RESEARCH ON SELF-PROPELLED ROBOTS CONTROL APPLICATION FOR INTELLIGENT NAVIGATION BASED ON Q-LEARNING ALGORITHM

Tran Thi Huong
University of Economics - Technology for Industries

ARTICLE INFO

Received: 22/3/2022
Revised: 12/5/2022
Published: 19/5/2022

KEYWORDS

Automotive robot
ROS
SLAM
Gazebo
Intelligent navigation

ABSTRACT

This paper presents a study on controlling automotive robots applied in industry, civil, etc. for intelligent navigation in unknown environment on the basis of Q-Learning algorithm. The programming tool is the operating system for the robot ROS (Robot Operating System) and performs automatic intelligent navigation for the robot with the process of locating the robot in a flat environment and mapping (called SLAM-Simultaneous Localization and Mapping). Research results using ROS programming tool, in Gazebo environment. The information is updated from the map, operating environment, control position of the robot, and obstacles to calculate the trajectory for the robot in the automatic navigation system. The goal is to safely avoid the obstacles without encountering any obstacles along the way.

NGHIÊN CỨU ĐIỀU KHIỂN ROBOT TỰ HÀNH ỨNG DỤNG CHO ĐIỀU HƯỚNG THÔNG MINH TRÊN CƠ SỞ THUẬT TOÁN Q-LEARNING

Trần Thị Hường
Trường Đại học Kinh tế - Kỹ thuật Công nghiệp

THÔNG TIN BÀI BÁO

Ngày nhận bài: 22/3/2022
Ngày hoàn thiện: 12/5/2022
Ngày đăng: 19/5/2022

TỪ KHÓA

Robot tự hành
ROS
SLAM
Gazebo
Điều hướng thông minh

TÓM TẮT

Bài báo trình bày nghiên cứu về vấn đề điều khiển robot tự hành ứng dụng trong công nghiệp, trong dân dụng,v.v... để điều hướng thông minh trong môi trường không xác định trên cơ sở thuật toán Q-Learning. Công cụ lập trình là hệ điều hành cho robot ROS (Robot Operating System) và thực hiện điều hướng thông minh tự động cho robot với quá trình định vị robot trong môi trường phẳng và lập bản đồ hóa (gọi là SLAM - Simultaneous Localization and Mapping). Các kết quả nghiên cứu sử dụng công cụ lập trình ROS, trong môi trường Gazebo. Các thông tin được cập nhật từ bản đồ, môi trường hoạt động, vị trí điều khiển của robot và vật cản để tính toán quỹ đạo cho robot trong hệ thống điều hướng tự động. Mục tiêu nhằm tránh các chướng ngại vật một cách an toàn mà không gặp bất kỳ trở ngại nào trên đường đi.

DOI: https://doi.org/10.34238/tnu-jst.5745

Email: huongtt @uneti.edu.vn

1. Mở đầu

Hiện nay trên thế giới cũng như ở Việt Nam nhằm đáp ứng công nghệ 4.0 và vai trò của các hệ thống robot thông minh đang chiếm lĩnh vị trí quan trọng trong công nghiệp, dân dụng. Trong đó, thuật ngữ robot và điều khiển robot ngày nay trở nên thông dụng, đang từng bước gắn chặt với cuộc sống hàng ngày của con người như robot phục vụ (robot hút bụi, robot lau nhà, robot đưa hàng), robot công nghiệp (các robot trong các dây chuyền sản xuất), robot trong y tế, robot trong lĩnh vực quân sự, trong giao thông vận tải,... Kỹ thuật robot nói chung, robot tự hành và robot di động nói riêng là một lĩnh vực đa ngành gồm: cơ khí, Điện - Điện tử, điều khiển tự động và công nghệ thông tin. Đây một lĩnh vực thu hút được nhiều sự chú ý của cộng đồng khoa học bởi vai trò quan trọng của nó trong cuộc sống hàng ngày cũng như trong công việc sản xuất và các dây chuyền tự động tại các nhà máy công nghiệp, nơi sản xuất. Robot tự hành được định nghĩa là một loại xe robot có khả năng tự di chuyển, tự vận động để thực hiện tốt những công việc được giao trong nhà kho, trong nhà máy, nơi sản xuất,... [1]-[5]. Một trong các yêu cầu cơ bản của robot là tự động thực thi các nhiệm vụ và khả năng điều hướng tốt trong phạm vi với các môi trường không xác định. Bằng cách sử dụng những quan sát tích hợp từ thiết bị điều khiển đến môi trường, kết hợp với bản đồ hóa trong cùng một lúc để điều hướng tự động cho robot. Việc đồng thời hóa định vị bản đồ cùng một lúc là một phương pháp chung có liên quan đến việc triển khai một hệ thống robot tự động trong môi trường không xác định cho một robot tự hành đến đích một cách an toàn trong toàn bộ hành trình của nó [6]-[10].

