OPTIMAL NAVIGATION PLANNING FOR MOBILE ROBOTS USING REINFORCEMENT LEARNING (RL) ALGORITHM

TỐI ƯU ĐIỀU HƯỚNG CHO ROBOT TỰ HÀNH SỬ DỤNG THUẬT TOÁN HỌC TĂNG CƯỜNG

Ho Manh Tien[1], Vo Thanh Ha[1,*]

DOI: http://doi.org/10.57001/huih5804.2024.120

ABSTRACT

The QL controller solved the path optimization problems for mobile robots. The QL algorithm predicts the mobile robot's course by learning from prior observations of the surroundings. On the other hand, the QL method calculates the states' Q values to offer massive deals to the Q table. The QL algorithm had optimal navigation planning for mobile robots in a dynamic environment. The mobile robot communicated with the control script by the robot operating system (ROS). The mobile robot is code-programmed using Python in the ROS operating system and the QL controller on Gazebo software. This QL controller is improved for the computation time, convergence time, planning trajectories accuracy, and avoidance of obstacles. Therefore, the QL controller solved the path optimization problems for mobile robots. The QL controller's efficiency was evident in its ability to adjust to changing environments swiftly, ensuring seamless navigation without compromising safety. By leveraging the power of machine learning and advanced algorithms, the mobile robot could adapt its trajectory in real-time, responding to obstacles and dynamic conditions with precision. This groundbreaking approach optimized path planning and enhanced the overall performance of mobile robots, paving the way for a new era of intelligent and autonomous robotic systems.

Keywords: *Mobile robot, RL, ROS, QL.*

TÓM TẮT

Thuật toán QL dự đoán đường đi của robot di động bằng cách học hỏi từ những quan sát trước đó về môi trường xung quanh. Mặt khác, phương pháp QL tính toán các giá trị Q của các trạng thái để đưa ra các tính toán lớn cho bảng ma trận Q. Thuật toán QL có kế hoạch điều hướng tối ưu cho robot di động trong môi trường năng động. Robot di động giao tiếp với tập lệnh điều khiển bằng hệ điều hành robot (ROS). Robot di động được lập trình mã bằng ngôn ngữ Python trên hệ điều hành ROS kết hợp với bộ điều khiển QL trên phần mềm Gazebo. Bộ điều khiển QL này được cải thiện về thời gian tính toán, thời gian hội tụ, lập kế hoạch quỹ đạo chính xác và tránh chướng ngại vật. Do đó, bộ điều khiển QL đã giải quyết được vấn đề tối ưu hóa đường dẫn cho robot di động. Hiệu quả của bộ điều khiển QL thể hiện rõ ở khả năng điều chỉnh nhanh chóng theo môi trường thay đổi, đảm bảo điều hướng liền mạch mà không ảnh hưởng đến an toàn. Bằng cách tận dụng sức mạnh của máy học và các thuật toán tiên tiến, robot di động có thể điều chỉnh quỹ đạo của nó trong thời gian thực, phản ứng chính xác với các chướng ngại vật và điều kiện động. Cách tiếp cận đột phá này đã tối ưu hóa việc lập kế hoạch đường đi và nâng cao hiệu suất tổng thể của robot di động, mở đường cho kỷ nguyên mới của hệ thống rô bốt tự hành thông minh.

Từ khóa: *Robot di động, RL, ROS, QL.*

[1]Faculty of Electrical and Electronics Engineering, University of Transport and Communications, Vietnam

