550 episodes, each with 1000 repetitions. At the beginning of each training session, the simulation is refreshed. Whenever the robot's state exceeds the limit, it is penalized by assigning a reward of -100. Table Q is updated at each step according to expression (22). From there, we have the Q-learning algorithm to set up the automatic trajectory tracking and navigation process for the omnidirectional mobile robot, which is done as follows:

Algorithm: *Q-learning algorithm.*
1: Set all Q(s, a) randomly;
2: Repeat (for each episode):
3: Set s as one of the initial states;
4: Repeat (for each step of an episode)
5: Select action a according to state s using policy derived from Q;
6: Take action *a*, observe *r*, next state s_{t+1}

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

7: Reward ← 1;
8: Update Q;
9: Update π sate ← sate new
10: Until s is terminal.

The Q-learning algorithm implements action agents for intelligent automatic navigation for the self-propelled robot to perform trajectories to avoid dynamic obstacles as well as static obstacles during the robot's movement. Time to calculate the shortest trajectory for the robot to move to the destination with the fastest path. One of the most important breakthroughs in reinforcement learning was the development of Q-learning by Watkins in the literature [15]. The Q-learning algorithm performs the update process on the action values. The best action of the following state is used as the return expectation during the update. The Q-learning update process is a step taken according to the proposed algorithm. Then the identification and estimation of this parameter is the process of optimizing the value function, then from here, the value at this time will give the optimal policy; at this time, the accumulated values are considered as values correct update; Here (we do the recognition when the speed is moving, the process of tracking the robot's trajectory to perform intelligent navigation when there are nonlinear factors appear) so that the robot moves without encountering any problems an obstacle on the way.

4. Results and Discussion

From the research, calculation, and setting up of the controller for the control system for the three-wheeled omnidirectional mobile robot, combined with Q-learning reinforcement learning algorithm. The author has conducted research with real robots; combined with building simulation models on Matlab Simulink 2021. The process of training and implementing the algorithm runs with the ROS operating system with the computer and the internet running on the processor, high configuration Dell computer: Core i7 intel, graphics card GTX 2022 TI Ram 8GB, with the following parameters: The self-propelled robot used is the Turtlebot3 Buger robot with a maximum linear speed of 0.22 m/s and a maximum angular velocity of 2.84 rad/s (162.72 degrees/s).

To implement intelligent navigation and traction for a three-wheeled mobile robot. The author implemented based on the Q-learning algorithm studied and proposed above to conduct some simulations with the following results:

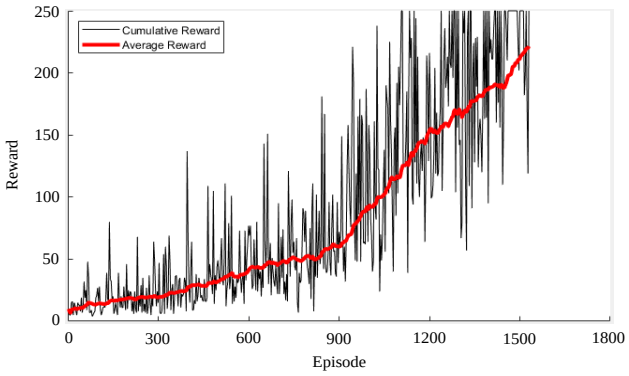

Fig. 3 The average reward results of the learning process

From the simulation results, we can see that the process of traction control and dynamic obstacle overcoming can be

randomly generated in the environment. The Q-learning algorithm performs the task. The author has deployed 1550 learning batches; in the simulation, the first 1000 episodes are for accessing a set of waypoints with a total optimal orbital length of 125m. The last 1050 sets are for a set of waypoints projection with a total path length of 155m. Figure 3 depicts the total reward obtained after 1550 learning sessions. Although there are many fluctuations due to the changing complexity of the environment and the efficiency of current navigation algorithms, the recognition process also tends to show an increasing total reward of total reward over the period study has yielded high results. Detailed parameters used to train the robot: state size is 26, contains 24 values of Laser distance sensor (LDS), distance to target and angle to target. The size of the training sample pool we choose here is 64, and the optimizer is Adam [4, 8], with a learning rate of 0.0003, amortization factor $\gamma = 0.999$.

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 3 Model B + embedded computer directly processes information from a range of sensors, including the Astra smart camera. The smart sensor then transmits commands to a smart microcontroller.