Trong lĩnh vực kỹ thuật điều khiển và công nghệ thông tin, học tăng cường (reinforcement learning) là một lĩnh vực con của học máy, nghiên cứu cách thức một tác nhân trong một môi trường nên chọn thực hiện các hành động nào để cực đại hóa một khoản phần thưởng (reward) nào đó về lâu dài. Các thuật toán học tăng cường cố gắng tìm một chiến lược ánh xạ các trạng thái của môi trường tới các hành động mà tác nhân nên chọn trong các trạng thái đó [3], [5]. Môi trường làm việc để điều khiển cho robot thường được biểu diễn dưới dạng một quá trình quyết định Markop trạng thái hữu hạn (Markov decision process - MDP) và các thuật toán học tăng cường cho ngữ cảnh này có liên quan nhiều đến các kỹ thuật quy hoạch động. Các xác suất chuyển trạng thái và các xác suất thu lợi trong MDP thường là ngẫu nhiên nhưng lại tĩnh trong quá trình của bài toán điều khiển robot. Khác với học có giám sát, trong học tăng cường không có các cặp dữ liệu vào/kết quả đúng, các hành động gần tối ưu cũng không được đánh giá đúng sai một cách tường minh. Hơn nữa, ở đây hoạt động trực tuyến (on-line performance) được quan tâm, trong đó có việc tìm kiếm một sự cân bằng giữa khám phá (môi trường thiết lập bản đồ hóa) và khai thác (tri thức hiện có). Có hai phương pháp thường được sử dụng để giải các bài toán quyết định đó là tìm kiếm trong không gian chiến lược và tìm kiếm trong không gian hàm giá trị hay còn gọi là "phép lặp chiến lược" và "phép lặp giá trị". Hai phương pháp này chính là các giải thuật học tăng cường đặc trưng. Bên cạnh đó, trong những nghiên cứu gần đây các nhà khoa học đề xuất một phương pháp kết hợp giữa hai phương pháp trên, đó chính là phương pháp Actor-Critic learning [7], [8].

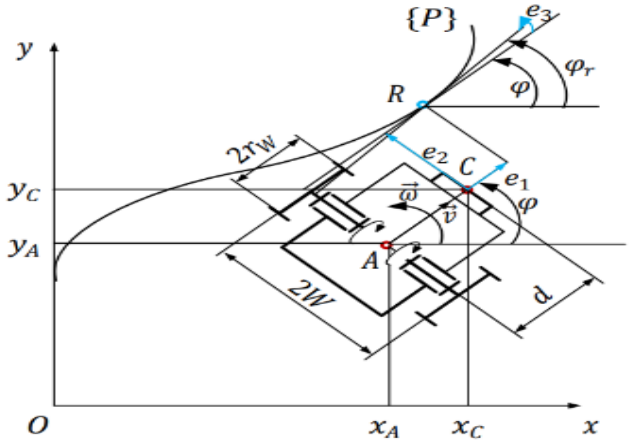
Trong lĩnh vực điều khiển robot, thuật toán Q-learning thuộc nhóm phương pháp dựa trên giá trị (value-based method), nghĩa là chúng tìm cách tính toán hàm giá trị, rồi từ hàm giá trị đưa ra chính sách tối ưu. Dù đây là thuật toán được cho là đơn giản nhưng Q-learning lại là nền tảng của hầu hết các thuật toán học tăng cường quan trọng sau này. Trong một số các nghiên cứu gần đây, ở [6] nghiên cứu về SLAM, ROS, mới chỉ dừng lại với robot Turtlebot vấn đề áp dụng thuật toán một cách máy móc cho bài toán tìm đường ngắn nhất, mà chưa đi sâu vào xử lý các tình huống thực tế trong môi trường làm việc rộng lớn có nhiều yếu tố bất định. Công trình [7] sử dụng thuật toán Q - learning thông thường và đưa ra những hạn chế cơ bản để thực hiện quá trình điều khiển robot, chưa đề cập đến tránh vật cản trên đường đi. Công trình [8] thực hiện quá trình điều hướng trên cơ sở ROS và đã triển khai sử dụng việc tạo bản đồ và định vị đồng thời trên cơ sở (SLAM - Simultaneous Localization and Mapping).

Bài báo nghiên cứu thuật toán Q-learning để nhằm giải quyết bài toán tìm đường đi và tránh vật cản (vật cản cố định, vật cản di động) cho robot trong môi trường làm việc và không gian hành động thực tế cho robot tự hành nhằm điều hướng thông minh, thiết lập kế hoạch đường đi và tránh chướng ngại vật an toàn mà không xảy ra bất kỳ va chạm nào.