[*]Email: vothanhha.ktd@utc.edu.vn

Received: 20/01/2024

Revised: 20/3/2024

Accepted: 25/4/2024

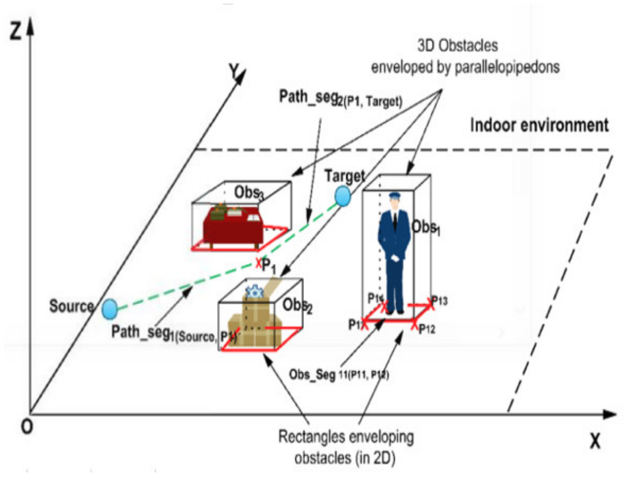
1. INTRODUCTION

Mobile robots are becoming a crucial element of contemporary society's progress. It does dangerous tasks that are challenging for people, including search and rescue operations, aiding in epidemic areas, and investigating remote worlds. Therefore, a crucial aspect of designing mobile robots is strategizing the robot's path [1, 2]. The trajectory planning will focus on reaching the destination quickly, with little energy consumption, and avoiding obstacles. Mobile robots currently include global route planning, local path planning, static path planning, and dynamic path planning [4]. Researchers have recently published more trajectory-planning studies for mobile robots aimed at avoiding obstacles in the operating environment. The system includes linear, nonlinear, and intelligent algorithms. The mobile robots described in reference [5] use an Artificial Potential Field (APF) algorithm for motion planning. The checkerboard strategy was used in a different study to find the best mobile route by repeatedly running the simulation software. Other notable pathfinding methods for static obstacle avoidance include the A* algorithm, D algorithm, random tree algorithm (RRT), and optimization particle swarm (PSO). Based on the research results, mobile robots are limited to functioning in a static environment. The mobile robot adjusted its speed and path to avoid obstacles. Orbital navigation planning for mobile robots is effective in dynamic scenarios [13]. Therefore, artificial intelligence techniques like the GA-Fuzzy approach [14] and ANFIS [15] have been used. As stated in reference, scientists have extensively used the RL algorithm in entertainment games and information technologies [16]. Several scholars have used the algorithm's primary feature, simple controller design, to strategize orbital navigation for mobile robots [17-19]. Conversely, the RL method improves the route somewhat and extends the time needed for convergence computation. The QL algorithm forecasts the trajectory of a mobile robot by analyzing previous observations of its environment, as shown by the research [20, 21]. Q-Learning (QL) enhances computational efficiency and convergence. The QL technique computes the Q values of states to generate large deals in the Q table. Planning the path for the mobile robot using the QL algorithm took a lengthy time due to the need to access all action-state pairs in complicated and dynamic scenarios. The QL method provided excellent navigation planning for mobile robots in a dynamic environment.

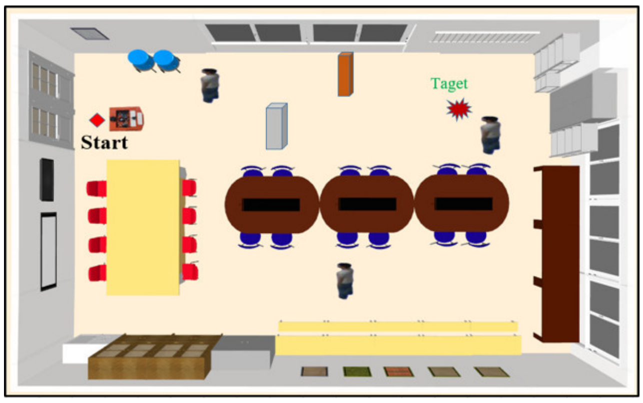
This paper will be broken into four sections. The introduction discusses the route planning control issues faced by the mobile robot. Section two details the mathematical modeling of an operating system designed for a mobile robot. Section three will outline the process of creating the optimal path for mobile robots via the QL algorithm. Part four concludes by evaluating the solution's effectiveness for the QL algorithm via simulation.