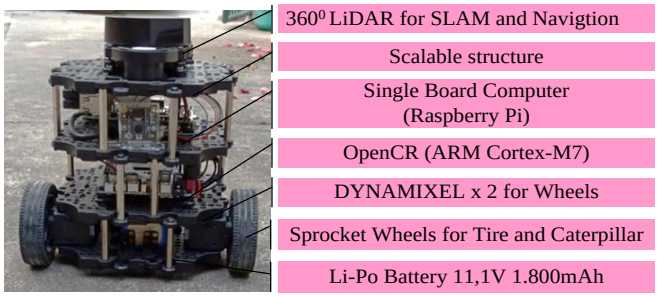

Fig. 4 Realistic image of TurtleBot omnidirectional mobile robot

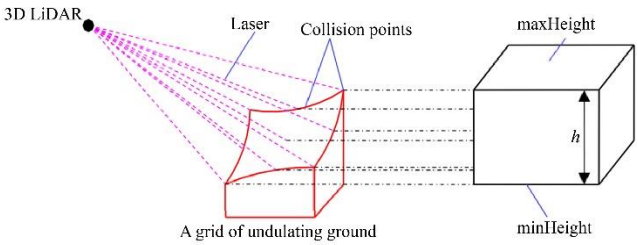

Fig. 5 The structure Diagram of sensor 3D LiDAR terrain detection

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 of laser scanning and ranges within 15m to generate map data to be used for the mapping process.

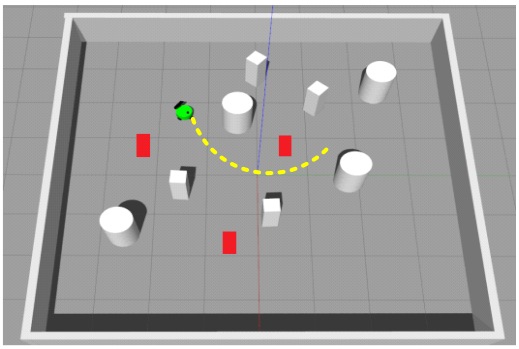

Fig. 6 Build visual maps and robot models in the Gazebo environment

Figure 6 shows a map built on Gazebo; the generated map has rectangular obstacles, a circular cylinder, and a mobile robot (green) with a depth camera and other objects. The obstacles are randomly placed in the Gazebo, as shown in figure 6. The controlled and guided robot automatically moves around the environment to obtain the necessary data that will be used to build the map. The light brown line is the laser scan signal generated from the RPLidar, and the robot's current position is updated using geometric measurements.

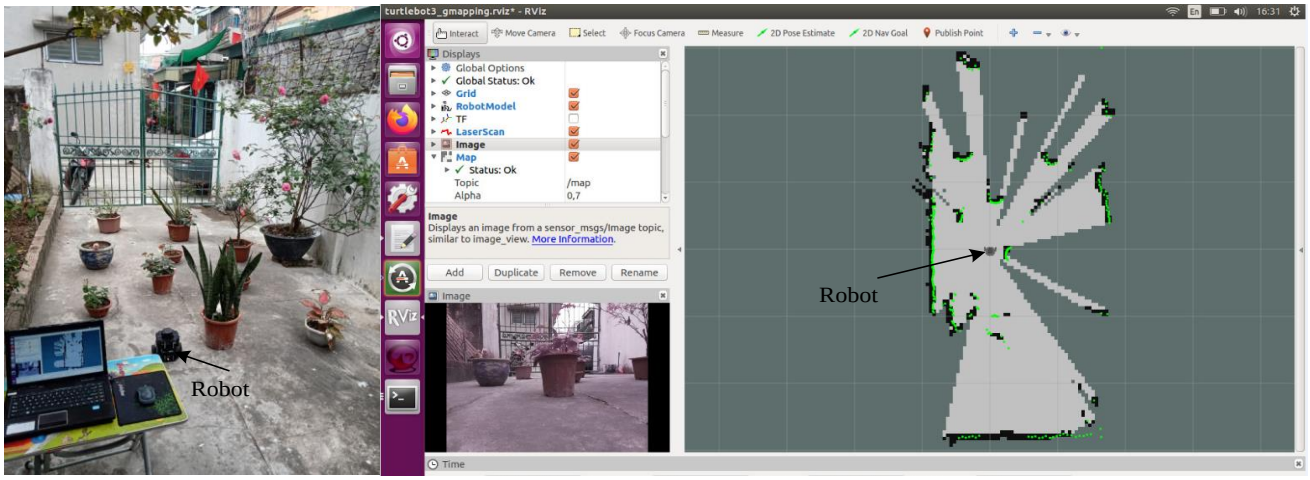

Fig. 7 The interface to execute the Q-learning algorithm and SLAM on ROS with the area around the robot containing obstacle information