2. Nội dung nghiên cứu

2.1. Xây dựng mô hình điều khiển cho robot tự hành

Cấu trúc và các thiết bị phần cứng được sử dụng minh họa như hình 1, bao gồm các khối và chức năng như sau: khung robot được thiết kế theo kiểu hình tròn và khoảng cách giữa hai bánh xe là 0,35 m và bán kính bánh xe là 0,065 m.



Hình 1. *Mô hình robot tự hành ba bánh*

Phương trình động học của thiết bị dẫn động của hai động cơ chính (TBDĐ) viết tại điểm A là trung điểm của đoạn thẳng nối tâm hai bánh dẫn động là:

$$v = v_A = \frac{(\omega_R - \omega_L) r_W}{2} \tag{1}$$

$$\omega = \frac{(\omega_R - \omega_L) r_W}{2W} \tag{2}$$

Từ (1) và (2), ta có các phương trình chuyển động của TBDĐ viết trong hệ tọa độ tại điểm *A* và điểm *C* (tâm cảm biến dò đường):

$$\begin{bmatrix} \dot{x}_A \\ \dot{y}_A \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} cos\varphi & 0 \\ sin\varphi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \dot{x}_A - \dot{\varphi}.d.sin\varphi \\ \dot{y}_A + \dot{\varphi}.d.cos\varphi \\ \omega \end{bmatrix} \tag{4}$$

Các bộ điều khiển bám qũy đạo đường đi robot theo tiêu chuẩn ổn định của Lyapunov [1], [2] tuy đáp ứng tốt trong mô phỏng nhưng gặp hạn chế trong thực tế vì cảm biến dò line chỉ đo được độ lệch ngang (hay e_2). Chính vì thế, bài báo trình bày bộ điều khiển thiết kế theo dạng hồi tiếp tuyến tính [2], [5]. Phương trình của sai số dò line như sau:

$$\begin{bmatrix}\dot{e}_1\\ \dot{e}_2\\ \dot{e}_3\end{bmatrix}=\begin{bmatrix}v_r\cos e_3\\ v_r\sin e_3\\ \omega_r\end{bmatrix}+\begin{bmatrix}-1 & e_2\\ 0 & -d-e_1\\ 0 & -1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} \quad (5)$$

Khi TBDĐ bám theo line, quỹ đạo của TBDĐ chỉ dao động xung quanh line, do đó, điểm cân bằng của hệ phi tuyến ở phương trình (5) chính là $X_0=\begin{bmatrix}e_{10} & e_{20} & e_{30}\end{bmatrix}^T=\begin{bmatrix}000\end{bmatrix}^T$ với đầu vào sơ khởi là $u_{sk0}=\begin{bmatrix}v_r & \omega_r\end{bmatrix}^T$. Tuyến tính hóa (5) xung quanh điểm cân bằng ta được:

$$\begin{bmatrix}\dot{e}_1\\ \dot{e}_2\\ \dot{e}_3\end{bmatrix}=\begin{bmatrix}0 & -\omega_r & -v_r\\ -\omega_r & 0 & v_r\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}e_1\\ e_2\\ e_3\end{bmatrix}+\begin{bmatrix}-1 & 0\\ 0 & -d\\ 0 & -1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} \quad (6)$$

Trong thực tế, vận tốc TBDĐ quá trình hoạt động là không đổi nên $e_1\approx 0$, khi đó, (6) thành:

$$\begin{bmatrix}\dot{e}_2\\ \dot{e}_3\end{bmatrix}=\begin{bmatrix}0 & v_r\\ 0 & 0\end{bmatrix}\begin{bmatrix}e_2\\ e_3\end{bmatrix}+\begin{bmatrix}-d\\ -1\end{bmatrix}u, u=\omega \quad (7)$$

Đạo hàm phương trình thứ nhất của (7) và thế phương trình thứ hai vào ta được:

$$\ddot{e}_2=-d\times\dot{u}-v_r\times u \quad (8)$$

Đặt: $x_1=e_2, x_2=\dot{x}_1-\beta.u$ với $\beta=-d$, ta có:

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}-d\\ -v_r\end{bmatrix}u \quad (9)$$

Phương trình (9) có dạng $\dot{X}=A.X+B.U$ và ma trận điều khiển được *M = [B AB]* có $\det(M)\neq 0\ \forall\ v_r\neq 0$ nên hệ (9) là hệ điều khiển được. Đặt luật điều khiển hồi tiếp *u = -K.X* với *K = [k₁ k₂]*. Khai triển luật điều khiển ta được:

$$u=\frac{-k_1}{1+k_2d}e_2+\frac{-k_2}{1+k_2d}\dot{e}_2 \quad (10)$$

Luật điều khiển (10) có dạng luật điều khiển PD, khi đó ta có: $u=K_pe_2+K_d\dot{e}_2$ với $K_p=\frac{k_1}{1+k_2d}, K_d=\frac{-k_2}{1+k_2d}$. Các hệ số *k₁, k₂* được xác định dựa vào phương pháp tọa độ điểm cực - vị trí hoặc điều khiển tối ưu LQR. Tâm vận tốc tức thời của TBDĐ với robot tại thời điểm (*i*) lần lượt là $C_{0(i)}$ và $C_{A(i)}$. Góc lệch giữa thiết bị dẫn động và robot tại thời điểm (*i*) là:

$$\theta_{(i)}=\varphi_{(i)}-\varphi_{A_{(i-1)}} \quad (11)$$

Vì TBDĐ và robot liên kết với nhau qua khớp bản lề tại O nên $v_D=v_A=v_0=v$, suy ra bán kính cua và vận tốc góc của robot tự hành tại thời điểm (*i*) là:

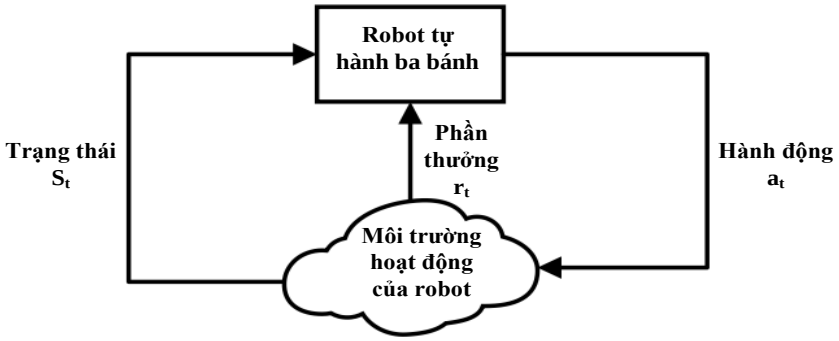
$$C_AO_{(i)}=\frac{L}{sin(\theta_{(i)})} \quad (12)$$

$$\omega_{A(i)}=\frac{v_{(i)}}{C_AO_{(i)}}=\frac{v_{(i)}sin(\theta_{(i)})}{L} \quad (13)$$

2.2. Ứng dụng thuật toán Q-learning để điều khiển robot tự hành

2.2.1. Học tăng cường với thuật toán Q-learning

Phương pháp học tăng cường với thuật toán Q-learning là một nhánh của học máy được phát triển để phục vụ cho việc tính toán thông minh cho lĩnh vực khoa học kỹ thuật nói chung và trên phương diện điều khiển học nói riêng.



Hình 2. *Sơ đồ tương tác với môi trường học tập của robot tự hành*

Với Q-learning nói riêng và học tăng cường nói chung, thì mọi thứ được chia thành "trạng thái - s_t" và "hành động - a_t" với thời gian được biểu thị bằng một chuỗi các bước thời gian (t = 0, 1, 2, .v.v.). Đối với môi trường làm việc liên tục như điều khiển robot tự hành thì việc đầu tiên cần làm là lượng tử hóa không gian trạng thái để có cập nhật $S=\left\{S_1,S_2,...S_m\right\}$ và lượng tử hóa được không gian hành động thành tập $A=\left\{a_1,a_2,...a_n\right\}$, kết quả là môi trường tạo ra phần thưởng $r_t=r(s_t,a)\in R$, để hiểu rõ hơn ta có sơ đồ *tương tác môi trường* học tập như hình 2.

Khi đó, cách Q-learning hoạt động là tính toán và lưu giữ giá trị Q trên một hành động và trạng thái cụ thể, Q(s, a). Tất cả những thông tin, kinh nghiệm tích lũy từ những lần tính toán trước đó sẽ được mã hóa thành một bảng đánh giá.

Chúng ta tính toán tổng phần thưởng thu được sau thời gian t là R_t được hoàn trả về như sau:

$$R_t=\sum_{k=0}^{\infty}\gamma^k r_{t+k} \tag{14}$$

trong đó, $0\le\gamma<1$ là hệ số khấu trừ cho các phần thưởng. Giá trị của γ càng nhỏ thì phần thưởng càng được chú trọng trong khi thực hiện hành động. Khi đó, hàm giá trị hành động (hàm Q) được xác định như sau:

$$Q^n(s,a)=E_\pi\left\{R_t\middle|s_t=s,a_t=a\right\} \tag{15}$$

trong đó, $E_\pi\{...\}$ là đại diện cho kỳ vọng theo chính sách ngẫu nhiên trong không gian hành động. Hàm $Q^n(s,a)$ đại diện cho tổng phần thưởng chiết khấu dự kiến khi ta chọn hành động a dưới trạng thái s và sau đó chọn hành động theo chính sách π. Hàm Q được mô tả dưới dạng công thức đệ quy như sau:

$$Q^\pi(s,a)=\sum_{s'\in S}\Pr(s'|s,a)r.\left((s,a,s')+\gamma\sum_{a'\in A}\pi(a'|s')Q^\pi(s',a')\right) \tag{16}$$