2. MATHEMATICAL MODELING OF OPERATING FOR A MOBILE ROBOT

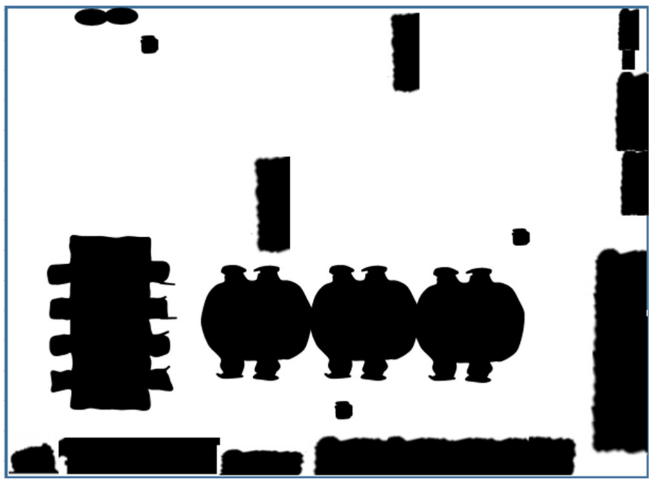
2.1. Obstacle modeling in mobile robot operating environment



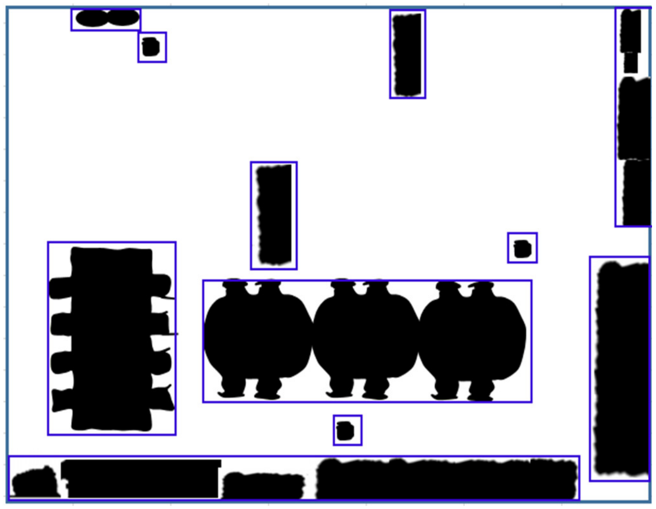
(a) Obstacles using cubes



(b)3D modelling of indoor environments



(c) Binary map of environment



(d) Approximation of the obstacles by rectangles

Fig. 1. Obstacle estimation for route planning

The obstacle modelling is built on a rectangular box. It is considered the optimal geometry that allows easy estimation of any obstacle shape among static and dynamic path optimization techniques that collect obstacle estimates. Obstacles using cubes, as shown in Fig. 1(a). Using this geometry to plan the robot in 2D, we can replace blocks using this geometry while avoiding 3D obstacles. A realistic scene with 3D blocks (chairs, tables, etc.) is depicted in Fig. 1(b). The binary map of the modelled environment is shown in Fig. 1(c).

2.2. Mathematical model for a mobile robot

The mobile robot will move from the starting point start (x_S, y_s) to the destination point target (x_T, y_T), the purpose of the robot path planning is to find the optimal path from the Start point to the Target point, this path is connected by n nodes (N_i, i = 1…n) and (n-1) each segment is 2 consecutive nodes connected to each other. Assume the robot moves in an environment with m known obstacles as shown in Fig.1. Each obstacle is modelled surrounded by a rectangle with four vertices $p_1(x_1, y_1)…p_4(x_4, y_4)$ and four edges obs_seg_{tj} ($t \in \{1…4\}, j \in \{1…m\}$). Where p_1 is the bottom left corner of the rectangle. Similar to obstacles in the environment, the mobile robot is also estimated by a rectangle of four point; their coordinates change according to the robot's current position. The mathematical equation of each segment defined by the two points (p_k, p_l) of a rectangle enclosing an obstacle is given as the following Eq. (1):

$$\text{seg}(p_k, p_l) = \begin{cases} y - y_k = \dfrac{y_l - y_k}{x_l - x_k}(x - x_k) \\ \text{Min}(x_l, x_k) \le x \le \text{Max}(x_l, x_k) \end{cases} \quad (1)$$

