BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

VŨ XUÂN THẮNG

NGHIÊN CỨU ĐIỀU KHIỂN ROBOT DI ĐỘNG CHỊU NHIỄU BẤT ĐỊNH DỰA TRÊN HỌC TĂNG CƯỜNG

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT CƠ ĐIỆN TỬ

Hà Nội – 2025

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

VŨ XUÂN THẮNG

NGHIÊN CỨU ĐIỀU KHIỂN ROBOT DI ĐỘNG CHỊU NHIỄU BẤT ĐỊNH DỰA TRÊN HỌC TĂNG CƯỜNG

Ngành: Kỹ thuật cơ điện tử
Mã số:14 8520114

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT CƠ ĐIỆN TỬ

NGƯỜI HƯỚNG DẪN:

1. TS. Nguyễn Văn Trường

Hà Nội – 2025

LỜI CAM ĐOAN

Tôi là Vũ Xuân Thắng, học viên lớp Cao học cơ điện tử khóa 12, trường Đại học Công nghiệp Hà Nội. Tôi xin cam đoan đề án thạc sĩ "Nghiên cứu điều khiển robot di động chịu nhiễu bất định dựa trên học tăng cường" là công trình nghiên cứu của tôi dưới sự hướng dẫn của TS. Nguyễn Văn Trường.

Tôi xin cam đoan các số liệu và kết quả nêu trong Đề án là trung thực, không sao chép từ công trình nghiên cứu khác. Các tài liệu tham khảo được sử dụng hợp lý và minh bạch, nội dung trích dẫn đều được nêu trong các tài liệu có nguồn gốc rõ ràng, uy tín.

Tôi cam kết chịu hoàn toàn trách nhiệm và chấp nhận mọi hình thức kỷ luật theo quy định liên quan đến tính chính xác và trung thực của đề án nghiên cứu này.

Tác giả

Vũ Xuân Thắng

LỜI CẢM ƠN

Đầu tiên tôi xin gửi cảm ơn chân thành nhất tới TS. Nguyễn Văn Trường - giảng viên hướng dẫn trực tiếp của tôi. Cảm ơn thầy vì đã luôn tận tụy giúp đỡ và cho tôi những lời khuyên quý báu không chỉ về chuyên môn mà còn về cuộc sống.

Tôi cũng xin cảm ơn các thầy cô trường Đại học Công nghiệp Hà Nội đã truyền đạt cho tôi những kiến thức về chuyên ngành Kỹ thuật Cơ điện tử trong suốt thời gian học tập để tôi có được nền tảng kiến thức quá trình làm đề án thạc sĩ.

Cuối cùng, xin gửi lời cảm ơn đến gia đình đã luôn động viên tôi trong suốt quá trình học tập, cảm ơn bạn bè vì đã hỗ trợ tôi trong quá trình nghiên cứu và viết đề án này.

Xin chân thành cảm ơn

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

DANH MỤC TỪ VIẾT TẮT

Viết tắt	Tiếng Anh	Tiếng Việt
RL	Reinforcement learning	Học tăng cường
RMSE	Root mean square error	Sai số căn bậc hai bình phương trung bình
GD	Gradient decent	Giảm độ dốc
PID	Proportional Integral Derivative	Vi tích phân tỉ lệ
SMC	Sliding mode control	Phương pháp điều khiển trượt
RBF	Radial basis function	Hàm cơ sở xuyên tâm
TD3	Twin delayed DDPG	DDPG trì hoãn đôi
DDPG	Deep deterministic policy gradient decent	Học tăng cường chính sách xác định

MỞ ĐẦU

Lý do chọn đề tài

Trong những năm gần đây, robot di động đã trở thành một thành phần quan trọng trong nhiều ứng dụng thực tiễn, bao gồm sản xuất công nghiệp, vận chuyển thông minh, chăm sóc y tế, etc. Tuy nhiên, một trong những thách thức lớn đối với việc điều khiển robot di động là khả năng duy trì hiệu suất hoạt động ổn định trong môi trường thay đổi liên tục, vật cản động và chứa nhiều nhiễu bất định, có thể kể đến như sai số cảm biến [1], ma sát không xác định, hoặc sự thay đổi động lực học do tác động từ môi trường bên ngoài [2]. Nghiên cứu này được phát triển bộ điều khiển cho robot di động với tầng điều khiển vị trí dựa trên học tăng cường và tầng điều khiển chấp hành dựa trên phương pháp điều khiển thời gian cố định thích nghi sử dụng mạng nơ ron. Từ đó đảm bảo robot hoạt động trong các điều kiện vật cản trong môi trường thay đổi liên tục, và chịu tải không biết trước.

Lịch sử nghiên cứu

Vấn đề điều khiển robot di động là một bài toán đã được nghiên cứu từ những năm 1960, Một trong những ví dụ ban đầu đáng chú ý nhất là Shakey the Robot, được phát triển tại SRI International, đây là robot đầu tiên có khả năng suy luận tượng trưng và lập kế hoạch cấp cao [3].

Trong những năm 1980 và 1990, điều hướng robot di động bắt đầu kết hợp mô hình động học, điều khiển phản hồi và thuật toán lập kế hoạch đường đi toàn cầu. Các kỹ thuật như điều khiển PID, tuyến tính hóa phản hồi và ổn định dựa trên Lyapunov đã được áp dụng cho các nhiệm vụ theo dõi quỹ đạo [4]. Đối với việc lập kế hoạch đường dẫn, các thuật toán như A* và Dijkstra đã được giới thiệu cho các môi trường dựa trên lưới, trong khi các phương pháp

dựa trên lấy mẫu như phương pháp lộ trình xác suất (PRM) và cây ngẫu nhiên khám phá nhanh (RRT) cho phép lập kế hoạch trong các không gian cấu hình có nhiều chiều [5].

Ngược lại với các phương pháp dựa trên mô hình, các kiến trúc dựa trên hành vi đã xuất hiện để cho phép phản ứng thời gian thực mà không yêu cầu các mô hình môi trường hoàn chỉnh. Kiến trúc bao hàm do Brooks đề xuất nhấn mạnh vào khả năng kiểm soát theo lớp, trong đó các hành vi cấp cao hơn có thể ghi đè lên các hành vi cấp thấp hơn [6]. Điều hướng được phân tách thành các hành vi nguyên thủy như tránh chướng ngại vật, bám theo tường và tìm kiếm mục tiêu. Các thuật toán như biểu đồ trường vectơ (VFH) và phương pháp tiếp cận cửa sổ động (DWA) cho phép kiểm soát phản ứng nhanh và hiệu quả trong các môi trường động [7], [8].

Với sự phát triển của robot xác suất, trọng tâm chuyển sang xử lý sự không chắc chắn trong nhận thức và chuyển động. Định vị và lập bản đồ đồng thời (SLAM) đã trở thành một vấn đề trung tâm, với các giải pháp như SLAM bộ lọc Kalman mở rộng (EKF-SLAM) và FastSLAM giới thiệu các phương pháp có thể mở rộng để xây dựng bản đồ trong khi ước tính tư thế của robot [9]. Các kỹ thuật định vị như định vị Monte Carlo (MCL) đã cải thiện tính mạnh mẽ trong môi trường động. Các phương pháp này cho phép robot điều hướng các không gian chưa biết với thông tin trước tối thiểu.

Những tiến bộ gần đây đã kết hợp học sâu và học tăng cường (RL) vào các hệ thống dẫn đường, cho phép rô-bốt học các chính sách điều khiển từ dữ liệu cảm biến thô. Gradient chính sách xác định sâu (DDPG) và các thuật toán tương tự đã cho phép học trong các không gian hành động liên tục phù hợp với điều khiển rô-bốt trong thế giới thực [10]. Các phương pháp đầu cuối học cách ánh xạ đầu vào

cảm biến (ví dụ: hình ảnh camera, quét lidar) trực tiếp để điều khiển đầu ra, cho phép vận hành tự động trong môi trường không đầy đủ thông tin [11].

Mục đích nghiên cứu của đề án tốt nghiệp.

Mục tiêu của luận văn này là đề xuất một phương pháp điều khiển robot di động chịu nhiễu bất định (vật cản động, thay đổi tham số động học hệ thống):

- Mô hình hóa phương trình động học của robot di động dạng vi sai
- Thiết kế bộ điều khiển điều hướng cho robot dựa trên học tăng cường
- Mô hình hóa phương trình động lực học cho cơ cấu chấp hành động cơ của robot di động
- Thiết kế bộ điều khiển cơ cấu chấp hành thời gian định trước thích nghi dựa trên mạng RBF
- Triển khai hệ thống trên mô phỏng để kiểm tra tính khả thi của phương pháp
- So sánh đánh giá sự hiệu quả của bộ điều khiển với các phương pháp hiện có

Đối tượng, phạm vi nghiên cứu

Đối tượng nghiên cứu chính của đề án là hệ robot di động hoạt động trong môi trường chịu nhiễu bất định như vật cản động, thiếu hụt thông tin bản đồ địa phương, robot chịu tải trọng thay đổi hoặc không biết trước.

Tóm tắt cô đọng các luận điểm cơ bản và đóng góp mới

Đề án này nghiên cứu xây dựng phương pháp điều khiển cho robot di động chịu nhiễu bất định. Các đóng góp chính gồm:

- Thiết kế bộ điều khiển điều hướng bậc cao cho robot di động dạng vi sai dựa trên phương trình động học sử dụng phương pháp học tăng cường.

- Thiết kế bộ điều khiển cơ cấu chấp hành dựa trên phương pháp điều khiển thời gian định trước.
- Mạng RBF thích nghi được sử dụng để ước tính tham số hệ thống chưa biết trước cho bộ điều khiển cơ cấu chấp hành.
- Phân tích ổn định của bộ điều khiển bậc thấp sử dụng tiêu chuẩn Lyapunov
- Xây dựng mô phỏng để kiểm tra chất lượng hai bộ điều khiển được đề xuất.
- Kết hợp so sánh với các phương pháp điều khiển cơ cấu chấp hành hiện đại khác nhằm nêu bật sự ưu việt.

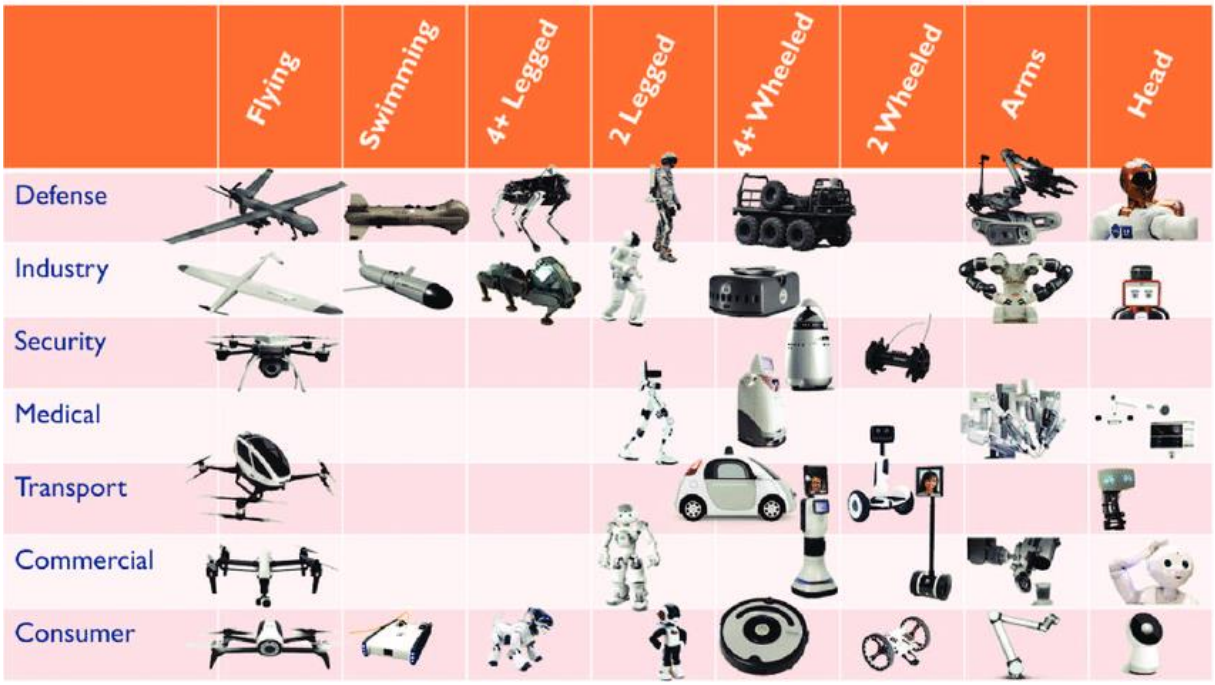
Phương pháp nghiên cứu.

- Phương pháp nghiên cứu, tổng hợp tài liệu.
- Phương pháp mô hình hóa và phân tích lý thuyết
- Phương pháp mô phỏng.
- Phương pháp phân tích dữ liệu, đánh giá hiệu quả.

CHƯƠNG 1: TỔNG QUAN VỀ ĐIỀU KHIỂN ROBOT DI ĐỘNG CHỊU NHIỄU BẤT ĐỊNH

1.1. Khái quát chung về robot và robot di động

Trong những thập kỷ gần đây, sự phát triển vượt bậc của công nghệ tự động hóa và trí tuệ nhân tạo đã thúc đẩy mạnh mẽ quá trình nghiên cứu và ứng dụng robot trong nhiều lĩnh vực khác nhau của đời sống và sản xuất. Robot được định nghĩa là một thiết bị cơ điện tử có khả năng thực hiện các hành động được lập trình sẵn, hoặc điều chỉnh hành vi của mình dựa trên thông tin thu nhận từ môi trường thông qua các cảm biến, có khả năng thực hiện một loạt các nhiệm vụ từ đơn giản đến phức tạp một cách tự động hoặc bán tự động [12], [13]. [1, 2]. Chúng thường được lập trình để cảm nhận môi trường, xử lý thông tin và thực hiện các hành vi tương ứng [14]. Tùy thuộc vào cấu trúc và chức năng, robot có thể thực hiện các tác vụ nguy hiểm, lặp đi lặp lại hoặc yêu cầu độ chính xác cao mà con người khó có thể đảm nhận.



Hình 1.1. Một số dạng robot di động phổ biến [15]

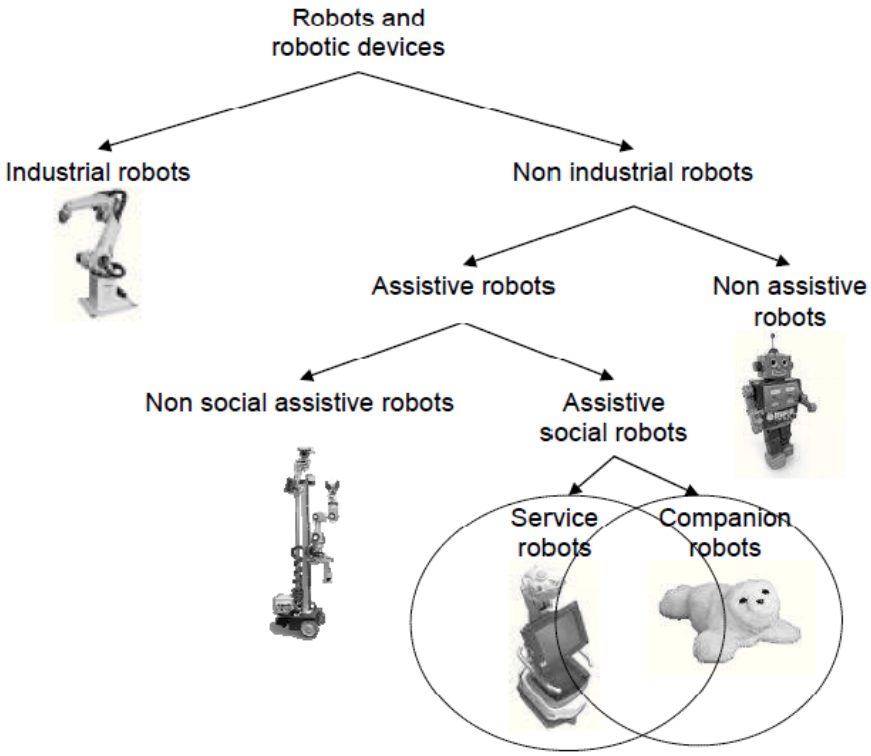
Ngày nay, robot được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau như: sản xuất công nghiệp (robot hàn, lắp ráp, sơn...), y tế (robot phẫu thuật, robot chăm sóc bệnh nhân), quân sự (robot trinh sát, robot chiến đấu), nông nghiệp (robot gieo hạt, thu hoạch) và cả trong đời sống hằng ngày (robot hút bụi, robot phục vụ) [16], [17], [18]. Với sự phát triển mạnh mẽ của các công nghệ như trí tuệ nhân tạo, học máy, thị giác máy tính và cảm biến thông minh, robot hiện đại ngày càng thông minh, linh hoạt và tương tác tốt hơn với con người và môi trường [19], [20].