trong đó, S và A lần lượt là tập trạng thái và tập hành động. Từ công thức này, chúng ta có thể xác định rằng hàm Q theo chính sách tối ưu π^*, tức là hàm Q tối ưu, thỏa mãn phương trình sau, được gọi là phương trình tối ưu Bellman:

$$Q^*(s,a)=E_{s'}\left\{r_t+\gamma\max_{a'}Q^*(s',a')\right\} \tag{17}$$

Trong thuật toán Q-learning, bằng cách cập nhật lặp đi lặp lại hàm Q sử dụng ở biểu thức (10)

dựa trên dữ liệu thực nghiệm, hàm Q hội tụ ngẫu nhiên thành $Q^*(s,a)$ và do đó, chính sách tối ưu có thể được xác định là chính sách tham vọng của $Q^*: a^* = argmax_a(s,a)$. Trong thực tế, tác nhân học tập của robot khi di chuyển phải khám phá môi trường hành động vì hàm Q không đáng tin cậy và cần phải lựa chọn hành động để được sử dụng một cách rộng rãi như một chính sách ngẫu nhiên, khi đó cho phép để chọn một hành động có xác suất cho một trạng thái đầu vào s. Cụ thể hơn, chính sách μ sẽ tham gia lựa chọn một hành động nhằm tối đa hóa hàm Q ở trạng thái s với xác suất a của 1- μ, μ ∈ [0, 1] và cho phép lựa chọn một hành động ngẫu nhiên với xác suất còn lại. Khi các trạng thái và hành động là rời rạc và khác nhau, một cách đơn giản để biểu diễn hàm Q là sử dụng như một bảng giá trị cho tất cả cặp trạng thái, hành động như sau:

$$Q(s,a) \leftarrow Q(s,a) + \alpha.\left((r + \gamma \max_{a'} Q(s',a') - Q(s',a')\right) \quad (18)$$

Trong đó, $0 < \alpha \leq 1$ là tốc độ học và tốc độ học càng lớn thì dữ liệu mới cập nhật càng nhanh. Với thuật toán này, bảng Q hội tụ đến hàm Q tối ưu trong điều kiện hội tụ của xấp xỉ ngẫu nhiên. Mặt khác, vì điều này dựa trên phương pháp xấp xỉ ngẫu nhiên, nên cần có một số lượng dữ liệu thích hợp cho tất cả các cặp (s, a).

Trong phương pháp Q-learning dạng bảng, khi số lượng phần tử trong trạng thái hoặc không gian hành động là rất lớn hay trạng thái hoặc không gian hành động là liên tục, chúng ta thường biểu diễn hàm Q dưới dạng hàm tham số Q(s, a; θ) bằng cách sử dụng các tham số θ và sau đó cập nhật các thông số theo biểu thức gradient như sau:

$$\theta \leftarrow \theta + \alpha\left(target_Q - Q(s,a;\theta)\right)\nabla_\theta Q(s,a;\theta) \quad (19)$$

Ở đây, "$target_Q$" là giá trị mục tiêu dựa trên phương trình Bellman tối ưu (10) và nó được tính toán như sau:

$$target_Q\, r(s,a;s') + \gamma \max_{a'} Q(s',a';\theta) \quad (20)$$

Hàm Q được cập nhật theo một trình tự nhất quán của nó. Thuật toán Q-learning là một phương pháp dựa trên các hàm giá trị và từ hàm giá trị đưa ra chính sách tối ưu, trong đó giá trị xấp xỉ hàm Q được hồi quy về giá trị mục tiêu, giá trị này phụ thuộc vào chính nó. Điều này ngụ ý rằng giá trị đích thực thay đổi tự động khi luật học tập được cập nhật. Do đó, khi một hàm phi tuyến tính, chẳng hạn như mạng nơ-ron được sử dụng để xấp xỉ p, quá trình học tập này trở nên không ổn định do các thay đổi động học trong mục tiêu và trong trường hợp xấu nhất, hàm Q sẽ phân kỳ [3], [4].

2.2.2. Điều hướng thông minh robot tự hành sử dụng thuật toán Q-learning

Trong thuật toán Q-learning giá trị của các vị trí điều khiển điều hướng cho robot thường được cập nhật theo phương pháp vi phân tức thời, sử dụng sai lệch giữa một bước lặp để ước lượng, tính toán hàm giá trị Q theo các biểu thức (18) ở trên. Khi gặp các bài toán điều hướng cho robot di chuyển với nhiều trạng thái khác nhau và hành động dịch chuyển (sang trái, sang phải), tránh chướng ngại vật (vật cản di động, vật cản cố định), .v.v. Khi đó ta chọn α = 0,1; γ = 0,95; lúc này robot di chuyển với nhiều tình huống khác nhau, lúc này quá trình cập nhật bảng Q được thực hiện.