The following notations describe indices and parameters used in the mathematical model: n: is the number of nodes on the trajectory created from the starting point to the destination; m: is the number of obstacles in the environment; i ($i \in \{1\ldots n\}$): fragment index generated by the node i and i + 1; j ($j \in \{1\ldots m\}$): index of the jth obstacle in the navigation environment; k, l ($k, l \in \{1\ldots 4\}$): indices of the point that defines an obstacle; r ($r \in \{1\ldots 4\}$): index of the segment r from the rectangle approximating the mobile robot; t ($t \in \{1\ldots 4\}$): index of the segment t that defines the rectangle of an obstacle; N_i:ith node of the path; obs_j ($j \in \{1\ldots m\}$) : jth obstacle; $path\_seg_{tj}$ (N_i, N_{i+1})($i \in \{1\ldots n\text{-}1\}$) : ith segment of the path defined by two nodes (N_i, N_{i+1}); obs_seg_{lj} (p_k, p_l)($k, l \in \{1\ldots 4\}$) j($j \in \{1\ldots m\}$) : lth segment of jth obstacle defined by two points (p_k, p_l); CurrentPos: current location of the robot; Rob_seg_r (p_k, p_l)(r, k, $l \in \{1\ldots 4\}$): rth segment of the rectangle approximating the robot.

The mathematical model's decision variables are computed as follows:

$$B_{i,t,j} = \begin{cases} 1 \text{ if } \exists P_1, P_2 \left| \{P_1, P_1\} \in (path\_seg_i) \wedge \{P_1, P_1\} \in (obs\_seg_{tj}) \right. \\ \qquad \forall j(j \in \{1...m\}, \forall t(t \in \{1...4\}, \forall i(i \in \{1...n\} \\ 0 \qquad\qquad \text{Otherwise} \end{cases} \quad (2)$$

$$A_{r,t,j} = \begin{cases} 1 \text{ if } \exists P_1, P_2 \left| \{P_1, P_1\} \in (path\_seg_r) \wedge \{P_1, P_1\} \in (obs\_seg_{tj}) \right. \\ \qquad \forall j(j \in \{1...m\}, \forall r, t(t \in \{1...4\} \\ 0 \qquad\qquad \text{Otherwise} \end{cases} \quad (3)$$

The objective function is to find the shortest path is written according to the Eq. (4)

$$\text{Minimize}\left(\sum_{i=1}^{i=n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}, \forall i \in \{1...n-1\} \right) \quad (4)$$

Eq. (5) requires each node to be unique:

$$(x_{i+1} \ne x_i) \vee (y_{i+1} \ne y_i), \forall i \in \{1...n-1\} \quad (5)$$

Path segments do not overlap in the environment by Eq. (6).

$$\sum_{i=1}^{i=n-1} \sum_{j=1}^{m} B_{i,t,j}, \forall i \in \{1...n\}, t \in \{1...4\}, j \in \{1...m\} \quad (6)$$

The way nodes for the robot to overcome obstacles are calculated by Eq. (7).

$$\sum_{j=1}^{j=m} A_{r,t,j}, \forall r, t \in \{1...4\}, j \in \{1...m\} \quad (7)$$

All variables A and B must be binary to satisfy requirement Eq. (8).

$$B_{i,t,j} \in \{0,1\}, A_{r,t,j} \in \{0,1\}, \forall i = \{1...n\}, r, t = \{1...4\}, j = \{1...m\} \quad (8)$$

3. Q-LEARNINGS ALGORITHMS IN PATH PLANNING FOR MOBILE ROBOTS

The Q-learning (QL) algorithm uses the concept of reward and punishment is created the environment. First, state and action variables are developed and discretized according to the path-planning job. The reinforcement value is then stored in a Q-value function matrix, and a reward function is built to meet the conditions of obstacle avoidance and the shortest route. To overcome the exploration and exploitation balance issue and increase convergence speed, the -balancing methods and action selection algorithm are devised to improve the QL learning process. Following QL training, the optimum pairings of state and action are acquired, as are the optimal control rules, which are then utilized to execute local route planning. Steering rules are included to avoid the incomplete visiting issue for the pairings of state and action to increase route planning efficiency. Finally, the developed procedure was tested. The findings indicate that, even in a complex environment, the robot can design an optimum or suboptimal course while avoiding obstacles.