Một hệ thống robot điển hình gồm các thành phần cơ bản như:

- Bộ chấp hành: Tạo chuyển động cho robot, thường là các động cơ điện, thủy lực hoặc khí nén.
- Cảm biến: Thu thập thông tin về môi trường xung quanh hoặc trạng thái bên trong của robot.
- Bộ điều khiển: Xử lý tín hiệu từ cảm biến và đưa ra lệnh điều khiển cho bộ chấp hành.
- Hệ truyền động: Truyền lực từ bộ chấp hành tới bộ phận di chuyển hoặc cánh tay robot.

Robot có thể được phân loại theo nhiều cách như: cấu trúc (robot tay máy, robot hình người, robot di động...), chức năng (robot công nghiệp, robot dịch vụ...), hoặc khả năng tự chủ (robot điều khiển từ xa, bán tự động, hoàn toàn tự động). Trong số đó, robot di động là một trong những phân nhóm quan trọng và có nhiều tiềm năng ứng dụng trong thế giới thực [21], [22], [23]. Từ khi ra đời, robot đã mang lại những thay đổi đáng kể trong cách thức con người làm việc, giúp tăng năng suất lao động, cải thiện độ chính xác, đồng thời giảm thiểu rủi ro trong các môi trường nguy hiểm như hầm mỏ, nhà máy hóa chất, hoặc không gian vũ trụ. Sự kết hợp giữa robot và trí tuệ nhân tạo đang mở ra những cơ hội mới cho các hệ thống robot thông minh, có khả năng học hỏi từ dữ liệu, thích nghi với môi trường và ra quyết định trong thời gian thực.

Trong bối cảnh chuyển đổi số và công nghiệp 4.0, nghiên cứu và phát triển robot không chỉ là xu hướng tất yếu, mà còn đóng vai trò chiến lược trong việc nâng cao năng lực cạnh tranh của các quốc gia và doanh nghiệp. Do đó, việc tiếp cận các phương pháp thiết kế, điều khiển và ứng dụng robot một cách có hệ thống là điều cần thiết đối với các nhà khoa học, kỹ sư và sinh viên trong lĩnh vực cơ điện tử, tự động hóa và trí tuệ nhân tạo.

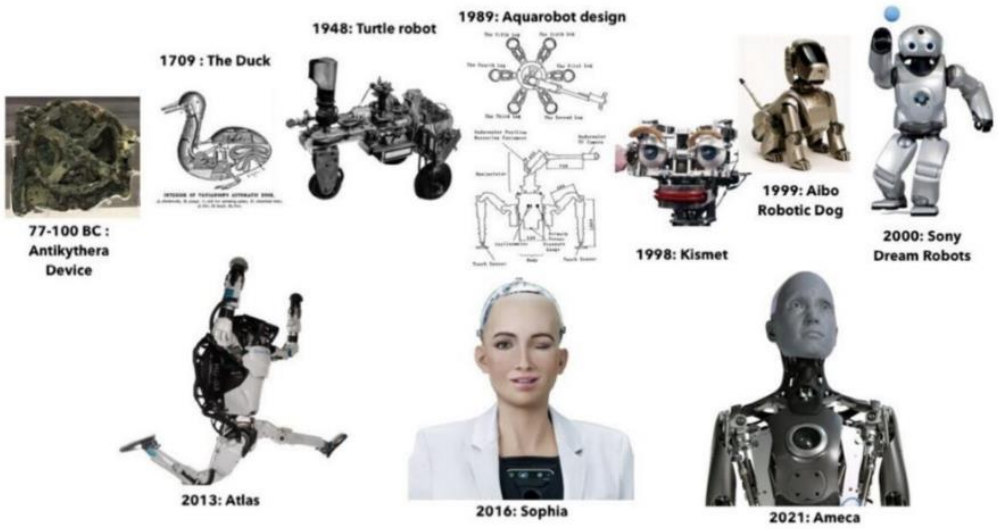


Hình 1.2. Phân loại robot theo cấu trúc và chức năng [24]

1.2. Tổng quan về robot di động

1.2.1. Lịch sử phát triển robot di động

Robot di động, với khả năng tự hành và tương tác với môi trường một cách linh hoạt, đã trở thành một lĩnh vực nghiên cứu quan trọng trong ngành robot học hiện đại. Lịch sử phát triển của robot di động là một quá trình dài gắn liền với sự tiến bộ của các ngành khoa học liên quan như cơ điện tử, trí tuệ nhân tạo, cảm biến và xử lý tín hiệu.

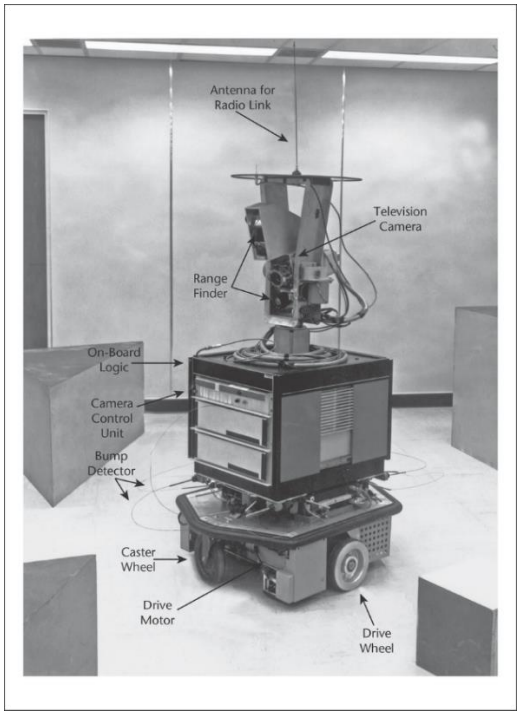


Hình 1.3. Các robot tiêu biểu trong lịch sử [25]

Ý tưởng về các cỗ máy có khả năng tự di chuyển đã xuất hiện từ rất sớm trong lịch sử, tiêu biểu là các thiết bị tự động thời kỳ Hy Lạp cổ đại và các cơ cấu tự hành thời Phục Hưng. Tuy nhiên, phải đến giữa thế kỷ 20, khi ngành điều khiển học và máy tính số phát triển, các robot di động hiện đại mới thực sự bắt đầu hình thành. Các robot đầu tiên trong giai đoạn này chủ yếu là thiết bị cơ điện đơn giản, hoạt động theo chương trình định sẵn và chưa có khả năng cảm nhận môi trường.

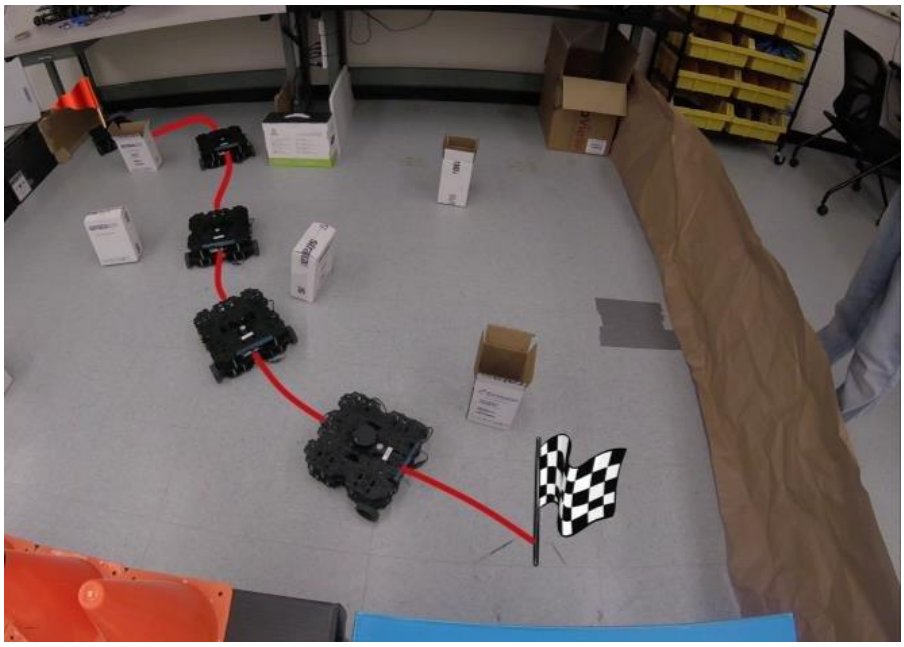
Một trong những robot di động hiện đại đầu tiên là Shakey, được phát triển vào cuối những năm 1960 tại SRI International. Shakey có khả năng di chuyển, quan sát môi trường và lên kế hoạch hành động – đặt nền móng cho các nghiên cứu sau này về robot thông minh.

Đến thập niên 1970, dự án Shakey của Viện Nghiên cứu Stanford (SRI) trở thành một trong những robot di động đầu tiên có khả năng lập kế hoạch hành động dựa trên dữ liệu cảm biến [26]. Từ những năm 1980, với sự phát triển của các vi điều khiển và công nghệ điện tử, robot di động được ứng dụng rộng rãi hơn, điển hình là các robot xe tự hành tại Carnegie Mellon University [27], [28].

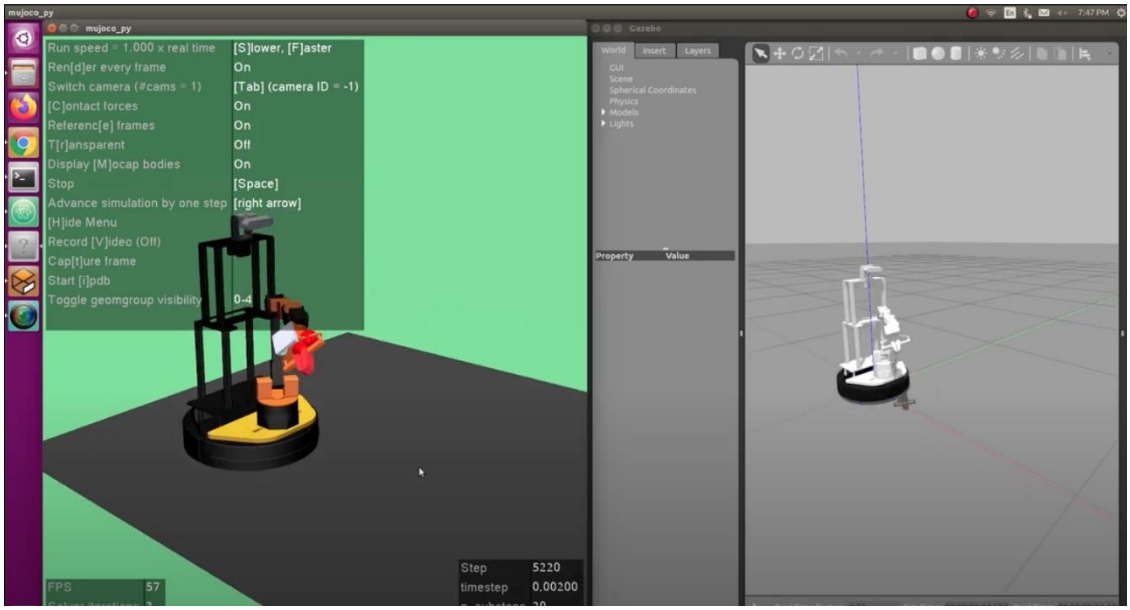


Hình 1.4. Mô hình robot Sharkey [29]

Sang những thập niên 1980–1990, cùng với sự tiến bộ về vi điều khiển, hệ thống nhúng và cảm biến, nhiều nền tảng robot di động đã được phát triển phục vụ nghiên cứu và ứng dụng. Các robot như Khepera, Pioneer, TurtleBot hay Clearpath Jackal được sử dụng rộng rãi trong môi trường học thuật và công nghiệp [30].

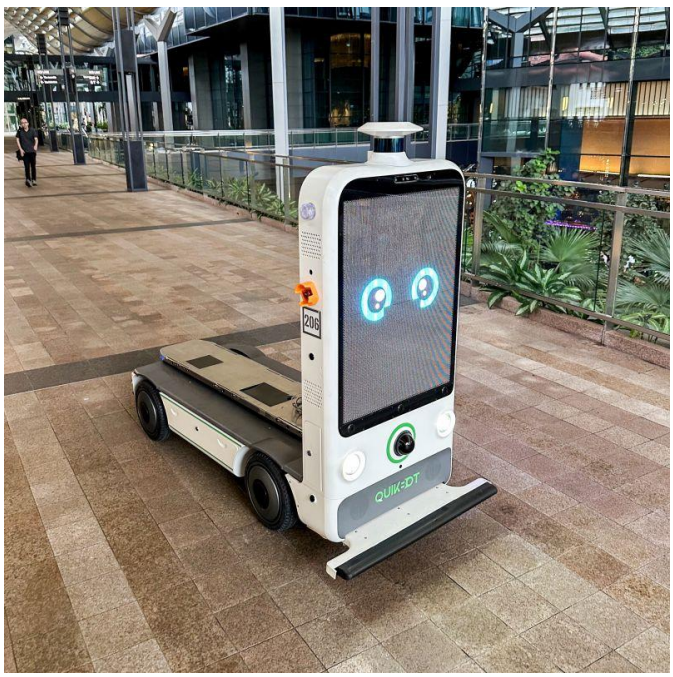


Hình 1.5. Robot tự hành turtlebot 3 tránh vật cản [31]



Hình 1.6. Mô phỏng robot trên ROS Gazebo [32]

Đến thời điểm hiện nay, các thành tựu trong lĩnh vực trí tuệ nhân tạo, thị giác máy tính, học sâu và học tăng cường đã giúp robot di động có thể tự động điều hướng, né tránh vật cản, học hỏi từ môi trường và tương tác phức tạp với con người. Các ứng dụng như xe tự lái, robot giao hàng, robot hỗ trợ y tế trong bệnh viện là minh chứng rõ ràng cho tiềm năng và ảnh hưởng của robot di động trong cuộc sống hiện đại.



Hình 1.7. Robot dịch vụ trang bị các cảm biến tự hành [33]

1.2.2. Lịch sử nghiên cứu điều khiển robot di động

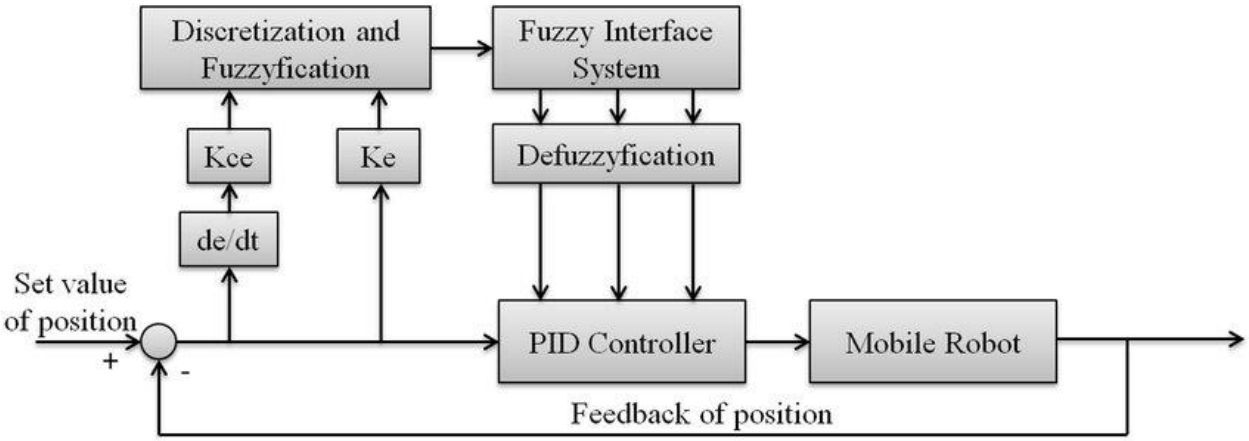
Trong lĩnh vực robot di động, điều khiển là một thành phần cốt lõi quyết định đến khả năng vận hành chính xác, ổn định và linh hoạt của robot trong môi trường thực. Ban đầu, các phương pháp điều khiển cổ điển như điều khiển tỉ lệ – tích phân – đạo hàm (PID), điều khiển tuyến tính hóa, điều khiển phản hồi trạng thái được áp dụng cho các robot có động học đơn giản trong môi trường xác định.