Khi bắt đầu huấn luyện thuật toán, đối tượng sẽ đi một hoặc hai lần sang phải, nhưng ngay khi hành động sang trái được chọn, thì hành động này sẽ tiếp tục được chọn ở những lần di chuyển tiếp theo bởi vì luôn nhận được phần thưởng khi thực thi hành động sang trái này.

Mục tiêu của mô hình trong quá trình điều hướng cho robot là giữ cho nó trong một giới hạn cho phép, tức là ± 5 độ. Lúc đầu, mô hình robot, ma trận Q, chính sách π sẽ được khởi tạo. Có một số điểm quan trọng để thực hiện điều hướng trong quá trình di chuyển, như các trạng thái không hữu hạn. Trong phạm vi giới hạn, có thể có hàng trăm và hàng nghìn góc cao độ và có

hàng nghìn cột là không thể xảy ra khi cập nhật thuật toán Q-learning. Vì vậy, ta đã sắp xếp các giá trị trạng thái thành 20 góc trạng thái từ -10 độ đến 10 độ. Đối với giá trị hành động, chúng ta đã chọn mười vận tốc khác nhau và chúng là [- 200, - 100, - 50, - 25, - 10, 10, 25, 50, 100, 200] ms^{-1}. Ma trận Q có 20 cột, mỗi cột đại diện cho một trạng thái và mười hàng mỗi hàng đại diện cho mọi hành động. Ban đầu, các giá trị Q được giả định là 0 và một số hành động ngẫu nhiên được chỉ định cho mọi trạng thái trong chính sách π. Chúng ta đã huấn luyện trong 1400 tập, mỗi tập có 2000 lần lặp lại. Vào đầu mỗi tập dạy đều được mô phỏng được làm mới. Bất cứ khi nào trạng thái của robot vượt quá giới hạn, nó sẽ bị phạt bằng cách gán phần thưởng cho -100. Bảng Q được cập nhật ở mỗi bước theo biểu thức (18). Từ đó ta có thuật toán thiết lập quỹ đạo điều hướng tự động cho robot được hiển thị như hình 3.

Thuật toán Q-learning thực hiện các tác nhân hành động cho việc điều hướng tự động thông minh cho robot tự hành nhằm thực hiện các quỹ đạo để tránh các vật cản động cũng như vật cản tĩnh trong quá trình di chuyển của robot, đồng thời tính ra quỹ đạo ngắn nhất cho robot di chuyển đến đích với một đường đi nhanh nhất.

Algorithm: Q Learning Algorithm as applied in the system

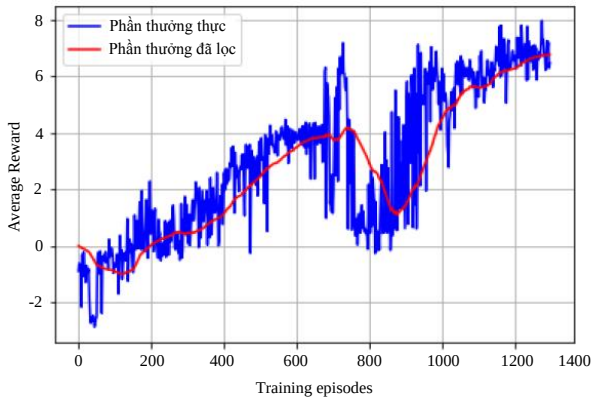
```
Initialize Robot;
Initialize Q Matrix Q;
Initialize Policy π;
Initialize Penalty Reward pen;
for number of episodes do
    Reset simulation ;
    Wait for 1-second ;
    Pause simulation ;
    Read the pitch angle φ of the robot ;
    state ← φ ;
    Unpause simulation ;
    for number of iterations do
        Generate a random number rand;
        if rand ≤ μ then
            take random action ;
        end
        else
            take action based on π ;
        end
        state_new ← φ;
        Pause simulation;
        if absolute value of state_new ≥ limit then
            if reward_total ≤ Target then
                reward ← pen;
                Update Q ;
                Update π;
            end
            Break ;
            else
                Print Passed ;
                Break ;
            end
        end
        else
            reward ← 1;
            Update Q;
            Update π state ← state_new
        end
    end
end
```

Hình 3. *Thuật toán Q-learning cho robot tự hành ba bánh*

3. Kết quả mô phỏng

Để thực hiện quá trình điều hướng thông minh cho robot tự hành, tác giả thực hiện trên cơ sở