Fig. 2 illustrates how a mobile robot selects an action based on the appropriate policy, executes that action, and receives status (s) and reward (r) from the navigation environment. A state contains the robot's current position in its workspace while optimizing paths, while an action is a movement the robot makes to transition from one state to another.

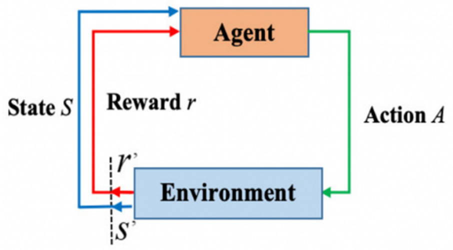


Fig. 2. Q-learning algorithm

The Q value is built for the robot to decide to earn the greatest reward. It is calculated as follows:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\left[r(s_t, a_t) + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad (9)$$

Where: α represents the learning rate, γ is the discount factor, $s_t \max\limits_{a \in A} Q(s_{t+1}, a)$ signifies the maximum Q-valua among all feasible actions in the new state a_t, and denotes the immediate reward/penalty earned by the agent after executing a move at the state s_{t+1}.

Based on Eq. (9) a state matrix - which acts like a lookup table. From there, for each state of the robot, find the action with the most significant Q value (Fig. 3).

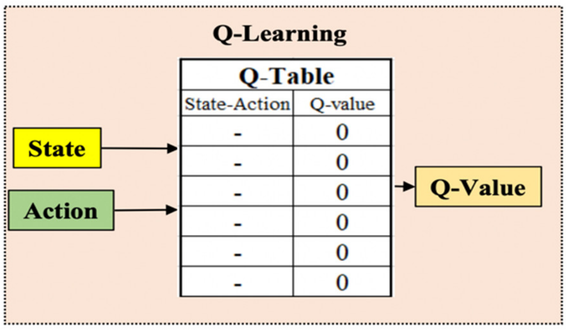


Fig.3. Table of Q parameters according to the state-action matrix

Reinforcement learning is a random process so the Q values will differ before and after the action. It is called a temporary difference.

$$TD(a,s) = r(s_t, a_t) + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t) \quad (10)$$

Thus, the matrix needs to update the weights based on the Eq. (10):

$$Q_t(s_t, a_t) = Q_{t-1}(s_t, a_t) + \alpha TD_t(s_t, a_t) \quad (11)$$

where: α is the arithmetic coefficient. Through the times the robot performs actions; $Q(s_t, a_t)$ it will gradually converge.

A programming program for the Q-learning algorithm in robot pathfinding is written as follows:

```
Algorithm 1: Classical Q-Learning algorithm begin
Initialization:
    Q(s_t, a_t) ← {0}, (states and m actions)
for (each episode):
(1) set s_t ← a random state from the states set s;
while (s_t ≠ Goal stage)
(2) Choose a_t in s_t by using an adequate policy (ε -greedy, etc.);
(3) Perform action a_t, and receive reward/penalty and s_{t+1};
(4) Update Q(s_t, a_t) using Eq (9);.
    s_t ← s_{t+1}
end-while
end-for
end
```

The size of the Q table grows exponentially following the number of states and actions in an environment with conditions.

In this situation, the process becomes computationally expensive and requires much memory to hold the Q values. Imagine a game where each state has 1000 actions. A table with a million cells will be needed. Given the vast amount of computational time, one of the main problems when using the QL algorithm in path optimization is that accessing all the Action-State pairs during the mining process is complexly generated, which affects the orbital convergence.

4. SIMULATION RESULTS

In this work, the control robot system built a scenario in ROS-Gazebo similar to a simulated factory to bridge the gap between the simulated environment and the natural world. Various obstacles are constructed to test the suggested QL navigation algorithm in this environment. Walls, static blocks, movable humans, targets, and mobile robots were all part of the domain. The mobile robot must approach the destination while avoiding static and dynamic impediments, such as those shown in Fig. 4.