Tuy nhiên, khi robot hoạt động trong môi trường không cấu trúc hoặc có nhiều nhiễu và bất định, các phương pháp cổ điển trở nên không hiệu quả. Từ đó, các phương pháp điều khiển nâng cao đã ra đời như:

Quá trình nghiên cứu điều khiển robot di động trải qua ba giai đoạn chính:

- **Giai đoạn điều khiển cổ điển:**

  Bao gồm các phương pháp điều khiển phản hồi như PID, điều khiển tuyến tính và điều khiển logic mờ [34], [35]. Các phương pháp này phù hợp với hệ thống đơn giản, ít nhiễu và có thể mô hình hóa chính xác.

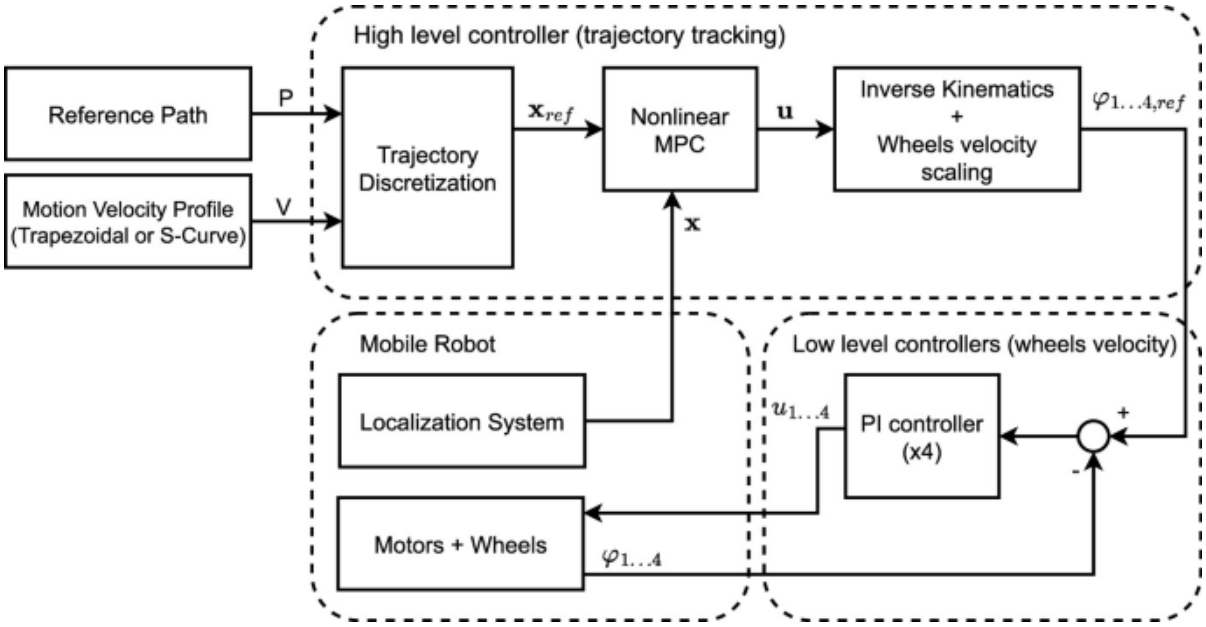


Hình 1.8. Bộ điều hướng robot dựa trên phương pháp PID mờ [36]

- **Giai đoạn điều khiển phi tuyến và thích nghi:**

  Để đối phó với tính phi tuyến và sự không chắc chắn trong động học robot, các phương pháp như điều khiển trượt, điều khiển thích nghi, và điều khiển tối ưu được phát triển [37]. Tuy nhiên, việc áp dụng các

phương pháp này phụ thuộc nhiều vào mô hình chính xác của hệ thống.



Hình 1.9. Điều khiển robot dựa trên phương pháp dự báo tối ưu [38]

- **Giai đoạn điều khiển học máy và học tăng cường:**

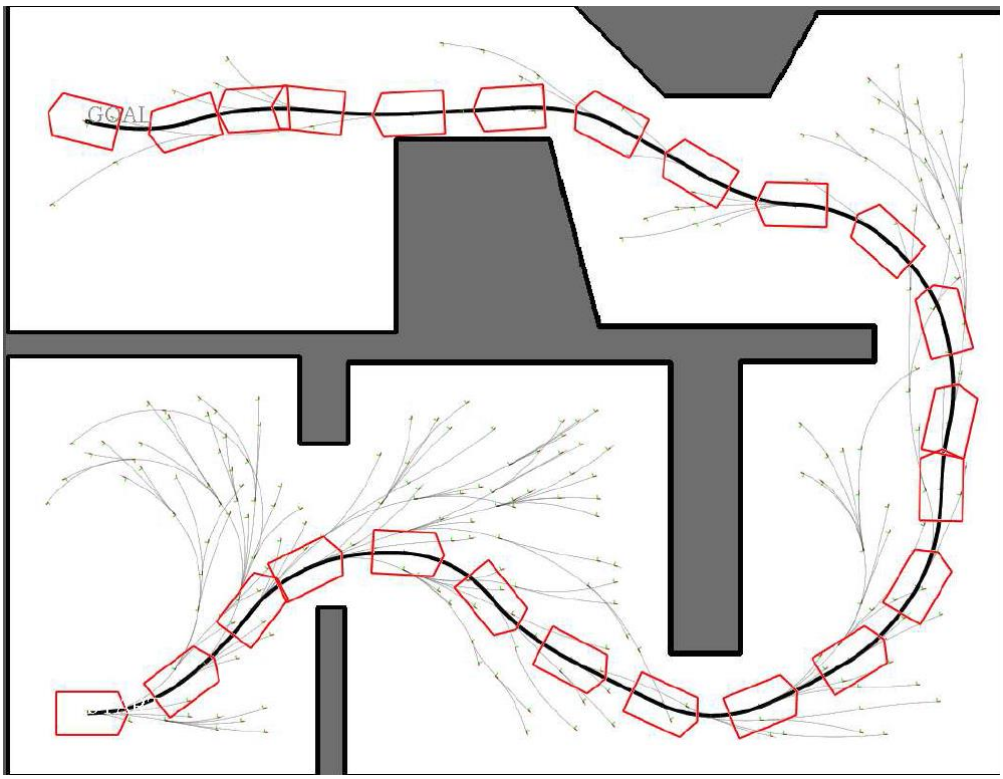
  Gần đây, với sự phát triển của trí tuệ nhân tạo, các phương pháp học máy, đặc biệt là học tăng cường sâu (Deep Reinforcement Learning - DRL), được ứng dụng để điều khiển robot di động mà không cần mô hình hệ thống [39], [40]. Robot có thể học chính sách điều khiển tối ưu thông qua tương tác với môi trường, giúp thích nghi với sự thay đổi và nhiễu không xác định.

Hình 1.10. Lịch sử phát triển của robot di động [41]

1.3. Các vấn đề thách thức trong điều khiển robot di động

Trong môi trường thực tế, robot di động thường phải hoạt động dưới điều kiện có nhiều bất định và nhiễu. Các thách thức lớn bao gồm sai số cảm biến, thay đổi môi trường và động lực học không chính xác hoặc thay đổi theo thời gian [42], [43], [44], [45] [46].

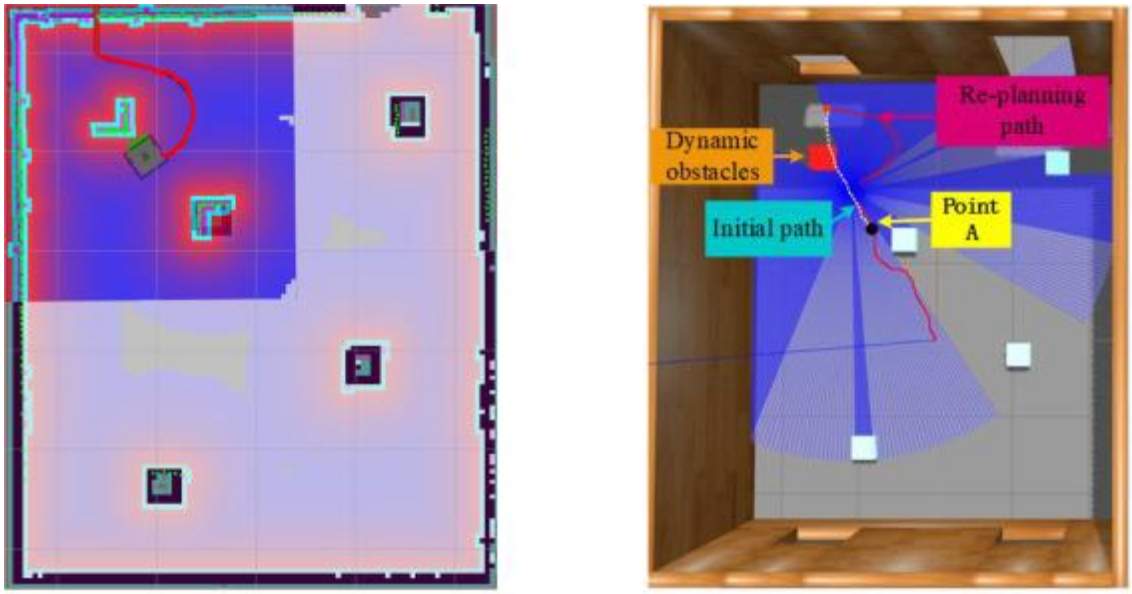
1.3.1. Bài toán lập quỹ đạo đường đi trong môi trường thay đổi



Hình 1.11. Lập quỹ đạo đường đi sử dụng phương pháp cây ngẫu nhiên [47]

Việc lập kế hoạch đường đi trong môi trường có chướng ngại vật là một vấn đề then chốt trong điều hướng robot. Các thuật toán truyền thống như A*, Dijkstra, hoặc D* Lite thường yêu cầu mô hình môi trường tĩnh và tiêu tốn tài nguyên tính toán đáng kể [48] .

Tuy nhiên, trong môi trường động hoặc không thể dự đoán, các phương pháp học tăng cường cho phép robot học chính sách điều hướng tối ưu từ tương tác, giúp né tránh vật cản mà không cần lập kế hoạch quỹ đạo cục bộ [49], [50], [51]. Điều này giúp giảm đáng kể chi phí tính toán và tăng tính phản ứng linh hoạt.



(a) Rviz visual interface (b) Navigation result

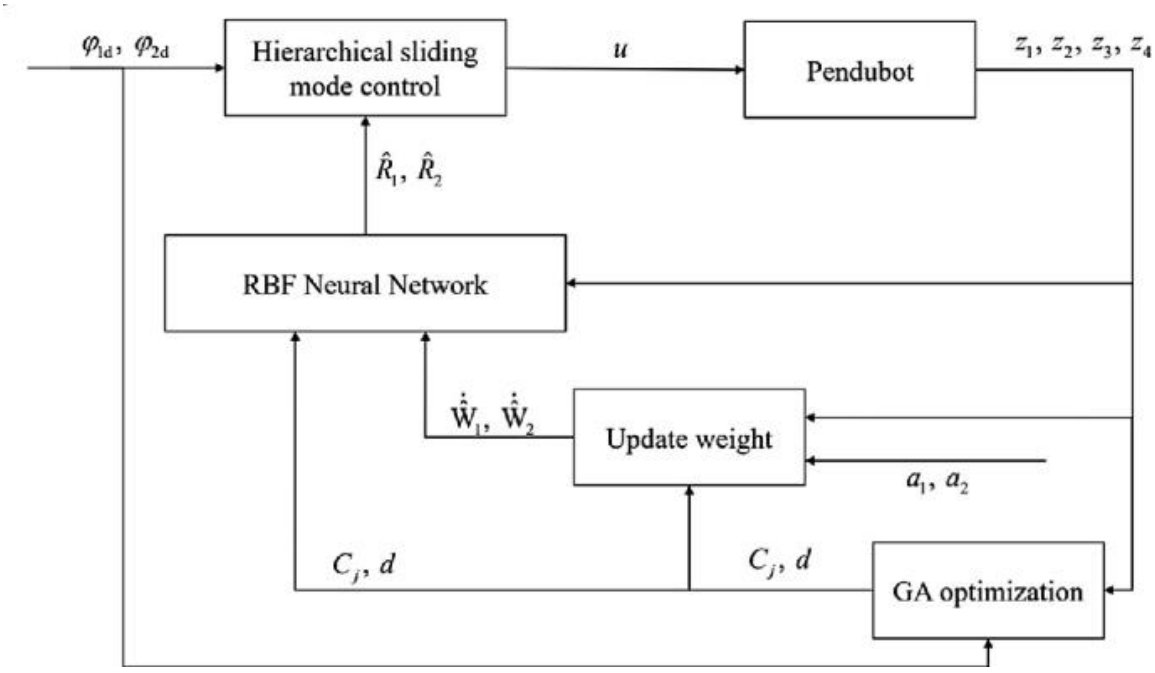
Hình 1.12. Điều hướng robot trong trường hợp thông tin về môi trường không biết trước [52]

1.3.2. Bài toán điều khiển robot khi động lực học thay đổi

Động lực học của robot di động có thể thay đổi do nhiều yếu tố như tải trọng, địa hình, hao mòn linh kiện hoặc điều kiện thời tiết. Việc điều khiển cơ cấu chấp hành sử dụng các bộ điều khiển thông thường như PID hoặc LQR không đảm bảo hiệu quả khi mô hình động học không còn chính xác [53], [54].

Để giải quyết vấn đề này, một số phương pháp điều khiển như điều khiển bền vững như điều khiển trượt (SMC) được xây dựng nhằm mục tiêu đảm bảo khả năng chống nhiễu cho robot. Tuy nhiên phương pháp điều khiển trượt chỉ thực sự có tác dụng với một số tham số nhiễu của hệ thống, và không đảm bảo chất lượng khi một số tham số hệ thống không xác định. Để xử lý vấn đề tham số hệ thống không xác định và ảnh hưởng của nhiễu, bộ điều khiển trượt thích nghi được phát triển (Adaptive SMC) [55] dưa trên mạng nơ-ron, đặc biệt là mạng nơ-ron cơ sở hàm (RBFNN), được sử dụng để mô hình hóa và điều chỉnh hành vi hệ thống cơ cấu chấp hành trong thời gian thực, mang lại khả năng thích nghi tốt hơn trước các biến động của hệ thống. Tuy nhiên những dạng điều khiển này sẽ có đáp ứng phụ thuộc vào trạng thái ban đầu của hệ thống có nghĩa là sai số ban đầu càng

lớn, thời gian đáp ứng càng kéo dài. Điều này ảnh hưởng tới tốc độ của robot phản xạ với các thay đổi của môi trường như vật cản động.



Hình 1.13. Phương pháp điều khiển thích nghi RBF cho hệ thống động lực không xác định [56]

1.4. Định hướng nghiên cứu đề tài

Trước những thách thức kể trên, đề án hướng đến giải quyết ba vấn đề trọng yếu nhằm nâng cao hiệu suất và độ tin cậy cho hệ thống robot di động trong điều kiện có nhiễu bất định:

- Thứ nhất, đề tài nghiên cứu và phát triển thuật toán điều hướng dựa trên học tăng cường sâu (Deep Reinforcement Learning), cho phép robot tự học cách di chuyển tới mục tiêu mà không cần lập quỹ đạo cục bộ, qua đó giảm thiểu khối lượng tính toán và tăng tính linh hoạt trong môi trường thay đổi.
- Thứ hai, đề tài xây dựng bộ điều khiển định thời thích nghi dựa trên mạng nơ-ron cơ sở hàm (RBFNN), giúp điều chỉnh kịp thời phản ứng điều khiển khi động lực học hệ thống biến đổi do tải trọng, địa hình hoặc nhiễu ngoài.

Bằng việc kết hợp các phương điều hướng thông minh và điều khiển thích nghi phi tuyến, đề tài kỳ vọng sẽ góp phần cải thiện độ chính xác, độ ổn định và tính linh hoạt cho robot di động, đặc biệt trong các ứng dụng thực tế như robot giao hàng, robot công nghiệp tự hành hoặc robot cứu hộ trong môi trường phức tạp.