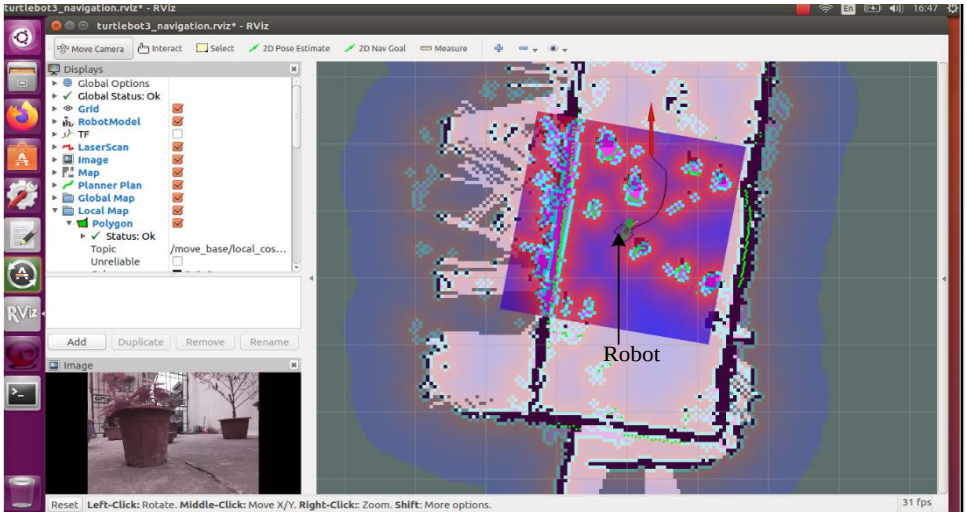

Fig. 8 The tracking control results for the omnidirectional mobile robot Turtlebot on a real map with fixed obstacles in Rviz

The results show that: Figure 7 shows the movement process when the robot performs Q-learning and SLAM algorithms on ROS, with the area around the robot containing full information about obstacles. Figure 8 shows the trajectories and automatic navigation results for the omnidirectional mobile robot Turtlebot in a real map with fixed obstacles in the Rviz environment.

Compared with other algorithms, the Q-learning deep learning algorithm has more advantages; in value accuracy and control strategy. Hierarchical reinforcement learning technology is used to achieve more accurate mapping and computational probabilities from states to actions and meet mobile robots' mobility needs. The data has also demonstrated that the deep reinforcement learning-based robot path planning and tracking method is an optimal method for mobile robots for efficient end-to-end travel. The above results illustrate the feasibility of the proposed method in planning the path and the control process of a three-wheeled omnidirectional mobile robot.

5. Conclusion

This paper has presented the study of kinematics and traction and motion control for the omnidirectional mobile robot system both in simulation and experiment with the omnidirectional mobile robot Turtlebot. The robot hardware is also optimally built to facilitate the integration of ROS-based peripherals. Furthermore, the robot's activity can be tracked and monitored through the visualization tool. The positioning system has calculated the robot's global and local trajectory based on applying the Q-learning algorithm. Research results show that: simulation and testing studies on the ROS operating system and Rviz environment show the robot's ability to automatically navigate to the desired target locations and avoid static obstacles and obstacles motion at the scene. Migrate in simple to complex environments safely and efficiently without any crashes. These new researches are completely applicable: self-propelled robots, industrial robots, medical robots, and robots in public transport, especially in tracking motion control and automatic navigation of mobile robots in the field factories in Vietnam as well as in the world.

Acknowledgments

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

[2] N. T. Tuan, *Base Deep Learning*, The Legrand Orange Book,Version 2, Last Update, 2020.

[3] Mohit Sewak, *Deep Reinforcement Learning*, Frontiers of Artificial Intelligence Springer Nature, 2019. [Publisher link]

[4] V. T. T. Nga, O. X. Loc, and T. H. Nam, *Enhanced Learning in Automatic Control with Matlab Simulink*, Hanoi Polytechnic Publishing House, 2020.

[5] N. T. Luy, *Textbook of Machine Learning and Intelligent Control Application*, Publishing House of Industrial University of Ho Chi Minh City, Ho Chi Minh, 2019.

[6] Xiaogang Ruan et al., "Mobile Robot Navigation Based on Deep Reinforcement Learning," *Chinese Control and Decision Conference*, pp. 6174-6178, 2019. [CrossRef] [Google Scholar] [Publiser link]