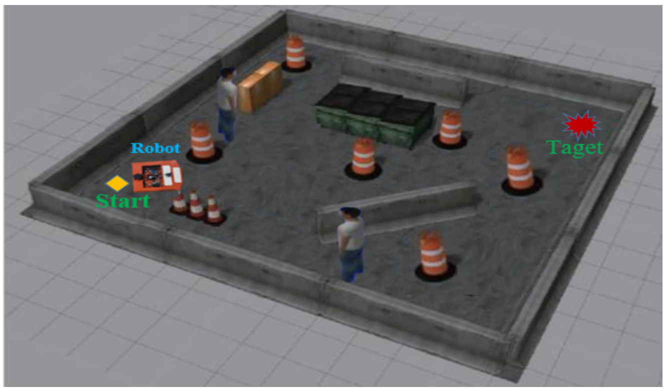
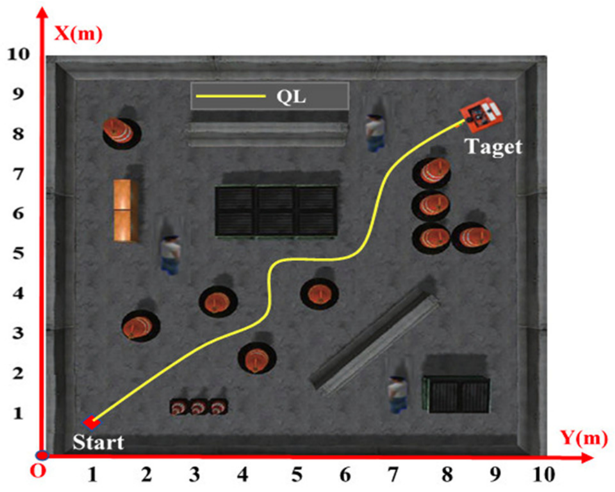


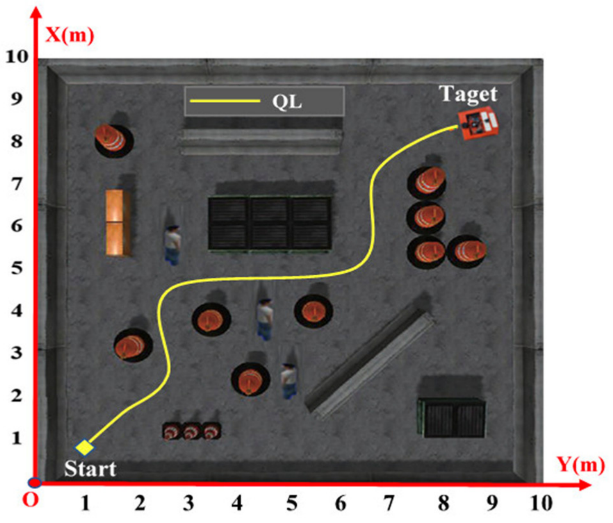
Fig. 4. Table of Q parameters according to the state-action matrix

A robot's training procedure might include numerous cycles. Each cycle terminates when the robot obtains the goal location, encounters an impediment in its route, or when the timer for each cycle runs out. Various obstacles, such as people, static blocks, and walls, were randomly put in this environment to evaluate the efficacy of the proposed mobile robot navigation algorithm. The robot aims for static and dynamic obstacles by maintaining a safe distance from them and reaching the right spots in the least possible time and space.

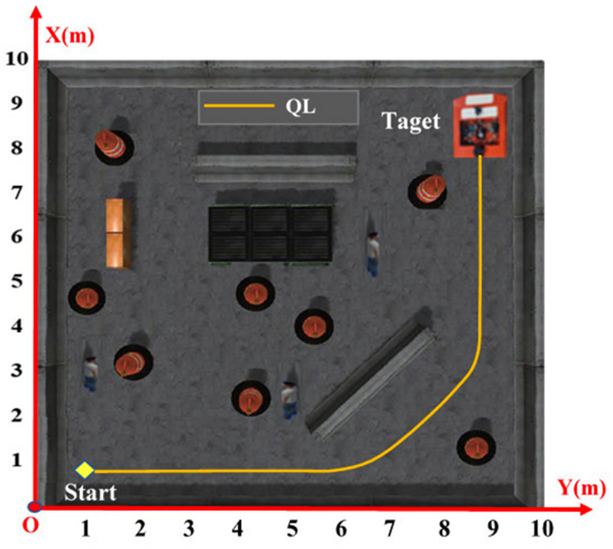
Fig. 5 depicts a realistic environment for route planning for a mobile robot, with a workspace of 12x12m and a minimum distance between blocks of 0.6m. The robot begins at location (1, 1) for all tests and moves to the goal (11, 11). Table 3 summarizes for 4 simulation case.