Kết luận chương 1

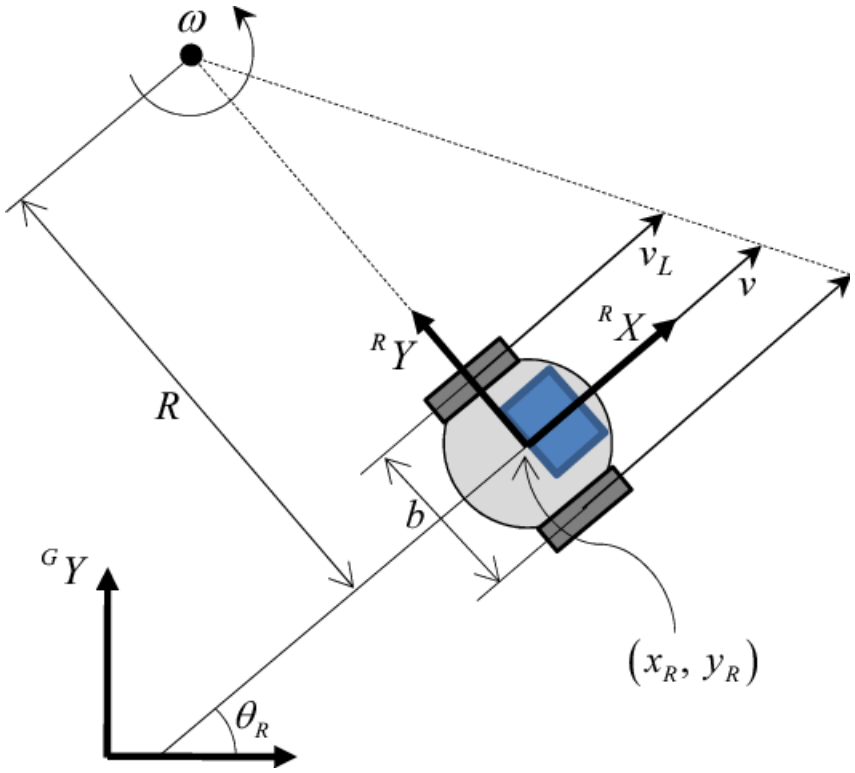
Chương đã cung cấp một nền tảng kiến thức cơ bản và tổng quan toàn diện về lĩnh vực robot di động, từ khái niệm và lịch sử phát triển đến các xu hướng nghiên cứu điều khiển hiện đại. Qua việc phân tích các nhóm thách thức chính trong điều khiển robot di động dưới điều kiện nhiễu bất định, chương này đã làm rõ các giới hạn của phương pháp truyền thống và khẳng định sự cần thiết của các giải pháp tiên tiến, thích nghi và thông minh hơn. Từ đó đặt ra vấn đề nghiên cứu đề tài: "NGHIÊN CỨU ĐIỀU KHIỂN ROBOT DI ĐỘNG CHỊU NHIỄU BẤT ĐỊNH DỰA TRÊN HỌC TĂNG CƯỜNG".

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Mô hình động học robot di động dạng vi sai

Trong điều khiển robot di động, việc lựa chọn mô hình toán học làm cơ sở thiết kế điều khiển có ảnh hưởng quan trọng đến độ phức tạp và tính khả thi của hệ thống. Phương pháp điều khiển dựa trên phương trình động học thường được ưu tiên trong các ứng dụng thực tế nhờ một số ưu điểm nổi bật so với phương pháp dựa trên phương trình động lực học. Trước hết, mô hình động học mô tả mối quan hệ hình học giữa vận tốc đầu ra và trạng thái robot, bỏ qua các yếu tố như mô-men quán tính, lực cản và ma sát, do đó có cấu trúc đơn giản hơn và dễ triển khai hơn trong các bộ điều khiển phản hồi. Điều này giúp giảm đáng kể yêu cầu tính toán, đặc biệt phù hợp với các hệ thống robot di động có tài nguyên phần cứng hạn chế. Thứ hai, phương pháp điều khiển dựa trên động học thường yêu cầu ít thông số hệ thống hơn, do đó ít bị ảnh hưởng bởi sai số mô hình, từ đó cải thiện tính ổn định và khả năng áp dụng trong môi trường thực. Ngoài ra, trong nhiều trường hợp như robot vi sai hoặc robot omni, việc điều khiển dựa trên vận tốc là đủ để đảm bảo định vị chính xác, mà không cần giải bài toán điều khiển mô-men ở tầng động lực học phức tạp. Nhờ các ưu điểm đó, trong đề án này điều khiển dựa trên mô hình động học là phương pháp được chọn để xây dựng bộ điều khiển vị trí cho robot với những ưu điểm được liệt kê sau:

- Mô hình động học được mô tả đơn giản, chính xác hơn so với sử dụng mô hình động lực học
- Thiết kế bộ điều khiển vị trí dựa trên mô hình động học kết hợp bộ điều khiển cơ cấu chấp hành bậc thấp cho kết quả đáp ứng tốt trong trường hợp robot chịu nhiễu bất định
- Giúp đơn giản quá trình thiết kế bộ điều khiển vị trí dựa trên phương pháp học tăng cường



Hình 2.1. Mô tả động học robot vi sai [57]

Robot di động dạng vi sai sử dụng hai bánh xe chủ động gắn hai bên thân và có thể điều khiển độc lập. Chuyển động của robot được mô tả dựa trên hai hệ tọa độ:

- Hệ quy chiếu quán tính: $\{{}^{G}X, {}^{G}Y\}$
- Hệ quy chiếu gắn với robot: $\{{}^{R}X, {}^{R}Y\}$

Trạng thái của robot tại một thời điểm bất kỳ được biểu diễn bằng vector vị trí $P = [x, y, \theta]$, trong đó x, y là tọa độ trong hệ quy chiếu quán tính và θ là góc quay so với trục ${}^{G}X$.

Mối quan hệ giữa vận tốc trong hệ robot và hệ quán tính được mô tả bằng công thức biến đổi:

$$\dot{\varepsilon}_R = R(\theta) \cdot \dot{\varepsilon}_I = R(\theta) \cdot \left[\dot{x}, \dot{y}, \dot{\theta}\right]^T \tag{2.1}$$

Trong đó:

$$R(\theta)=\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix} \tag{2.2}$$

Trong cấu trúc robot này, nếu hai bánh quay với cùng tốc độ, robot sẽ di chuyển thẳng. Nếu một bánh quay nhanh hơn bánh còn lại, robot sẽ rẽ theo hướng ngược với bánh có tốc độ cao hơn. Nếu hai bánh quay ngược chiều với cùng tốc độ, robot sẽ quay tại chỗ. Một bánh đỡ (bánh xoay tự do) thường được gắn để tăng độ ổn định cơ học.

Để đảm bảo mô hình phù hợp với thực tế, một ràng buộc phi tuyến (phi holonomic) được áp dụng, phản ánh giả định rằng robot không trượt ngang:

$$\dot{x}\sin\theta-\dot{y}\cos\theta=0 \tag{2.3}$$

Hơn nữa, quãng đường bánh xe đi được không đủ để xác định vị trí cuối cùng của robot nếu không xét đến sự thay đổi theo thời gian:

$$\begin{aligned}&S_{1R}=S_{2R};\ S_{1L}=S_{2L};\ S_1=S_2\\ &x_1\neq x_2;\ y_1\neq y_2\end{aligned} \tag{2.4}$$

Mở rộng: Mô hình động học tổng quát sử dụng cơ sở không gian

Sắp xếp lại các ràng buộc động học dạng $V^T(q)\dot{q}=0$, ta thấy rằng các vận tốc tổng quát $\dot{q}$ thuộc về không gian null của $V^T(q)$, có chiều là $n-k$. Giả sử $\{b_1(q),\ldots,b_{n-k}(q)\}$ là cơ sở của không gian null này, ta có thể viết mô hình động học như sau:

$$\dot{q}=\sum_{i=1}^{n-k}b_i(q)u_i=B(q)u \tag{2.5}$$

Trong đó:

- $u=[u_1\ldots u_{n-k}]^T\in\mathbb{R}^{n-k}$: vector đầu vào
- $q\in\mathbb{R}$: vector trạng thái
- $B(q)$: ma trận cơ sở không gian null của $V^T(q)$

Đối với robot di động, các thành phần thường được chọn sao cho có ý nghĩa vật lý, chẳng hạn như vận tốc tiến và vận tốc quay của robot, nhưng không trực tiếp đại diện cho lực hay mô men.

Với robot vi sai có tọa độ tổng quát $q=\begin{bmatrix} x & y & \theta \end{bmatrix}$, ràng buộc không trượt ngang được viết lại:

$$V^T(q)\dot{q}=\begin{bmatrix} \sin\theta & -\cos\theta & 0 \end{bmatrix}\dot{q}=0 \tag{2.6}$$

Cơ sở của không gian null □ $\left(V^T(q)\right)$ có thể chọn là:

$$B(q)=\begin{bmatrix} b_1(q) & b_2(q) \end{bmatrix}=\begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \tag{2.7}$$

Do đó, mô hình động học trở thành:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}=\begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix}v+\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\omega \tag{2.8}$$

Trong đó v và ω lần lượt là vận tốc tuyến tính và góc của robot.

Giữa v, ω và vận tốc góc của hai bánh xe ω_L, ω_R tồn tại mối quan hệ:

$$\begin{aligned} v&=\frac{r}{2}\left(\omega_R+\omega_L\right) \\ \omega&=\frac{r}{l}\left(\omega_R-\omega_L\right) \end{aligned} \tag{2.9}$$

Với r là bán kính bánh xe, l là khoảng cách giữa hai bánh.

2.2. Mô hình động lực học cơ cấu chấp hành cho robot di động

Trong hệ thống robot di động, cơ cấu chấp hành đóng vai trò quan trọng trong việc chuyển đổi tín hiệu điều khiển thành chuyển động thực tế. Đây là thành phần trực tiếp điều khiển hành vi động học của robot, đảm bảo khả năng di chuyển linh hoạt và chính xác trong môi trường làm việc. Cơ cấu chấp hành bao gồm các loại

động cơ, hộp số, hệ truyền động, cũng như các thành phần cơ khí liên quan đến bánh xe, chân hoặc hệ thống bánh xích tùy thuộc vào thiết kế robot.

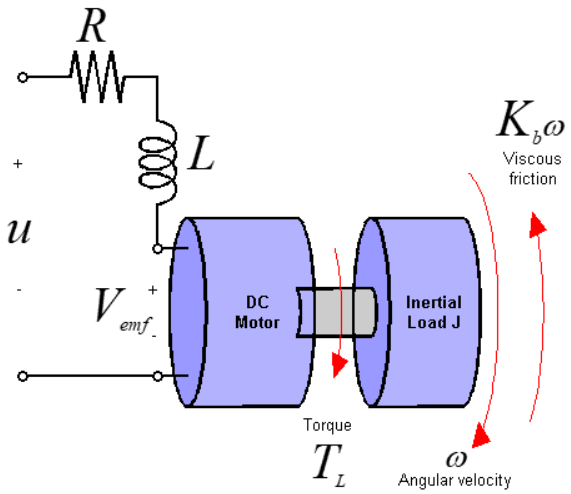
Trong kiến trúc tổng thể của robot di động, bộ điều khiển vị trí gửi các tín hiệu đầu ra đến cơ cấu chấp hành để thực thi các lệnh di chuyển như tiến, lùi, rẽ trái, rẽ phải, hay duy trì vận tốc và hướng di chuyển ổn định. Do đó, hiệu suất, độ chính xác và độ bền của cơ cấu chấp hành có ảnh hưởng trực tiếp đến khả năng điều hướng, điều khiển và tự chủ của robot

Các loại động cơ điện là bộ chấp hành phổ biến nhất trong robot di động nhờ ưu điểm điều khiển dễ dàng, phản hồi nhanh và hiệu suất cao. Một số loại động cơ tiêu biểu bao gồm:

- Động cơ điện một chiều
- Động cơ servo
- Động cơ bước

Hình 2.2. Động cơ DC dùng cho cơ cấu dẫn động [58]

Trong đề án này, động cơ DC được sử dụng trong robot di động để tạo chuyển động cho bánh xe. Việc mô hình hóa đúng động cơ là cơ sở quan trọng để thiết kế bộ điều khiển hiệu quả.



Hình 2.3. Mô hình động cơ DC dưới tác dụng của tải thay đổi và ma sát [59]

Mô hình điện của động cơ DC:

$$u = L\frac{di}{dt} + Ri + K_b\omega \tag{2.10}$$

Mô hình cơ học:

$$J\frac{d\omega}{dt} = K_t i - B\omega - T_L \tag{2.11}$$

Trong đó:

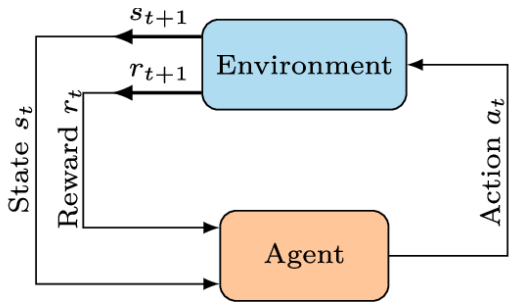
- u: điện áp cung cấp
- i: dòng điện phần ứng
- ω: tốc độ quay của rotor
- L: điện cảm, R: điện trở phần ứng
- K_b, K_t: hệ số hằng số động cơ
- J: mô men quán tính, B: hệ số ma sát, T_L momen tải ngoại lực tác động lên trục động cơ. Đây là 3 phần tử gây nhiễu ảnh hưởng tới chất lượng điều khiển

Do giá trị của điện cảm L là rất nhỏ, do đó trong đề án này, để đơn giản quá trình thiết kế và mô phỏng, giá trị điện cảm được bỏ qua. Các tham số điện trở phần cứng R, hệ số động cơ K_t, K_b là không đổi, được xác định trước từ nhà cung

cấp. Các hệ số momen quán tính J và hệ số ma sát B là không xác định trước. Đồng thời giá trị momen tải ngoại lực T_L được coi là nhiễu ngẫu nhiên không biết trước tác động lên động cơ.

2.3. Cơ sở lý thuyết về học tăng cường

Học tăng cường (Reinforcement Learning - RL) là một hướng tiếp cận trong học máy, nơi một tác nhân (agent) học cách tương tác với môi trường nhằm đạt được chiến lược hành động tối ưu nhằm tối đa hóa phần thưởng tích lũy theo thời gian. Tại mỗi thời điểm rời rạc t, tác nhân quan sát trạng thái hiện tại $s \in S$, sau đó lựa chọn một hành động $a \in A$ theo một chính sách (policy) $\pi : S \to A$. Sau khi thực hiện hành động, tác nhân nhận được phần thưởng r và trạng thái mới của môi trường s'.



Hình 2.4. Mô tả cấu trúc hoạt động của học tăng cường [60]

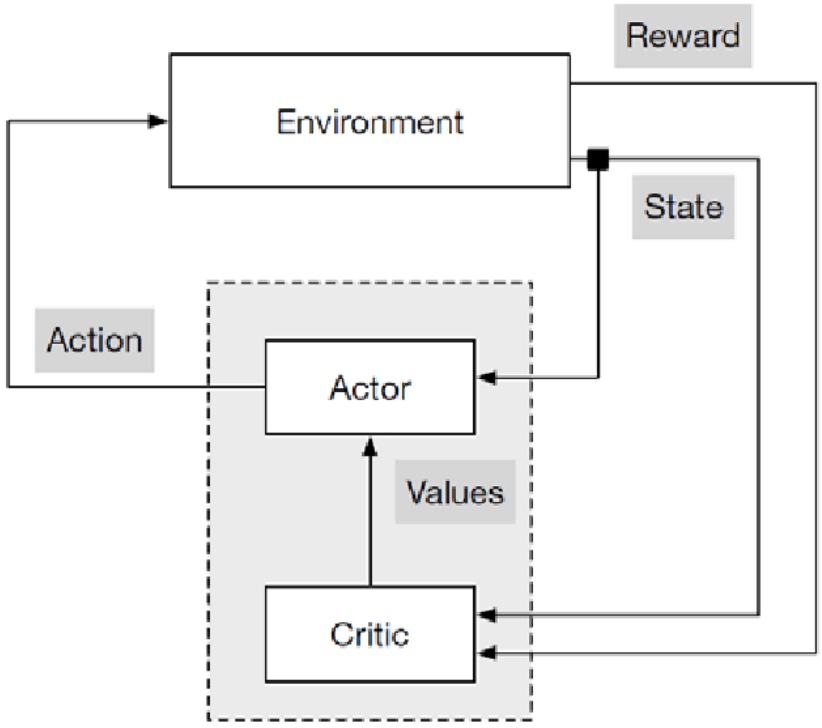
Tổng phần thưởng được định nghĩa là tổng phần thưởng chiết khấu theo thời gian:

$$R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i) \tag{2.12}$$

Trong đó $\gamma \in [0,1]$ là hệ số chiết khấu, quyết định mức độ ưu tiên của phần thưởng ngắn hạn so với dài hạn.