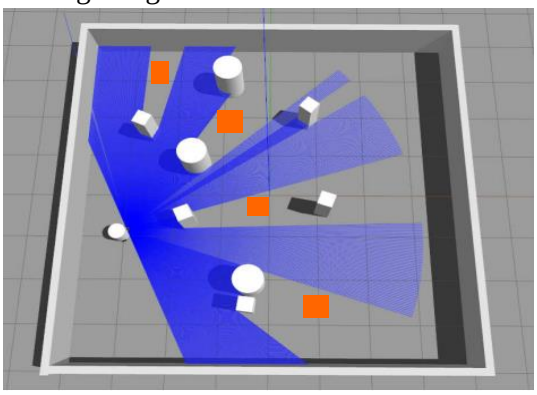
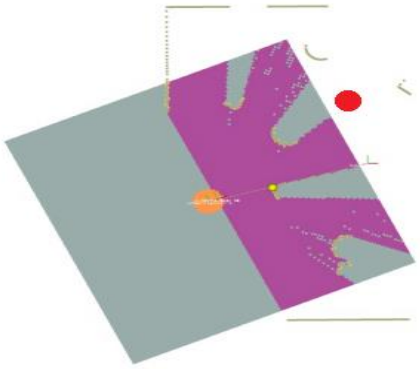
thuật toán đã nghiên cứu và đề xuất ở phần hai như trên tiến hành nghiên cứu một số mô phỏng được dựa trên công cụ nghiên cứu mạnh mẽ là Gazebo.



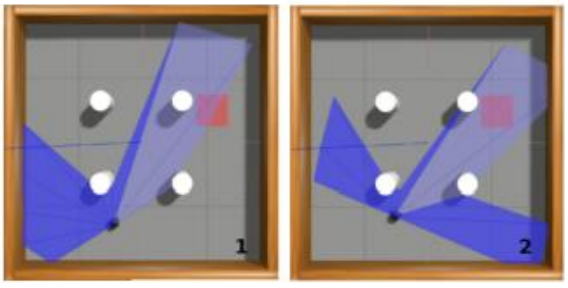
Hình 4. *Kết quả phần thưởng trung bình của quá trình học tập*

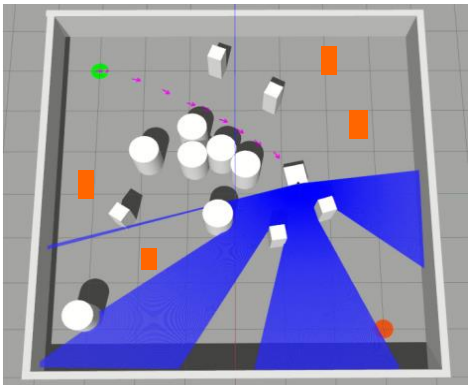
Hình 4 trình bày kết quả học tập trong môi trường mô phỏng Gazebo. Ta có thể thấy, phần thưởng trung bình (phần thưởng thực) của robot trong mỗi tập dạy không ngừng tăng lên khi quá trình đào tạo tiếp tục. Robot sẽ học được kiến thức về môi trường thông qua việc tương tác với môi trường. Cuối cùng, robot có thể điều hướng đến đích một cách nhanh chóng và tự chủ trong cả môi trường đơn giản và phức tạp mà không có bất kỳ những va chạm nào với các chướng ngại vật. Nghiên cứu thử nghiệm cho thấy tính hiệu quả của mô hình mà tác giả đã đề xuất.

Trong phần này, một số mô phỏng được thực hiện dựa trên công cụ mô phỏng mạnh mẽ và môi trường trong Gazebo.

a)  b) 

Hình 5. *a) Môi trường đào tạo Gazebo; b) Bản đồ hóa tỷ lệ tương ứng được thực hiện trong Rviz*

a) 

b) 

Hình 6. a) *Một số hành động được đào tạo trong môi trường Gazebo, b) Xây dựng bản đồ trực quan và đường đi của robot trong môi trường Gazebo*

Như trong hình 5 cho thấy bản đồ hóa đồng thời được xây dựng trên Gazebo và trong Rviz là bản đồ được tạo ra với các bức tường nghiêm ngặt và một robot tự hành có thể được thực hiện điều khiển để di chuyển xung quanh các chướng ngại vật cố định tạo bối cảnh cho các chuyển động của robot được sử dụng để xây dựng bản đồ hành động của robot tự hành được điều hướng một cách thông minh.

Chương trình được xây dựng cho quá trình điều hướng thông minh như hình 6, thể hiện được một số các hành động được thực hiện bởi robot từ vị trí ban đầu cho đến khi thực hiện được hành động có thể đến mục tiêu sau các đợt học tập. Vấn đề này thể hiện mối quan hệ đặc biệt quan trọng, khi thực hiện vợt các chướng ngại vật tĩnh hay vật cản động trên đường đi mà không gây ra bất kỳ trở ngại nào, khi đó robot về đích an toàn.