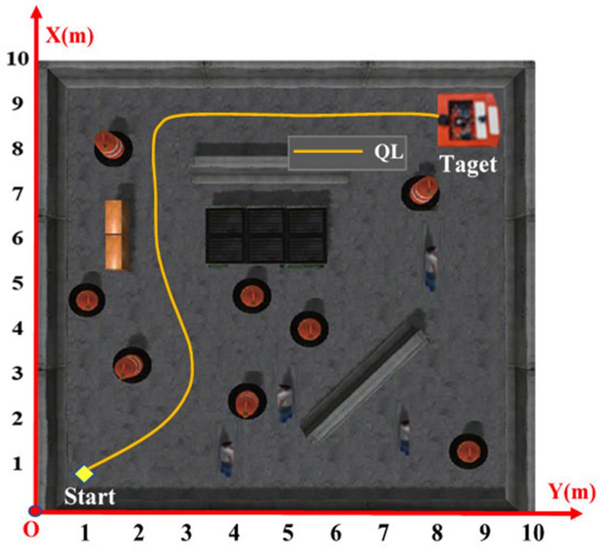
(a) Case 1



(b) Case 2



(c) Case 3



(d) Case 4

Fig. 5. Depicts a realistic environment for route planning for a mobile robot

Table 1. Simulation results on ROS-Gazebo

No	Distance (m)	Run time(s)
Case 1	17.758	12,314
Case 2	18.416	14.637
Case 3	18.690	12,320
Case 4	18.420	14.640

In all 4 cases, simulation results show promising results for the QL algorithm in orbiting planning for mobile robots, especially regarding time. In addition, the simulation results in Table 1 show that QL controllers ensure the mobile robot plans the optimal path. The DQL's trajectory is improved (shorter distance). At the same time, the computation time to establish the optimal rotation and motion is much better. In summary, the results demonstrate the superiority of the QL algorithm in orbiting planning for mobile robots. Not only does it excel in terms of efficiency and speed, but it also guarantees optimal path planning for the mobile robot. The enhancements observed in the trajectory of the DQL further underscore the effectiveness of QL controllers in minimizing distances travelled. These findings highlight the significant computation time improvements for optimal rotation and motion, solidifying the QL algorithm as a top choice for mobile robot orbiting planning.

5. CONCLUSION

This work introduces the QL algorithm as a practical approach for creating paths for mobile robots in complicated and dynamic situations. The algorithm assists the robot in selecting the most suitable action, expedites the robot's learning process, determines the ideal trajectory, and provides the optimal Q value for each pair. Act - Operate within an intricate setting. The simulation results for robots utilizing the QL algorithm have shown the effectiveness and superiority of the proposed method in terms of (1) quickly and safely producing optimal or near-optimal paths, (2) being deterministic and taking only a few milliseconds to calculate a satisfactory solution in terms of length and safety; and (3) QL-based algorithms being non-deterministic and struggling to find a suitable balance between convergence speed and path length. The suggested QL performance has shown remarkable improvement compared to the latest relevant work. Finally, the proposed optimal path for mobile robots was increase.

REFERENCES

[1]. Volos CK, Kyprianidis IM, Stouboulos I N, "A chaotic path planning generator for autonomous mobile robots," *Robots Auton Syst*, 60: 651-656, 2012.

[2]. Chaari I, Koubaa A, Trigui S, et al., "SmartPATH: an efficient hybrid ACO-GA algorithm for solving the global path planning problem of mobile robots," *Int J Adv Robot Syst,* 11: 399-412, 2014.

[3]. MS Gharajeh, HB Jond, ''An intelligent approach for autonomous mobile robot's path planning based on adaptive neuro-fuzzy inference system,'' *Ain Shams Eng. J.*, 2021. DOI: 10.1016/j.asej.2021.05.005.

[4]. C. Zhang, L. Zhou, Y. Li, Y. Fan, "A dynamic path planning method for social robots in the home environment," *Electronics*, 9, 7, 1173, 2020.