Mục tiêu của học tăng cường là tìm ra chính sách tối ưu π_ϕ (với ϕ là tham số của chính sách) sao cho giá trị kỳ vọng của tổng phần thưởng $J(\phi) = \mathrm{E}_{s_i \sim p_\pi, a_i \sim \pi}[R_0]$ được tối đa hóa. Trong các bài toán điều khiển liên tục, chính sách có thể được tối ưu thông qua đạo hàm của hàm mục tiêu theo tham số ϕ.

Phương pháp Actor-Critic:



Hình 2.5. Mô tả cấu trúc hoạt động của phương pháp Actor-Critic [61]

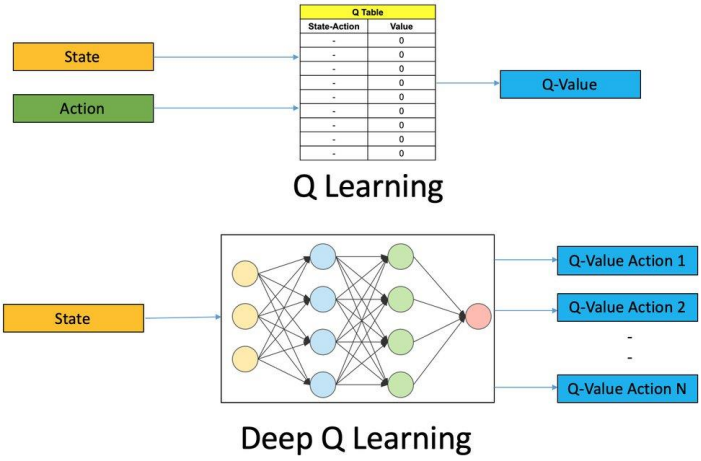
Một trong những cách tiếp cận phổ biến trong học tăng cường là kiến trúc actor-critic, trong đó:

- Actor: đại diện cho chính sách hành động $\pi_\phi(s)$
- Critic: là hàm giá trị $Q^\pi(s,a)$, đại diện cho giá trị kỳ vọng của phần thưởng khi thực hiện hành động a aaa tại trạng thái s, sau đó tiếp tục theo chính sách π

Thuật toán Deterministic Policy Gradient cập nhật actor theo công thức:

$$\nabla_\phi J(\phi) = \mathrm{E}_{s\sim p_\pi}\left[\nabla_a Q^\pi(s,a)|_{a=\pi(s)}\,\nabla_\phi \pi_\phi(s)\right] \qquad (2.13)$$

Phương pháp Q-Learning



Hình 2.6. Mô tả cơ chế phương pháp Q-learning và Deep Q-learning [62]

Trong Q-learning, hàm giá trị hành động $Q^{\pi}(s,a)$ được học thông qua phương pháp học sai số tạm thời (temporal difference learning), dựa trên phương trình Bellman:

$$Q^{\pi}(s,a) = r + \gamma \mathbb{E}_{s',a'}\left[Q^{\pi}(s',a')\right], \left(a' \sim \pi(s')\right) \tag{2.14}$$

Khi không gian trạng thái lớn, ta thường sử dụng các mô hình gần đúng (function approximator), ví dụ như mạng nơ-ron, để xấp xỉ hàm giá trị $Q_{\theta}(s,a)$ với tham số θ.

Trong Deep Q-Learning, mạng Q được huấn luyện bằng sai số tạm thời, sử dụng thêm một mạng mục tiêu cố định (target network) $Q_{\theta'}$ nhằm ổn định quá trình học. Mục tiêu huấn luyện tại mỗi bước được xác định bởi:

$$y = r + \gamma Q_{\theta'}(s',a'), \left(a' \sim \pi_{\phi'}(s')\right) \tag{2.15}$$

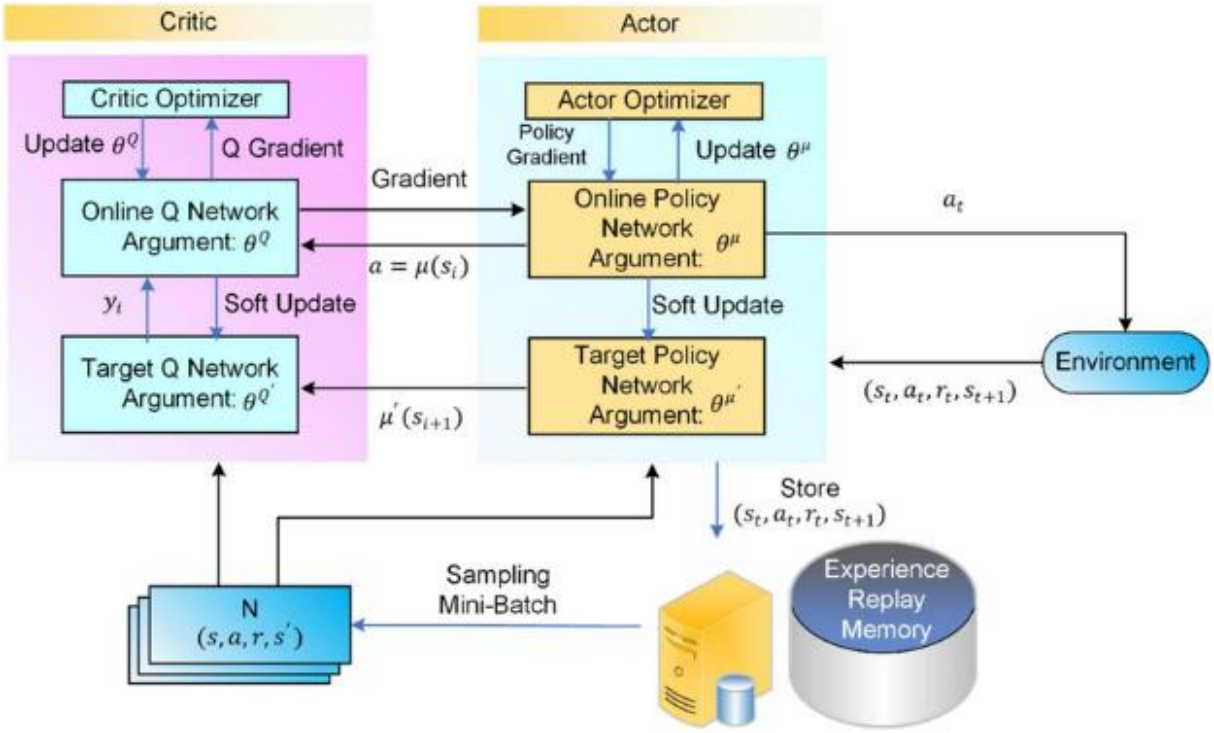
Trọng số của mạng mục tiêu θ' được cập nhật định kỳ bằng một trong hai cách: sao chép hoàn toàn từ mạng hiện tại, hoặc cập nhật dần theo công thức:

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta' \tag{2.16}$$

Trong thực tế, các mẫu kinh nghiệm (s,a,r,s') được lưu trong một **bộ nhớ phát lại kinh nghiệm** (Experience Replay Buffer) và được trích xuất ngẫu nhiên để huấn luyện, giúp giảm tương quan giữa các mẫu và cải thiện khả năng hội tụ.

Những năm gần đây, thuật toán học tăng cường cho các hệ thống điều khiển liên tục được phát triển, trong đó DDPG nổi lên như một phương pháp mới cho khả năng hoạt động trong môi trường hành vi liên tục.

Thuật toán DDPG



Hình 2.7. Mô tả hoạt động của phương pháp DDPG [63]

Thuật toán DDPG là sự kết hợp giữa DQN và Actor-Critic đã cho thấy sức mạnh tuyệt vời trong lĩnh vực điều khiển hệ thống, khả năng mở rộng ra các hệ MIMO đã được chứng minh.

Cấu trúc DDPG được triển khai gồm 4 mạng neural network:

θ^Q: Q network (Mạng phê bình) dùng để ước tính phần thưởng kỳ vọng tại trạng thái s_t, a_t

θ^μ: Mang diễn viên để xác định chính sách $\mu(s_t)$

$\theta^{Q'}$: trọng số mạng phê bình mục tiêu

$\theta^{\mu'}$: trọng số mạng diễn viên mục tiêu

Các mạng mục tiêu có cấu trúc tương tự mạng chính nó, tuy nhiên được cập nhập với tốc độ chậm hơn nhằm đảm bảo ổn định

Mạng phê bình có nhiệm vụ ước tính phần thưởng kỳ vọn $Q(s_t, a_t)$ từ phương trình (2.17)

$$Q^*(s,a) = \underset{\mathrm{P}(s'|s,a)}{\mathrm{E}}\left[r(s,a) + \gamma \max_{a'} Q^*(s,a) \right] \tag{2.17}$$

Với $\mathrm{P}(s'|s,a)$: là xác xuất chuyển đổi trạng thái s sang s' khi a tác động tới môi trường, $r(s,a)$ là giá trị điểm thưởng nhận được khi môi trường đổi trạng thái khi thực hiện hành động a.

Qua việc thực hiện quá trình thử sai, tác nhân thu được thông tin về môi trường

$\mathrm{X} = (s, a, r, s' ... d)$ với d cho biết liệu trạng thái thu được có là trạng thái cuối hay không. Thiết lập phương trình Bellman sai số trung bình để kiểm tra sự chính xác của hàm $Q_{\theta^Q}(s,a)$ do mạng phê bình đưa ra với thực tế.

$$L(\theta^Q, \mathrm{X}) = \underset{(s,a,r,s' ... d)\sim \mathrm{X}}{\mathrm{E}}\left[\left(Q_{\theta^Q}(s,a) - \left(r + \gamma(1-d) \max_{a'} Q_{\theta^Q}(s', a') \right) \right)^2 \right] \tag{2.18}$$

Phương trình cập nhập trọng số cho mạng phê bình được thực hiện như sau:

$$\nabla_{\theta^Q} = \nabla_{\theta^Q}\left(L(\theta^Q, N) \right) \tag{2.19}$$

Phương trình cập nhập trọng số cho mạng diễn viên được thực hiện theo phương trình (2.20)

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(\mathrm{s}, a \mid \theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s \mid \theta^\mu)|_{s_i} \tag{2.20}$$

Tuy nhiên có một đặc điểm của DDPG đó là thời gian training dài, thường có hiện tượng ước tính quá mức. Do đó thuật toán TD3 nổi lên như một sự cải thiện cho DDPG về hiệu quả training [64].

Thuật toán TD3:

Thuật toán TD3 là một thuật toán học tập củng cố được thiết kế cho các không gian hành động liên tục, được xây dựng dựa trên thuật toán DDPG (độ dốc chính sách xác định sâu). Nó giải quyết những hạn chế của DDPG, đặc biệt là sự đánh giá quá cao của giá trị Q, bằng cách kết hợp các kỹ thuật như được cắt đôi Q-learning, cập nhật chính sách bị trì hoãn và làm mịn chính sách mục tiêu.

Với sự cải tiến đáng kể với phương pháp DDPG có thể kể đến như:

- TD3 học được hai chức năng Q (Mạng phê bình) và sử dụng ước tính giá trị tối thiểu để hình thành các mục tiêu, giảm thiểu sai lệch đánh giá quá cao
- Chính sách (mạng diễn viên) và mạng mục tiêu được cập nhật ít thường xuyên hơn so với các chức năng Q, dẫn đến việc học ổn định hơn
- Tiếng ồn được thêm vào hành động mục tiêu trong quá trình cập nhật chính sách, khuyến khích thăm dò và ngăn chặn tác nhân khai thác các lỗi tiềm ẩn trong các ước tính chức năng Q.

Thuật toán TD3 được mô tả như sau:

Khởi tạo hai mạng định giá (critic) $Q_{\theta_1}, Q_{\theta_2}$ và một mạng chính sách (actor) π_ϕ với các tham số ngẫu nhiên θ_1, θ_2, ϕ

Khởi tạo các **mạng mục tiêu** tương ứng $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2, \phi' \leftarrow \phi$

Khởi tạo bộ nhớ kinh nghiệm (replay buffer) B

for $t = 1$ **to** T **do**

Chọn hành động với tiếng ồn khám phá $a \sim \pi_\phi(s) + \varepsilon$, $\varepsilon \sim \mathrm{N}(o, \sigma)$ và quan sát phần thưởng r và trạng thái mới s'.

Lưu trữ tuple chuyển tiếp (s, a, r, s') trong B.

Mẫu lô nhỏ của N quá trình chuyển đổi (s, a, r, s') từ B

$\tilde{a} \leftarrow \pi_{\phi'}(s') + \varepsilon$, $\varepsilon \sim clip\left(\mathrm{N}(0, \tilde{\sigma}), -c, c\right)$

$y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta_i'(s', \tilde{a})}$

Cập nhật các định giá $\theta_i \leftarrow \arg\min_{\theta_i} N^{-1}\left(y - Q_{\theta_i(s,a)}\right)^2$

If t mod d **then**

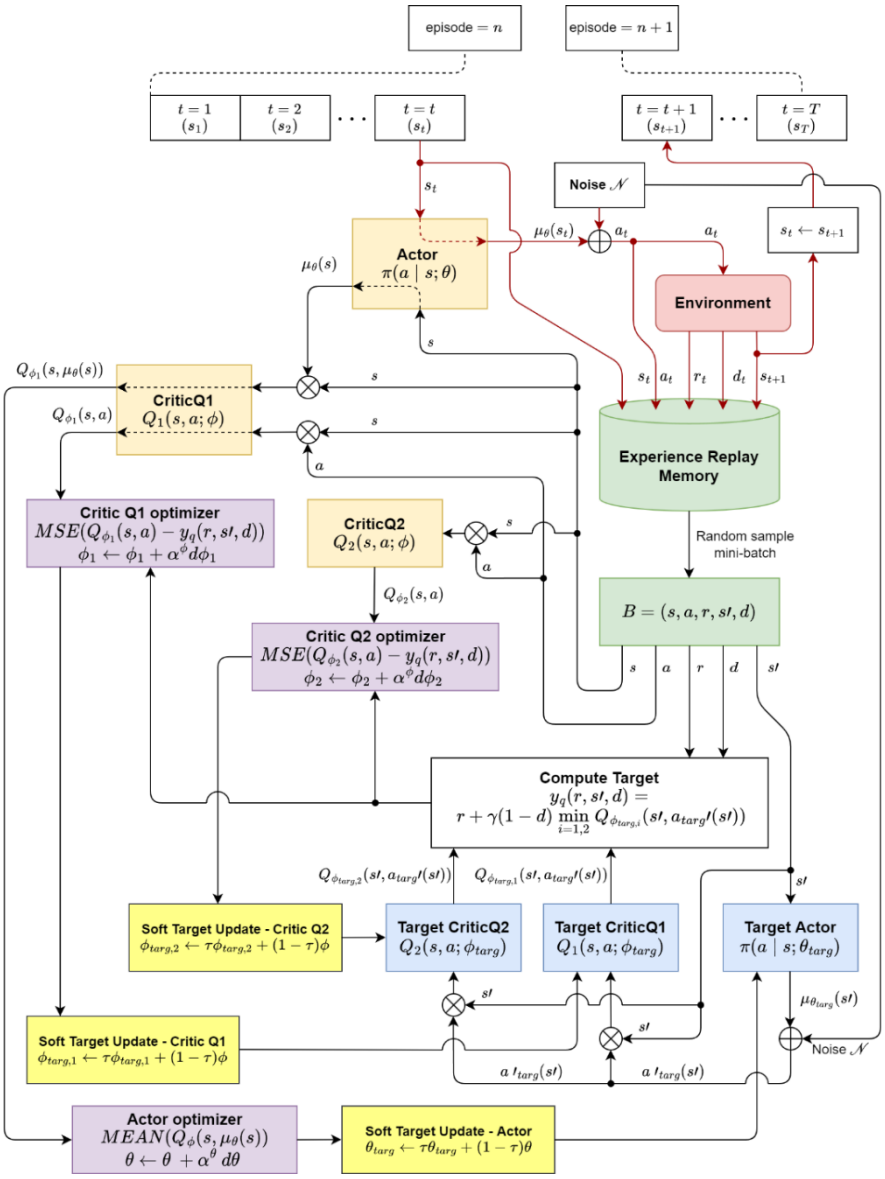
Cập nhật ϕ theo chính sách xác định:

$$\nabla_\phi J(\phi) = N^{-1}\sum \nabla_a Q_{\theta_1}(s,a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$$

Cập nhật mạng mục tiêu:

$$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$$

$$\phi' \leftarrow \tau\phi + (1-\tau)\phi'$$



Hình 2.8. Chi tiết giải thuật TD3 [65]

2.4. Cơ sở lý thuyết về phương pháp điều khiển tiền định thời gian

Điều khiển tiền định thời gian (Fixed-Time Control) là phương pháp điều khiển đảm bảo hệ thống hội tụ về điểm cân bằng trong một thời gian giới hạn, không phụ thuộc vào điều kiện ban đầu. Đây là một bước tiến so với điều khiển hữu hạn thời gian (finite-time control) – vốn phụ thuộc vào điều kiện ban đầu.