Trên hình 6b đường màu tím (mũi tên nét đứt) thể hiện đường đi của robot khi tránh chướng ngại vật được tạo bởi cảm biến thông minh, camera thông minh và vị trí hiện tại của robot được cập nhật (robot tự hành được ký hiệu bằng màu xanh lá cây) bằng cách sử dụng kích thước đo hình học. Đây là một công cụ trực quan có thể cung cấp cập nhật trực tiếp các bản đồ được tạo từ thuật toán SLAM để điều khiển robot. Hơn nữa, quỹ đạo chuyển động của robot tự hành đa hướng trong bản đồ cũng có thể tự động điều hướng và trong môi trường này luôn có thể tạo ra các chướng ngại vật, như trong hình 5; hình 6 để cho robot di chuyển. Kết quả cho thấy robot đã lập quỹ đạo chuyển động và di chuyển đến đúng mục tiêu mong muốn một cách chính xác và an toàn. Những kết quả này mang lại lợi ích thực tiễn cao, có thể thực hiện hầu hết trên các robot trong công nghiệp, robot di động trong giao thông, robot tự hành trong y tế và robot tự hành trong các nhà máy xí nghiệp công nghiệp.

4. Kết luận

Nội dung bài báo đã trình bày việc điều khiển robot tự hành ba bánh ứng dụng điều hướng thông minh trong môi trường phẳng không xác định, sử dụng công cụ là ROS để lập trình điều khiển. Các kết quả mô phỏng trên phần mềm Gazebo chứng minh khả năng của robot tự hành điều hướng tự động đến các vị trí mục tiêu mong muốn và tránh được các vật cản tĩnh và vật cản động trong quá trình di chuyển trong môi trường đơn giản và phức tạp. Nghiên cứu này đã cho thấy tính hiệu quả thực tế của quá trình điều khiển robot tự hành và thực hiện điều hướng tự động cho robot mà tác giả đã nghiên cứu có khả năng định vị robot trong môi trường, lập bản đồ 2D và thực hiện điều hướng thông minh để đến mục tiêu trong bản đồ đã xây dựng. Kết quả này cho thấy robot đã xây dựng được quỹ đạo chuyển động, di chuyển đến đúng mục tiêu và tự động tránh vật cản động xuất hiện trên đường đi. Hướng phát triển của vấn đề nghiên cứu là mong muốn sẽ được thực hiện áp dụng trên một số loại robot tự hành thực tế trong nhà máy sản xuất công nghiệp, trong đời sống, trong giao thông thông minh và trong y học với những thuật toán tối ưu hơn của học máy.

TÀI LIỆU THAM KHẢO/ REFERENCES

[1] Q. C. Hoang, H. V. Dao, A. V. Nguyen, and B. C. Le, *Electric drive systems in Robots*. People's Army Publishing House, (in Vietnamese), Hanoi, Vietnam, 2020.

[2] D. P. Nguyen, *Advanced Control Theory*. Science and Engineering Publishing House, (in Vietnamese), Hanoi, Vietnam, 2018.

[3] H. S. Le, C. D. Le, and V. H. Nguyen, *Industrial Robots Syllabus*. Ho Chi Minh City National University Publishing House, (in Vietnamese), Ho Chi Minh City, Vietnam, 2017.

[4] T. T. Nguyen, *Basic Deep Learning*, 2nd, The Legrand Orange Book Template by Mathias Legrand, Publishing by Vietnamme, (in Vietnamese), Hanoi, Vietnam, 2020.

[5] L. Joseph and J. Cacace, *Mastering ROS for Robotics Programming*. Second Edition: Design, build, and simulate complex robots using the Robot Operating System, Published by Packt Publishing Ltd. N0. 35 Livery Street Birmingham B3 2PB, UK, 2018.

[6] S. P. Thale *et al.*, "ROS based SLAM implementation for Autonomous navigation using Turtlebot," ITM Web of Conferences 32, 01011, 2020.

[7] S. Ohnishi, E. Uchibe, Y. Yamaguchi, K. Nakanishi, Y. Yasui, and S. Ishii, "Constrained Deep Q-Learning Gradually Approaching Ordinary Q-Learning," *Frontiers in Neurorobotics Journal*, vol. 13, pp. 7-12, 2019.

[8] R. K. e. a. Megalingam, "*ROS based autonomous indoor navigation simulation using SLAM algorithm*," *Int. J. Pure Appl.*, vol. 118, no. 7, pp. 199-205, March 2018.

[9] H. X. Dong, C. Y. Weng, C. Q. Guo, H. Y. Yu, and I. M. Chen, "Real-time avoidance strategy of dynamic obstacles via half model-free detection and tracking with 2D Lidar for mobile robots," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 4, pp. 2215-2225, Aug 2021.

[10] D. Kozlov, "Comparison of Reinforcement Learning Algorithms for Motion Control of an Autonomous Robot in Gazebo Simulator," *International Conference on Information Technology and Nanotechnology, IEEE Explore*, vol .9, pp. 1-5, 2021, doi: 10.1109/ITNT52450.2021.9649145.