[5]. X. Yingqi, S. Wei, Z. Wen, L. Jingqiao, L. Qinhui, S. Han, "A real-time' dynamic path planning method combining artificial potential field method and biased target RRT algorithm," *J. Phys., Conf. Ser*, 1905, 1, 2021, Art. no. 012015.

[6]. B. Yang, J. Yan, Z. Cai, Z. Ding, D. Li, Y. Cao, L. Guo, "A novel heuristic emergency path planning method based on vector grid map," *ISPRS Int. J. Geo-Inf*, 10, 6, 370, 2021.

[7]. S. Xiao, X. Tan, J. Wang, "A simulated annealing algorithm and grid map-based UAV coverage path planning method for 3D reconstruction," *Electronics*, 10, 7, 853, 2021.

[8]. T. Lin, "A path planning method for mobile robot based on A and antcolony algorithms," *J. Innov. Soc. Sci. Res.*, 7, 1, 157-162, 2020.

[9]. Jianming Guo. Liang Liu, Qing Liu, Yongyu Qu, "An Improvement of D* Algorithm for Mobile Robot Path Planning in Partial Unknown Environment," in *2009 Second International Conference on Intelligent Computation Technology and Automation*. ISBN: 978-0-7695-3804-4, DOI: 10.1109/ICICTA.2009.561

[10]. Lai X., Wu D., Wu D., Li JH, Yu H., "Enhanced DWA algorithm for local path planning of mobile robot," *Industrial Robot*, 50, 1, 186-194, 2023 https://doi.org/10.1108/IR-05-2022-0130

[11]. C. Zong, X. Han, D. Zhang, Y. Liu, W. Zhao, M. Sun, "Research on local path planning based on improved RRT algorithm," *Proc. Inst. Mech. Eng*, 235, 8, 2086-2100, 2021.

[12]. Tsai CC, Huang HC, Chan C K., "Parallel elite genetic algo rithm and its application to global path planning for autonomous robot navigation," *IEEE Trans Ind Electron*, 58: 4813-4821, 2011.

[13]. Saska M, Macas M, Preucil L, et al., "Robot path planning using particle swarm optimization of Ferguson splines," in *Proceedings of IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. New York: IEEE Press, 833-839, 2006.

[14]. Raja P, Pugazhenthi S., "On-line path planning for mobile robots in dynamic environments," *Neural Netw World*, 22: 67-83, 2012.

[15]. Chen X, Kong Y, Fang X, et al., "A fast two-stage ACO algorithm for robotic path planning," *Neural Computer Appl*, 22: 313-319, 2013.

[16]. Purcaru C, Precup RE, Iercan D, et al., "Optimal robot path planning using gravitational search algorithm," *Int J Artif Intell*, 10: 1-20, 2013.

[17]. Li P, Duan H B., "Path planning of unmanned aerial vehicle based on improved gravitational search algorithm," *Sci China Technol Sci*, 2012, 55: 2712–2719

[18]. Duan HB, Qiao P X., "Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning," *Int J Intell Comput Cybern*, 7: 24-37, 2014.

[19]. Liu J., Wang Q., He C., Jaffrès-Runser K., Xu Y., Li Z., Xu Y., "QMR: Q-learning based Multiobjective optimization Routing protocol for Flying Ad Hoc Networks," *Computer Communications*, 2019.

[20]. Low ES, Ong P., Cheah KC., "Solving the optimal path planning of a mobile robot using improved Qlearning," *Robotics and Autonomous Systems*, 115, 143-161, 2019.

[21]. Wang YH, Li THS, Lin CJ, "Backward Q-learning: The combination of Sarsa algorithm and Qlearning," *Engineering Applications of Artificial Intelligence*, 26(9), 2184-2193, 2013.

[22]. Das PK, Mandhata SC, Behera HS, Patro SN, "An improved Q-learning algorithm for pathplanning of a mobile robot," *International Journal of Computer Applications*, 51(9), 2012.

THÔNG TIN TÁC GIẢ

Hồ Mạnh Tiến, Võ Thanh Hà

Khoa Điện - Điện tử, Trường Đại học Giao thông Vận tải