Ưu điểm:

- Đáp ứng hệ thống đạt được trong thời gian được quy định trước mà không phụ thuộc vào trạng thái ban đầu, cường độ tin cậy và an toàn cho hệ thống robot.
- Phù hợp cho các ứng dụng thời gian thực như robot di động cần đảm bảo hành động trong thời hạn nhất định.

Cơ sở toán học của phương pháp điều khiển tiền định thời gian:

Xét hệ thống có phương trình động lực học có dạng:

$$\dot{x} = g(t, x), \quad x(0) = 0 \tag{2.21}$$

Phương trình sai số hệ thống $e = x_d - x$ với x_d là tín hiệu đặt, x là tín hiệu cần điều khiển.

Theo [66] để hệ thống đáp ứng với thời gian hữu hạn, đạo hàm phương trình sai số cần đưa về dạng:

$$\dot{e} \le -k_1 e^{\alpha}(x) - k_2 e^{\beta}(x) \tag{2.22}$$

Thời gian đáp ứng của hệ T được giới hạn theo phương trình sau:

$$T \le \frac{1}{k_1(1-\alpha)} + \frac{1}{k_2(\beta-1)} \tag{2.23}$$

2.5. Cơ sở lý thuyết về điều khiển thích nghi dựa trên mạng nơ ron hàm xuyên tâm (RBFNN)

Mạng nơ-ron RBF là một kiến trúc mạng nơ-ron nhân tạo có cấu trúc đơn giản nhưng mạnh mẽ, bao gồm ba lớp: lớp đầu vào, lớp ẩn sử dụng các hàm kích hoạt dạng cơ sở xuyên tâm (thường là Gaussian), và lớp đầu ra tuyến tính. Với khả năng xấp xỉ các hàm phi tuyến bất kỳ với độ chính xác cao, mạng RBF đã được chứng minh là một công cụ hiệu quả trong thiết kế bộ điều khiển thích nghi cho các hệ thống phi tuyến, bao gồm cả robot di động. Ưu điểm nổi bật của mạng RBF trong điều khiển thích nghi là khả năng học online và xấp xỉ động lực hệ thống theo thời gian thực, cho phép bộ điều khiển tự điều chỉnh theo sự biến đổi của mô hình, nhiễu không xác định hoặc điều kiện môi trường thay đổi. Hơn nữa, do đặc trưng cấu trúc cục bộ của các hàm cơ sở xuyên tâm, mạng RBF có khả năng hội tụ nhanh và tránh được hiện tượng lan truyền sai số toàn cục như trong mạng nơ-ron lan truyền ngược. Bên cạnh đó, việc thiết kế và huấn luyện mạng RBF tương đối đơn giản và dễ hiện thực hóa trong phần cứng nhúng, giúp tăng tính khả thi cho các ứng dụng điều khiển thời gian thực. Chính vì những ưu điểm trên, mạng nơ-ron RBF ngày càng được áp dụng rộng rãi trong các bộ điều khiển thích nghi thông minh cho hệ thống robot phi tuyến và không xác định.

2.5.1. Cấu trúc mạng neural network RBF

Mạng hàm cơ sở xuyên tâm (RBF) là một trường hợp đặc biệt của mạng truyền thẳng hai lớp. Trong đó:

Hàm tổng ngõ vào tế bào thần kinh lớp ẩn là hàm cầu:

$$f = net = \frac{\left(X - Cj\right)^2}{b_j^2} - \theta \tag{2.24}$$

Hàm kích hoạt ngõ ra tế bào thần kinh lớp ẩn là hàm mũ:

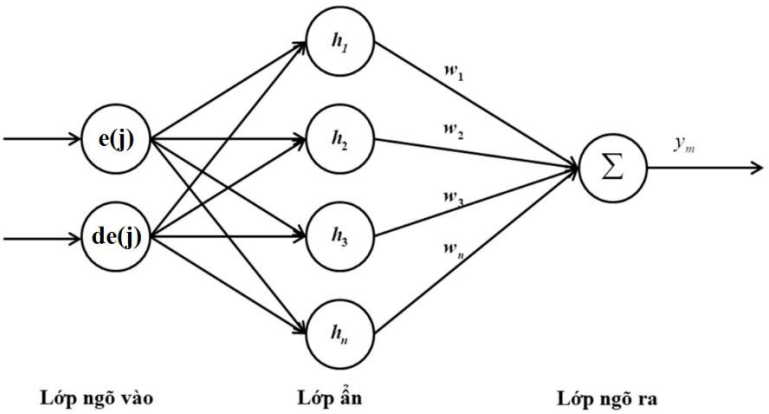
$$h_0\left(f\right) = e^{-f}$$

Hàm tổng ngõ vào tế bào thần kinh lớp ra là hàm tuyến tính:

$$f = net = W^T h_j - \theta$$

Hàm kích hoạt ngõ ra tế bào thần kinh lớp ra là hàm tuyến tính: $y_0(f) = f$

Mức ngưỡng của các tế bào thần kinh $\theta = 0$.



Hình 4. Cấu trúc mạng neural network RBF

Giả sử : $X = [x_1, x_2, ..., x_n]^T$ là vector ngõ vào , $H = [h_1, h_2, ..., h_m]^T$ là vector hàm kích hoạt, với h_j là hàm Gauss được xác định như sau:

$$h_j = \exp\left[\frac{-\|X - Cj\|^2}{2b_j^2}\right], j = 1, 2, ..., m \qquad (2.25)$$

Tâm của các hàm cơ sở tại nút j là $C_j = \left[c_{j1}, c_{j2}, c_{j3}, ..., c_{ji}, ...c_{jn}\right]^T$, $i = 1, 2, ..., n$. Giả sử độ rộng hàm cơ sở là $B = \left[b_1, b_2, b_3, ..., b_j, ...b_m\right]^T$, trong đó b_j là độ rộng hàm cơ sở của nút j và $b_j > 0$. Vector trọng số của mạng là $W = \left[w_1, w_2, w_3, ..., w_j, ...w_m\right]^T$. Đầu ra của mạng được xác định như sau:

$$y(k) = W^T h = w_1 h_1 + w_2 h_2 + ... + w_m h_m \qquad (2.26)$$

Với hàm lỗi là : $E = \frac{1}{2}\left(y_d(k) - y(k)\right)^2$, trong đó $y(k)$ là ngõ ra của hệ thống tại thời điểm thứ k và $y_d(k)$ là ngõ ra mong muốn.

2.5.2. Huấn luyện mạng neural network RBF

Thông qua cách học Gradient Descent, độ biến thiên trọng số ngõ ra $\Delta w_j(\mathrm{k})$, độ biến thiên trọng số giữa điểm nút cơ sở và điểm nút trung tâm $\Delta c_{ji}(\mathrm{k})$, độ biến thiến độ rộng tâm neural $\Delta b_j(\mathrm{k})$ được tính như sau:

$$\begin{cases} \Delta w_j(k) = \eta\left(y_d(k) - y(k)\right)\mathrm{h}_j \\ \Delta c_{ji}(k) = \eta\left(y_d(k) - y(k)\right)\mathrm{w}_j \dfrac{x_j - c_{ji}}{b_j^2} \\ \Delta b_j(k) = \eta\left(y_d(k) - y(k)\right)\mathrm{w}_j h_j \dfrac{\left\|X - C_j\right\|}{b_j^3} \end{cases} \tag{2.27}$$

Luật cập nhật các thông số của mạng như sau:

$$\begin{cases} w_j(\mathrm{k}) = w_j(\mathrm{k}-1) - \Delta w_j(\mathrm{k}) + \alpha(w_j(\mathrm{k}-1) - w_j(\mathrm{k}-2)) \\ c_{ji}(\mathrm{k}) = c_{ji}(\mathrm{k}-1) + \Delta c_{ji}(\mathrm{k}) + \alpha(c_{ji}(\mathrm{k}-1) - c_{ji}(\mathrm{k}-2)) \\ b_j(\mathrm{k}) = b_j(\mathrm{k}-1) + \Delta b_j(\mathrm{k}) + \alpha(b_j(\mathrm{k}-1) - b_j(\mathrm{k}-2)) \end{cases} \tag{2.28}$$

Trong đó: η là tốc độ học, α là hệ số động học.

Để đơn giản quá trình huấn luyện người ta thường tiến hành cách huấn luyện mạng RBF như sau:

Bước 1: Xác định tâm và độ rộng của các hàm cơ sở

Tâm của các hàm cơ sở c_{ji} (với $j = 1,2,...,m$ là số tâm neural; $i = 1,2,...,\mathrm{n}$ là số đầu vào) được xác định dựa vào tính hội tụ của các điểm dữ liệu đầu vào. Phân vector tín hiệu đầu vào thành j nhóm($j \leq$ số điểm dữ liệu đầu vào) khi đó điểm dữ liệu nào là trung tâm của mỗi nhóm(nghĩa là khoảng cách giữa tâm của nhóm đó đến các điểm còn lại trong nhóm là nhỏ nhất) chính là tâm c_{ji} cần tìm. Việc này có thể thực hiện bằng quan sát thực nghiệm hoặc sử dụng thuật toán phân cụm như K-Means clustering algorithm.

Độ rộng của tâm hàm cơ sở b_j có ảnh hưởng rất lớn đến chất lượng của mạng. Có nhiều cách để xác định giá trị này. Cách thông thường là dựa vào sự tương đồng giữa các nhóm dữ liệu vào. Đối với mỗi tế bào thần kinh ở lớp ẩn, khoảng cách từ tâm của nó đến tế bào thần kinh khác gần nhất chính là độ rộng b_j của tế bào thần kinh đó.

Bước 2: Huấn luyện trọng số lớp ra của mạng:

Thực hiện tuần tự như sau:

1. Chọn tốc độ học η và sai số cực đại $E_{\max}$.
2. Khởi động:
   - Gán sai số $E = 0$.
   - Gán biến chạy $k = \overline{1,K}$.
   - Gán các trọng số $w_{jq}(k)$, ($j = \overline{1,\mathrm{m}}$, m là số nhân neural; $q = \overline{1,\mathrm{p}}$, p là số đầu ra) bằng các giá trị ngẫu nhiên bất kì.
3. Tính ngõ ra của mạng với tín hiệu vào là $X(k)$:

$$h_j(k) = \exp\left[\frac{-\left\|X(k) - Cj\right\|^2}{2b_j^2}\right] \tag{2.29}$$

$$y_q(k) = W_q^T h_j(k) = w_{1q}h_1(k) + w_{2q}h_2(k) + ... + w_{mq}h_m(k) \tag{2.30}$$

4. Cập nhật trọng số lớp ra của mạng:

$$w_{jq}(k+1) = w_{jq}(k) + \eta\left(y_{qd}(k) - y_q(k)\right)h_j(k) \tag{2.31}$$

5. Tính sai số tích lũy:

$$E = E + \frac{1}{2}\sum_{q=1}^{p}\left(y_{qd}(k) - y_q(k)\right)^2 \tag{2.32}$$

6. Nếu $k < K$ thì gán $k = k+1$ và lặp lại từ 3.
   - Nếu $k = K$ thì tiếp tục.
7. Kết thúc một chu trình huấn luyện.
   - Nếu $E < E_{\max}$ thì kết thúc quá trình học.

- Nếu $E \geq E_{\max}$ thì gán $E = 0$, $k = 1$ và trở lại 3. bắt đầu một chu trình huấn luyện mới.

Mạng RBF có khả năng xấp xỉ hàm rất tốt và có thể huấn luyện dễ dàng nhanh chóng. Khả năng xấp xỉ chính xác và tốc độ của mạng RBF còn có thể được cải thiện hơn nữa bằng việc chọn tâm và độ rộng phù hợp cho các hàm cơ sở ở lớp ẩn. Tuy nhiên, mạng RBF thường đáp ứng chậm trong giai đoạn nhớ do lượng lớn tế bào thần kinh ở lớp ẩn.

Kết luận chương 2

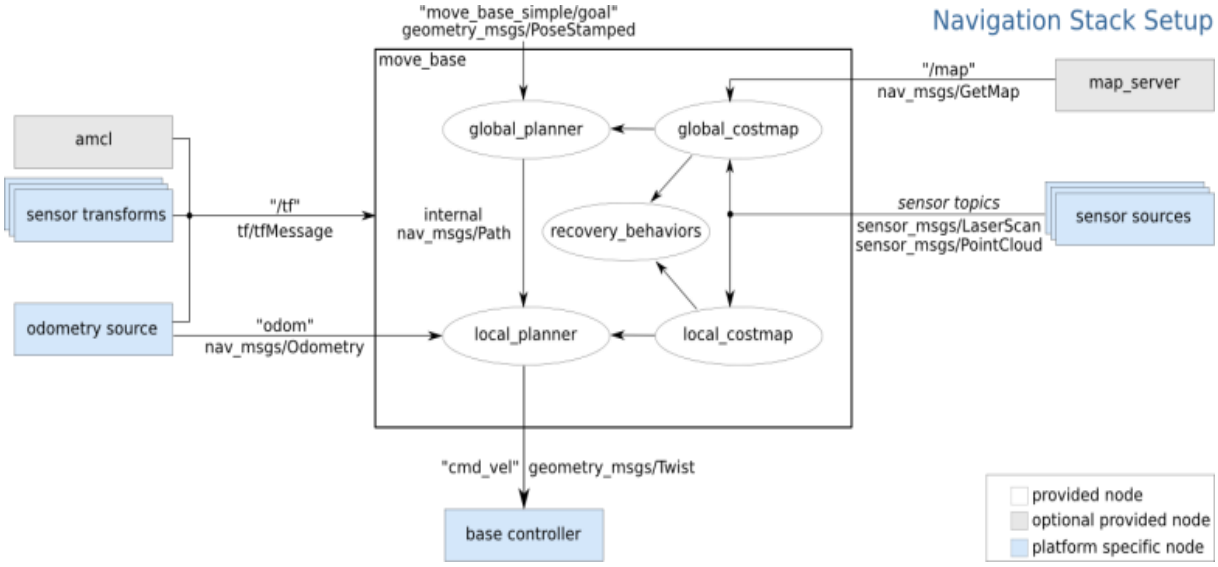
Chương 2 đã trình bày những cơ sở lý thuyết nền tảng cần thiết để triển khai hệ thống điều khiển robot di động chịu nhiễu bất định. Từ mô hình động học cho robot vi sai và động cơ DC, đến học tăng cường – đặc biệt là TD3, cùng với điều khiển tiền định thời gian và mạng RBFNN – các nội dung này cung cấp khung lý thuyết vững chắc để phát triển bộ điều khiển thích nghi thông minh trong các chương tiếp theo.

CHƯƠNG 3: THIẾT KẾ BỘ ĐIỀU KHIỂN CHO ROBOT DI ĐỘNG CHỊU NHIỄU BẤT ĐỊNH DỰA TRÊN HỌC TĂNG CƯỜNG

3.1. Cấu trúc bộ điều khiển vị trí cho robot di động

Bộ điều khiển vị trí là một trong những thành phần quan trọng nhất trong hệ thống điều khiển của robot di động, đảm nhiệm vai trò xác định và duy trì vị trí mong muốn của robot trong không gian hoạt động. Việc thiết kế một bộ điều khiển vị trí hiệu quả không những đảm bảo robot di chuyển chính xác đến vị trí mục tiêu, mà còn giúp nâng cao độ ổn định và độ tin cậy của hệ thống trong các điều kiện môi trường thay đổi.