[7] Rajesh Kannan Megalingam et al., “ROS Based Autonomous Indoor Navigation Simulation Using SLAM Algorithm”, *International Journal of Pure and Applied Mathematics*, vol. 118, no. 7, pp. 199-205, 2018. [Google Scholar] [Publisher link]
[8] Charu C. Aggarwal, *Neural Networks and Deep Learning*, Springer International Publishing AG, Part of Springer Nature, 2018. [Publisher link]
[9] Thanh Tung Pham, Minh Thanh, and Chi-Ngon Nguyen, “Omnidirectional Mobile Robot Trajectory Tracking Control with Diversity of Inputs,” *International Journal of Mechanical Engineering and Robotics Research*, vol. 10, no. 11, 2021. [CrossRef] [Google Scholar] [Publisher link]
[10] Hiep Do Quang et al., “Design a Nonlinear MPC Controller for Autonomous Mobile Robot Navigation System Based on ROS,” *International Journal of Mechanical Engineering and Robotics Research*, vol. 11, no. 6, pp. 379 - 388, 2022. [Google Scholar] [Publisher link]
[11] Hiep Do Quang et al., “Mapping and Navigation With Four-Wheeled Omnidirectional Mobile Robot Based on Robot Operating System,” *2019 International Conference on Mechatronics, Robotics and Systems Engineering (MoRSE)*, pp. 54–59, 2019. [CrossRef] [Google Scholar] [Publisher link]
[12] Yuankai Wu et al., “Deep Reinforcement Learning of Energy Management with Continuous Control Strategy and Traffic Information for a Series-Parallel Plug-in Hybrid Electric Bus,” *Applied Energy*, vol. 247, pp. 454-466, 2019. [CrossRef] [Google Scholar] [Publisher link]
[13] Shixiang Gu et al., “Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates,” *2017 IEEE International Conference on Robotics and Automation*, pp. 3389–3396, 2017. [CrossRef] [Google Scholar] [Publisher link]
[14] L. T. T. Nga, and L. H. Lan, “Controlling the Swarm Robot to Avoid Obstacles and Search for Targets,” 2015.
[15] Evan Prianto et al., “Path Planning for Multi-Arm Manipulators Using Deep Reinforcement Learning: Soft Actor–Critic with Hindsight Experience Replay,” *Sensors*, vol. 20, no. 20, pp. 1-22, 2020. [CrossRef] [Google Scholar] [Publisher link]
[16] H. T. K. Duyen et al., “Controlling Self-Propelled Robot Object Tracking by Exponential Sliding Control Algorithm,” *Research Journal Military Science and Technology*, ACME Special Issue, 2017.
[17] Van Nguyen Thi Thanh et al., “Autonomous Navigation for Omnidirectional Robot Based on Deep Reinforcement Learning”, *International Journal of Mechanical Engineering and Robotics Research*, vol. 9, no. 8, pp. 1134-1139, 2020. 10.18178/Ijmerr.9.8.1134-1139. [CrossRef] [Google Scholar] [Publisher link]
[18] D. N. Thang, P. T. Dung, and N. Q. Hung, “Research on Obstacle Avoidance Problems for Self-Propelled Robots on the Basis of Enhanced Deep Learning DQN,” *Journal of Military Science and Technology Research*, Special Issue of FEE National Conference , p. 48-56, 2020.
[19] Wen-Kung Tseng, and Hou-Yu Chen, "The Study of Tracking Control for Autonomous Vehicle," *SSRG International Journal of Mechanical Engineering*, vol. 7, no. 11, pp. 57-62, 2020. [CrossRef] [Publisher link]
[20] Avi Singh et al., “*End-to-End Robotic Rein-Forcement Learning without Reward Engineering*,” University of California, Berkeley 2019. [CrossRef] [Google Scholar] [Publisher link]
[21] Yuda Irawan, Hendry Fonda, and Yulisman, Mardeni, "Garbage Collecting Ship Robot Using Arduino Uno Microcontroller Based on Android Smartphone," *International Journal of Engineering Trends and Technology*, vol. 69, no. 6, pp. 25-30, 2021. [CrossRef] [Google Scholar] [Publisher link]
[22] Rajesh Kannan Megalingam, Jeeba M Varghese, and Aarsha Anil S, “Distance Estimation and Direction Finding Using I2C Protocol for an Auto-Navigation Platform,” *International Conference on VLSI Systems, Architectures, Technology and Applications (VLSI-SATA)*, pp. 1-4, 2016. [CrossRef] [Google Scholar] [Publisher link]
[23] Sandeep B.S., and Supriya P, “Analysis of Fuzzy Rules for Robot Path Planning,” *Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 309-314, 2016. [CrossRef] [Google Scholar] [Publisher link]
[24] Giada Giorgi, Guglielmo Frigo, and Claudio Narduzzi, “Dead Reckoning in Structured Environments for Human Indoor Navigation,” *IEEE Sensors Journal*, vol. 17, no. 23, pp. 7794-7802, 2017. [CrossRef] [Google Scholar] [Publisher link]
[25] [Online]. Available: http://wiki.ros.org/ros/tutorials, 2-2023
[26] [Online]. Available: https://emanual.robotis.com/
[27] Justin Fu et al., “Variational Inverse Control with Events: A General Framework for Data-Driven Reward Definition,” *32nd Conference on Neural Information Processing Systems*, 2018. [CrossRef] [Google Scholar] [Publisher link]