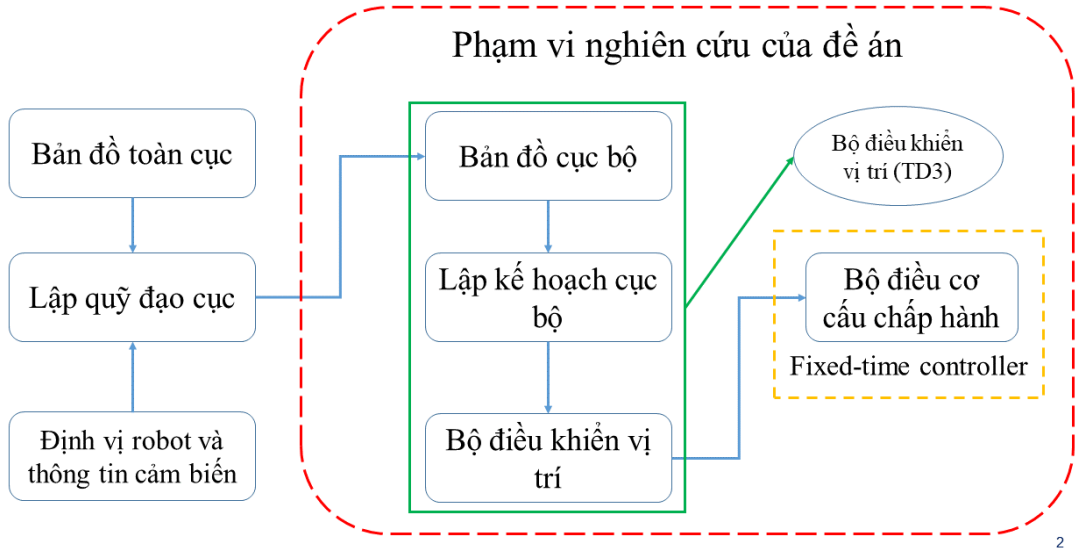
Một phương pháp chung cho vấn đề điều khiển robot di động được cộng đồng ROS đề xuất được mô tả như sau:



Hình 3.1. Khung điều hướng robot di động [67]

Trong đề án này, phương pháp TD3 được sử dụng như một bộ điều khiển vị trí bậc cao nhằm thay thế chức năng của bản đồ địa phương và lập quỹ đạo địa phương. Nhiệm vụ của bộ điều hướng TD3 nhằm nhận tín hiệu vị trí mong muốn và gửi tín hiệu điều khiển vận tốc cho mỗi bánh làm tham chiếu cho bộ điều khiển cơ cấu chấp hành sử dụng phương pháp điều khiển định thời thích nghi RBF, cấu

trúc bộ điều khiển vị trí và bộ điều khiển cơ cấu chấp hành được mô tả trong Hình 3.2.

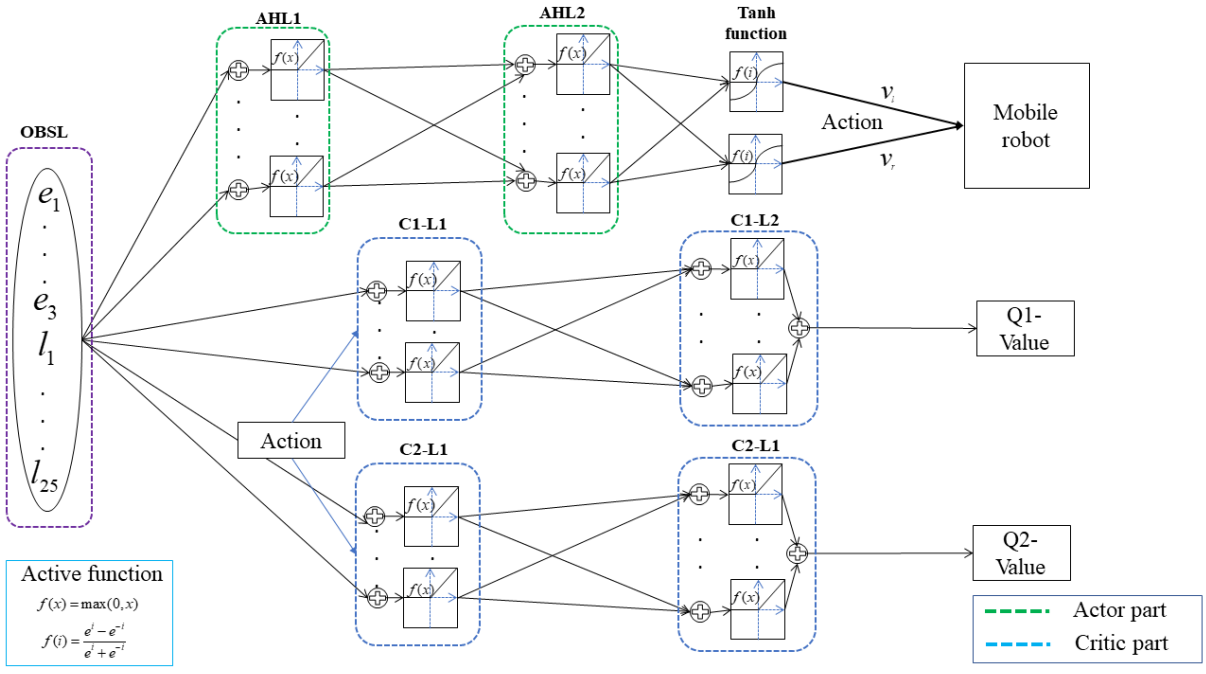


Hình 3.2. Cấu trúc bộ điều khiển vị trí và bộ điều khiển cơ cấu chấp hành

3.2. Xây dựng bộ điều hướng robot dựa trên phương pháp học tăng cường

3.2.1. Thiết kế cấu trúc mạng cho tác nhân TD3

Bộ điều khiển điều hướng robot được xây dựng trên giải thuật TD3, cấu trúc mạng bao gồm một mạng hành vi và hai mạng đánh giá được mô tả tại Hình 6.



Hình 3.3. Cấu trúc tác nhân TD3 gồm một mạng hành vi và hai mạng đánh giá

Trong đó, lớp trạng thái đầu vào gồm sai số vị trí (3.1) và tín hiệu laser đo khoảng cách từ robot đến vật cản môi trường:

$$\begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix} = \begin{bmatrix} x_{ref} - x & y_{ref} - y & \theta_{ref} - \theta \end{bmatrix} \quad (3.1)$$

Với $x_{ref}, y_{ref}, \theta_{ref}$ là vị trí điểm đến mong muốn, x, y, θ là vị trí hiện tại của robot.

Vector laser được mô tả gồm 25 khoảng cách thu được từ robot $\begin{bmatrix} l_1 & . & . & l_{25} \end{bmatrix}$.

Các hàm kích hoạt được sử dụng tại mỗi lớp ẩn là làm ReLU với công thức $f(x) = \max(0, x)$.

Tại đầu ra của mạng hành vi, sử dụng 2 tín hiệu đầu ra gồm $\begin{bmatrix} v & \omega \end{bmatrix}$ lần lượt là vận tốc tịnh tiến và vận tốc xoay của robot. Tại lớp đầu ra hàm kích hoạt sử dụng hàm Tanh $f(i) = \frac{e^i - e^{-i}}{e^i + e^{-i}}$ với hệ số nhân là 1 nhằm đảm bảo tín hiệu điều khiển robot bị chặn trong khoảng $\begin{bmatrix} -1 & 1 \end{bmatrix}$ cho cả vận tốc tịnh tiến và vận tốc xoay.

3.2.2. Thiết kế hàm phần thưởng cho tác nhân TD3

Trong thuật toán TD3, hàm phần thưởng giữ vai trò trung tâm trong việc dẫn dắt quá trình học của tác nhân. Cụ thể, phần thưởng đóng vai trò như tín hiệu phản hồi từ môi trường, giúp tác nhân đánh giá mức độ hiệu quả của hành động tại mỗi trạng thái. Do TD3 là một thuật toán học tăng cường theo hướng chính sách xác định, tác nhân học cách tối ưu hóa chuỗi hành động liên tục nhằm tối đa hóa tổng phần thưởng tích lũy trong dài hạn. Do đó, thiết kế hàm phần thưởng có ảnh hưởng trực tiếp đến hành vi hội tụ của chính sách và chất lượng điều khiển cuối cùng. Một hàm phần thưởng rõ ràng, định hướng đúng mục tiêu sẽ giúp tác nhân nhanh chóng học được chiến lược tối ưu; ngược lại, nếu phần thưởng bị nhiễu, không tương quan với mục tiêu điều khiển, hoặc không đủ phân biệt giữa các hành vi tốt và xấu, thì quá trình học có thể dẫn đến hội tụ sai lệch, dao động hoặc thất bại. Đặc biệt trong điều khiển robot di động, phần thưởng cần phản ánh đồng thời các tiêu chí như độ chính xác vị trí, hướng di chuyển và tránh va chạm, từ đó tạo ra tác nhân thông minh có khả năng hoạt động hiệu quả trong môi trường thực tế phức tạp.

Từ những luận điểm nêu trên, trong đề án này hàm phần thưởng được thiết kế gồm 2 thành phần r_1 và r_2 như sau:

$$r = \alpha_1 r_1 + \alpha_2 r_2 \tag{3.2}$$

Với α_1, α_2 là hằng số luôn dương.

Thành phần phần thưởng r_1 được thiết kế với mục tiêu đánh giá robot đạt được tới gần điểm đích hay chưa theo công thức:

$$r_1 = -\left(\beta_1 e_1^2 + \beta_2 e_2^2 + \beta_3 e_3^2\right) \tag{3.3}$$

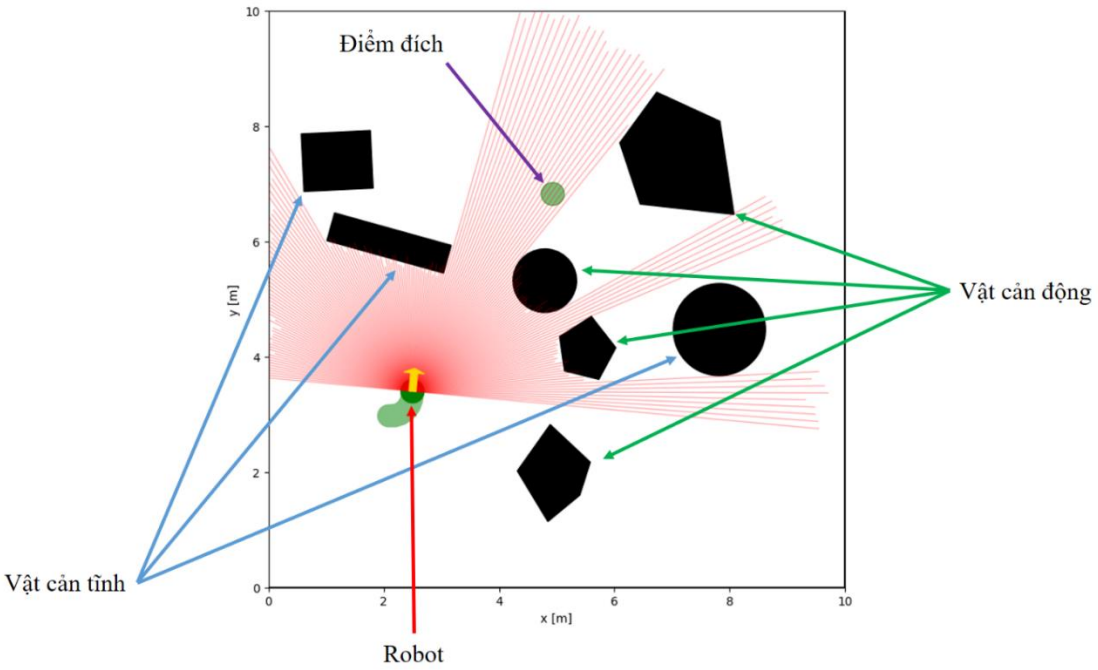
Với $\beta_1, \beta_2, \beta_3$ là hằng số luôn dương

Thành phần r_2 được thiết kế nhằm đánh giá robot có quá gần vật cản với ngưỡng được đặt ra không theo công thức:

$$r_2 = -\max\left((d_i < d_{\min})(d_{\min} - d_i)\right) \tag{3.4}$$

Với $d_{\min}$ là khoảng cách gần nhất cho phép từ robot đến vật cản, d_i là khoảng cách từ robot đến vật cản được đo với tia laser thứ i.

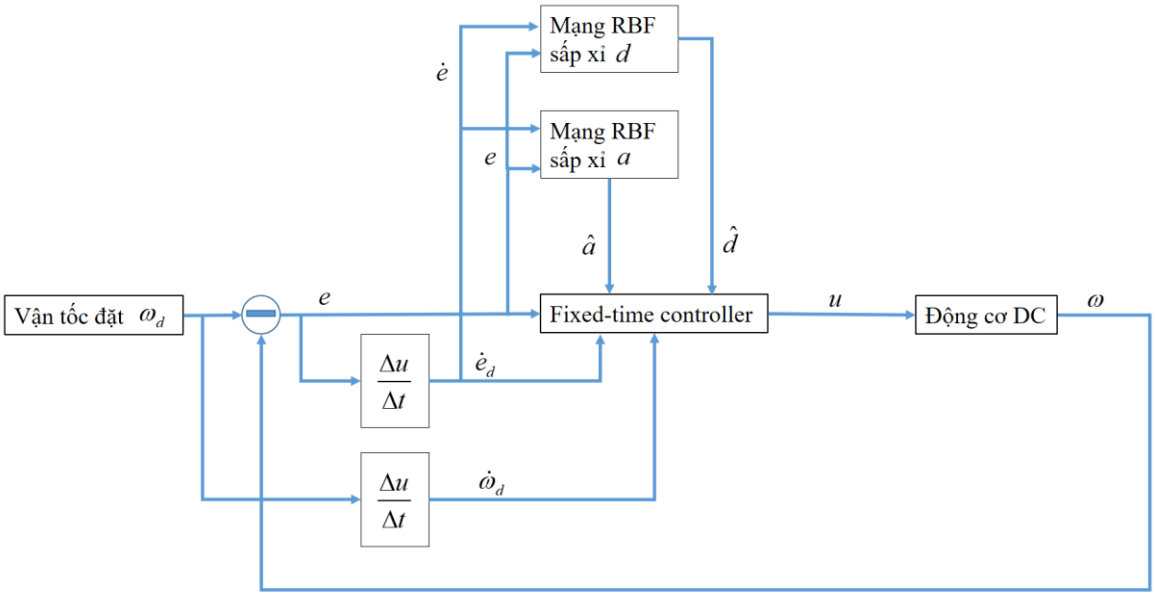
Mô phỏng được thực hiện trên Python với môi trường thử nghiệm gồm 7 vật cản, trong đó gồm 4 vật cản động và 3 vật cản tĩnh như sau:



Hình 3.4. Thiết lập môi trường mô phỏng

3.3. Thiết kế bộ điều khiển cho cơ cấu chấp hành sử dụng mạng RBF thích nghi

Trong hệ thống robot di động, cơ cấu chấp hành là thành phần đảm nhận chức năng biến đổi tín hiệu điều khiển thành chuyển động cơ học thực tế, như điều khiển vận tốc bánh xe, góc lái, hoặc mô-men xoắn tại các khớp truyền động. Để đảm bảo hiệu năng hoạt động chính xác, ổn định và tin cậy của robot trong các điều kiện môi trường thay đổi, việc thiết kế một bộ điều khiển thích nghi hiệu quả cho cơ cấu chấp hành là điều thiết yếu. Trong bối cảnh đó, phương pháp điều khiển thích nghi cố định thời gian đã nổi lên như một công cụ mạnh mẽ, đảm bảo tính hội tụ trong thời gian hữu hạn và độc lập với điều kiện ban đầu, mang lại nhiều lợi thế vượt trội trong ứng dụng robot di động hiện đại. Lưu đồ thuật toán điều khiển thời gian xác định thích nghi dựa trên mạng RBF xấp xỉ tham số được mô tả trong Hình 3.5.



Hình 3.5. Cấu trúc bộ điều khiển thích nghi RBF dựa trên bộ điều khiển thời gian xác định.

Xét động cơ DC được mô hình hóa theo (2.10) và (2.11). Tín hiệu điều khiển sử dụng là điện áp u và đầu ra điều khiển là vận tốc động cơ ω. Do trong thực tế, giá trị cảm kháng của động cơ là rất nhỏ ($L \approx 0$) nên trong quá trình thiết kế bộ điều khiển cảm kháng được bỏ qua.

Đặt $a=\frac{K_t K_b+BR}{JR}, b=\frac{K_t}{JR}, d=\frac{T_L}{J}$ trong đó a,d là hai thành phần không xác định, biến đổi theo điều kiện hoạt động của hệ thống. Thay a,b,d vào phương trình (2.11) phương trình động lực học của động cơ trở thành:

$$\dot{\omega}=-a\omega+bu-d \tag{3.5}$$

Phương trình sai số trạng thái của hệ thống được định nghĩa như sau:

$$e=\omega_d-\omega \tag{3.6}$$

Với ω_d là tín hiệu vận tốc đặt cho hệ thống.

Đạo hàm theo thời gian phương trình (3.6) ta được:

$$\dot{e}=\dot{\omega}_d+a\omega-bu+d=-ae+d^{'}-bu \tag{3.7}$$

Với $d^{'}=a\omega_d+d+\dot{\omega}_d$

Do trong thực tế, giá trị a,d là bất định do có tham số của ma sát và momen tải ngoại lực tác dụng lên trục động cơ. Vì vậy trong nghiên cứu này, 2 mạng nơ ron RBF được sử dụng để ước tính giá trị a và d.

Cấu trúc 2 mạng nơ ron RBF gồm 2 nơ ron tại lớp đầu vào ứng với giá trị $\begin{bmatrix} e & \dot{e} \end{bmatrix}$. Lớp ẩn sử dụng 5 nơ ron với hàm kích hoạt theo phương trình (2.29). Mỗi mạng ước tính một đầu ra $\hat{a}$ và $\hat{d}$ ứng với giá trị a và d của hệ thống. Từ đó ta có phương trình $\hat{d}^{'}=\hat{a}\omega_d+\hat{d}+\dot{\omega}_d$.

Giá trị a được xấp xỉ từ mạng nơ ron RBF có dạng:

$$\hat{a}=\mathrm{W}_a^T h(k) \tag{3.8}$$

Giá trị d được xấp xỉ từ mạng nơ ron RBF có dạng:

$$\hat{d}=\mathrm{W}_d^T h(k) \tag{3.9}$$

Đặt $\tilde{a}=a-\hat{a}$ là sai số của giá trị thực tế và giá trị ước tính bởi mạng nơ ron RBF cho giá trị a.

Đặt $\tilde{d} = d - \hat{d}$ là sai số giá trị thực tế so với ước tính của mạng nơ ron cho giá trị d. Từ đó ta có $\tilde{d}^{'} = \tilde{a}\omega_d + \tilde{d}$.

Tương tự ta đặt $\tilde{\mathrm{W}}_a = \mathrm{W}_a - \mathrm{W}_a^*, \tilde{\mathrm{W}}_d = \mathrm{W}_d - \mathrm{W}_d^*$ là sai số của ma trận trọng số của mạng nơ ron với giá trị tối ưu W^* đại diện cho mạng sấp xỉ được hoàn toàn tham số.

Ta có phương trình xấp xỉ sai số của a:

$$\tilde{a} = -\tilde{\mathrm{W}}_a^T h(k) \tag{3.10}$$

Tương tự ta có phương trình sấp xỉ sai số của d:

$$\tilde{d} = -\tilde{\mathrm{W}}_d^T h(k) \tag{3.11}$$

Đặt $z = \int_0^t e_\tau d\tau$ ta có $\dot{z} = e$.

Để xác định luật cập nhập cho mạng nơ ron, hàm Lyapunov được chọn như sau:

$$V = \frac{1}{2}e^2 + \frac{1}{2}k_i z^2 + \frac{1}{2\gamma_a}\tilde{\mathrm{W}}_a^T\tilde{\mathrm{W}}_a + \frac{1}{2\gamma_d}\tilde{\mathrm{W}}_d^T\tilde{\mathrm{W}}_d \tag{3.12}$$

Đạo hàm Lyapunov theo thời gian ta được:

$$\dot{V} = e\dot{e} + k_i z\dot{z} + \frac{1}{\gamma_a}\tilde{W}_a^T\dot{W}_a + \frac{1}{\gamma_d}\tilde{W}_d^T\dot{W}_d \tag{3.13}$$

Thay (3.7) vào (3.13) phương trình (3.13) trở thành:

$$\dot{V} = e(-ae + d^{'} - bu) + k_i z\dot{z} + \frac{1}{\gamma_a}\tilde{W}_a^T\dot{W}_a + \frac{1}{\gamma_d}\tilde{W}_d^T\dot{W}_d \tag{3.14}$$

Do a,d là không xác định, tín hiệu điều khiển u được chọn như sau:

$$u = \frac{1}{b}\left(-\hat{a}e + \hat{d}^{'} + k_1 \tanh(\kappa e)|e|^{\alpha} + k_2 \tanh(\kappa e)|e|^{\beta} + k_i z\right) \tag{3.15}$$

Thay (3.15) vào (3.14) ta được:

$$\dot{V} = \begin{bmatrix} e\left(\tilde{W}_a^T h(k)e - \tilde{W}_a^T h(k)\omega_d - \tilde{W}_d^T h(k) - k_1 \tanh(\kappa e)|e|^{\alpha} - k_2 \tanh(\kappa e)|e|^{\beta} - k_i z\right) + \\ k_i z e + \frac{1}{\gamma_a}\tilde{W}_a^T \dot{W}_a + \frac{1}{\gamma_d}\tilde{W}_d^T \dot{W}_d \end{bmatrix}$$

(3.16)

Chọn hàm cập nhập trọng số có dạng:

$$\begin{aligned} \dot{W}_a &= \gamma_a h(k) e^2 \\ \dot{W}_d &= \gamma_d h(k) e \end{aligned} \tag{3.17}$$

Thay (3.17) vào (3.16) ta được:

$$\begin{aligned} \dot{V} &= -k_1 e \tanh(\kappa e)|e|^{\alpha} - k_2 e \tanh(\kappa e)|e|^{\beta} \\ &= -k_1 2^{\frac{1+\alpha}{2}} V^{\frac{1+\alpha}{2}} - k_2 2^{\frac{1+\beta}{2}} V^{\frac{1+\beta}{2}} \le 0 \end{aligned} \tag{3.18}$$

Từ phương trình (3.18) hệ thống đảm bảo ổn định Lyapunov.

Đặt $p = \frac{1+\alpha}{2}$, q=$\frac{1+\beta}{2}$, $a = k_1 2^{\frac{1+\alpha}{2}}$, $b = k_2 2^{\frac{1+\beta}{2}}$ ta có phương trình (3.18) trở thành

$$\dot{V} \le -aV^p - bV^q \tag{3.19}$$

Theo (2.23) ta có thời gian đáp ứng T của hệ được giới hạn định trước theo công thức:

$$T \le \frac{1}{k_1 2^{\frac{1+\alpha}{2}}(1-p)} + \frac{1}{k_2 2^{\frac{1+\beta}{2}}(q-1)} = \frac{2}{k_1 2^{\frac{1+\alpha}{2}}(1-\alpha)} + \frac{2}{k_2 2^{\frac{1+\beta}{2}}(\beta-1)} \tag{3.20}$$

Kết luận chương 3

Chương 3 đã trình bày chi tiết các thiết lập cho công việc đào tạo tác nhân học tăng cường TD3. Bộ điều khiển cơ cấu chấp hành sử dụng phương pháp điều khiển thời gian định trước thích nghi cũng được trình bày cụ thể bao gồm quá trình thiết kế, chứng minh ổn định. Qua đó làm cơ sở cho quá trình mô phỏng, đánh giá tại chương tiếp theo.

CHƯƠNG 4: KẾT QUẢ VÀ ĐÁNH GIÁ BỘ ĐIỀU KHIỂN

4.1. Thiết lập các tham số mô phỏng

Bộ điều khiển điều hướng robot dựa trên học tăng cường TD3 được triển khai với số lượng nơ ron mỗi lớp tại Bảng 4-1. Số lượng nơ ron lớp OBSL là số lượng đầu vào gồm 3 đầu vào sai số trạng thái và 25 đầu vào là các giá trị đo khoảng cách laser của lidar. Số lượng nơ ron tại các lớp được lựa chọn theo phương pháp thử sai lấy giá trị lớn đảm bảo khả năng sấp xỉ của mạng nơ ron.

Bảng 4-1. Số lượng nơ ron tại các lớp của tác nhân

Tên lớp	Ý nghĩa	Số lượng
OBSL	Lớp quan sát	28
AHL1	Lớp hành vi số 1	400
AHL2	Lớp hành vi số 2	300
C1-L1	Lớp đánh giá 1 số 1	400
C1-L2	Lớp đánh giá 1 số 2	300
C2-L1	Lớp đánh giá 2 số 1	400
C2-L2	Lớp đánh giá 2 số 2	300

Giá trị các tham số của hàm phần thưởng (3.2) được lựa chọn tại Bảng 4-2. Do robot cần ưu tiên tránh vật cản hơn so với đến đích nên giá trị của α_1 được chọn nhỏ hơn so với α_2. Đồng thời robot ưu tiên đạt được điểm đích theo phương x và y so với đạt được góc đặt θ_d nên giá trị β_1, β_2 được chọn lớn hơn rất nhiều so với β_3. Giá trị $d_{\min}$ được chọn có giá trị 2 (m) nhằm đảm bảo 2(m) là vùng an toàn của robot, khi có vật cản di chuyển vào khu vực 2(m) bán kính quanh robot, điểm phạt sẽ được kích hoạt nhằm hướng robot tránh xa vật cản như được mô tả tại phương trình (3.4).

Bảng 4-2. Giá trị các tham số của hàm phần thưởng

Tên tham số	Giá trị tham số
α_1	0.1
α_2	10
β_1	1
β_2	1
β_3	0.001
$d_{\min}$	2

Giá trị tham số động cơ DC sử dụng cho mô phỏng được trình bày tại Bảng 4-3. Do giả định giá trị bỏ qua giá trị cảm kháng của động cơ nên L được chọn bằng 0. Momen ngoại lực tác động lên động cơ tại thời điểm ban đầu được chọn có giá trị là 0.1 và thay đổi đột ngột trong quá trình hoạt động của robot đại diện cho thời điểm robot chịu tải ngẫu nhiên.

Bảng 4-3. Giá trị tham số động cơ DC

Ký hiệu	Ý nghĩa	Giá trị
R	Trở kháng của động cơ (Ohm)	1
L	Cảm kháng của động cơ (H)	0
K_e	Suất điện động ngược (V.s/rad)	0.1
K_t	Hằng số momen xoắn (N.m/A)	0.1
B	Ma sát của trục động cơ (N.m.s/rad)	0.0001

T_L	Momen ngoại lực (N.m)	0.1
J	Momen quán tính trục đông cơ ($kg.m^2$)	0.01

Các giá trị của phương trình luật điều khiển thời gian định trước thích nghi sử dụng phương pháp RBF (3.15) được qua thử sai mô tả tại Bảng 4-4. Các giá trị k_1, k_2, α, β trực tiếp ảnh hưởng tới thời gian đáp ứng đinh trước của bộ điều khiển lý do chọn giá trị được giải thích tại phần tiếp theo. Trong phương pháp điều khiển thời gian xác định trước (2.22) hàm $sign()$ được sử dụng phổ biến, tuy nhiên do sự không tuyến tính của hàm $sign()$ nên gây ra hiện tượng chattering cho tín hiệu điều khiển. Vấn đề này được xử lý trong đề án bằng cách sử dụng hàm $\tanh()$ với giá trị κ phù hợp để sấp xỉ hàm $sign()$ và đảm bảo sự mượt mà liên tục của tín hiệu điều khiển.

Bảng 4-4. Giá trị tham số bộ điều khiển thời gian định trước thích nghi

Tham số	Giá trị
k_1	10
k_2	10
κ	10
k_i	5
γ_a	0.1
γ_d	10
α	0.5
β	1.5

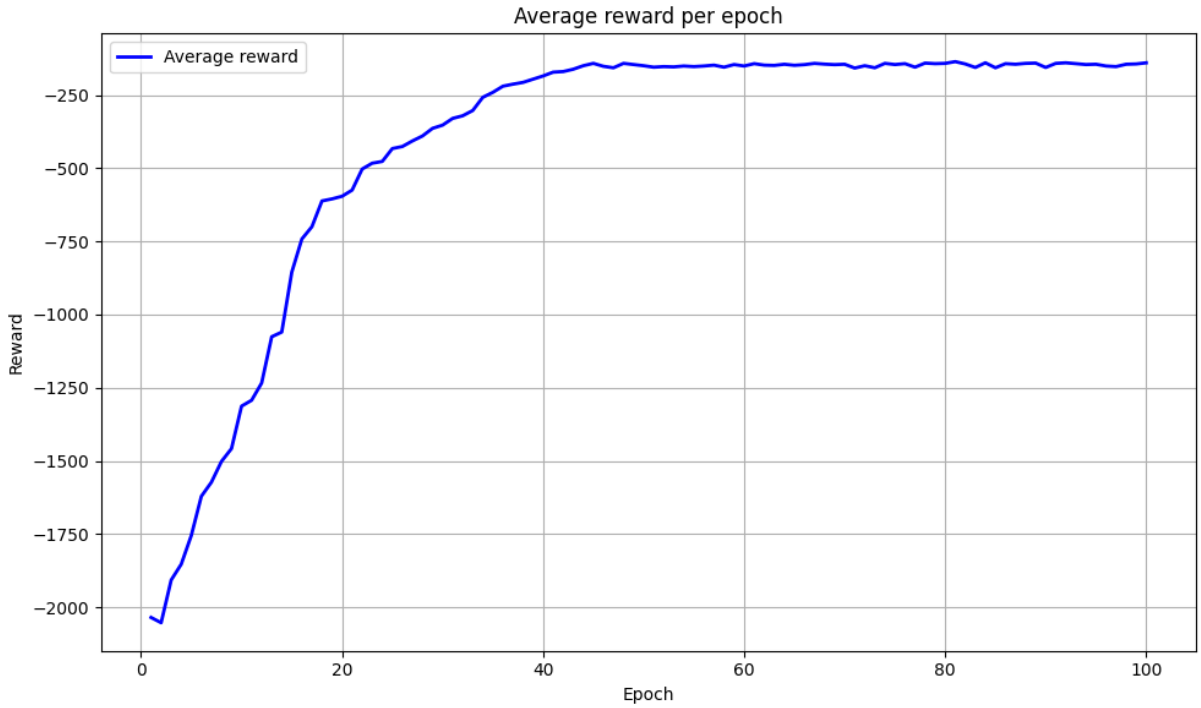
Các tham số mô phỏng về thời gian lấy mẫu, thời gian tối đa và các giá trị cài đặt cho đào tạo tác nhân TD3 được trình bày tại Bảng 4-5. Trong đó giá trị nhiễu ngẫu nhiên tác động lên tín hiệu điều khiển ρ được chọn qua phương pháp thử sai, nhằm đảm bảo robot có thể khám phá các chính sách mới mở rộng khả năng tìm kiếm dữ liệu về môi trường, tối ưu kinh nghiệm.

Bảng 4-5. Giá trị các tham số thực hiện cho mô phỏng

Ký hiệu	Ý nghĩa	Giá trị
t_s	Thời gian lấy mẫu	0.005 (s)
$t_{\max}$	Thời gian mô phỏng tối đa	5 (s)
Epoch	Số kỷ nguyên training	100
Episode	Số lần lặp cho mỗi kỷ nguyên	70
B	Kích thước bộ nhớ đệm batch size	64
τ	Hệ số cập nhập trọng số	0.005
γ	Hệ số chiết khấu điểm thưởng tương lai	0.99
ρ	Nhiễu khám phá tác ngẫu nhiên tác động lên mạng hành vi	0.2
$v_{\max}$	Tốc độ tịnh tiến tối đa của robot	1
$\omega_{\max}$	Tốc độ xoay tối đa của robot	0.5

4.2. Kết quả mô phỏng điều hướng robot dựa trên bộ điều khiển học tăng cường TD3

Kết quả training của tác nhân được mô tả tại Hình 4.1.



Hình 4.1. Hàm điểm thưởng trung bình tại mỗi kỷ nguyên

Kết quả huấn luyện thể hiện quá trình học của tác nhân sử dụng thuật toán TD3 trong bài toán điều hướng robot di động thông qua trung bình phần thưởng theo từng epoch. Có thể quan sát thấy rằng, trong giai đoạn đầu (khoảng từ epoch 0 đến epoch 30), phần thưởng trung bình tăng mạnh từ giá trị rất thấp (khoảng -2000) lên gần -300, phản ánh quá trình học ban đầu diễn ra nhanh chóng khi tác nhân bắt đầu khám phá và khai thác các hành vi điều khiển hiệu quả hơn. Giai đoạn này cho thấy TD3 có khả năng thích nghi tốt với môi trường điều hướng và dần dần học được chiến lược điều khiển tối ưu để cải thiện hiệu suất. Từ khoảng epoch 40 trở đi, đường phần thưởng bắt đầu tiến vào vùng ổn định, dao động nhẹ quanh giá trị xấp xỉ -250, cho thấy chính sách đã hội tụ và tác nhân đạt được hiệu năng ổn định trong việc điều hướng robot. Việc phần thưởng duy trì ổn định trong suốt giai đoạn cuối phản ánh tính ổn định và độ tin cậy cao của thuật toán TD3 trong việc giải quyết các bài toán điều khiển liên tục có không gian trạng thái và

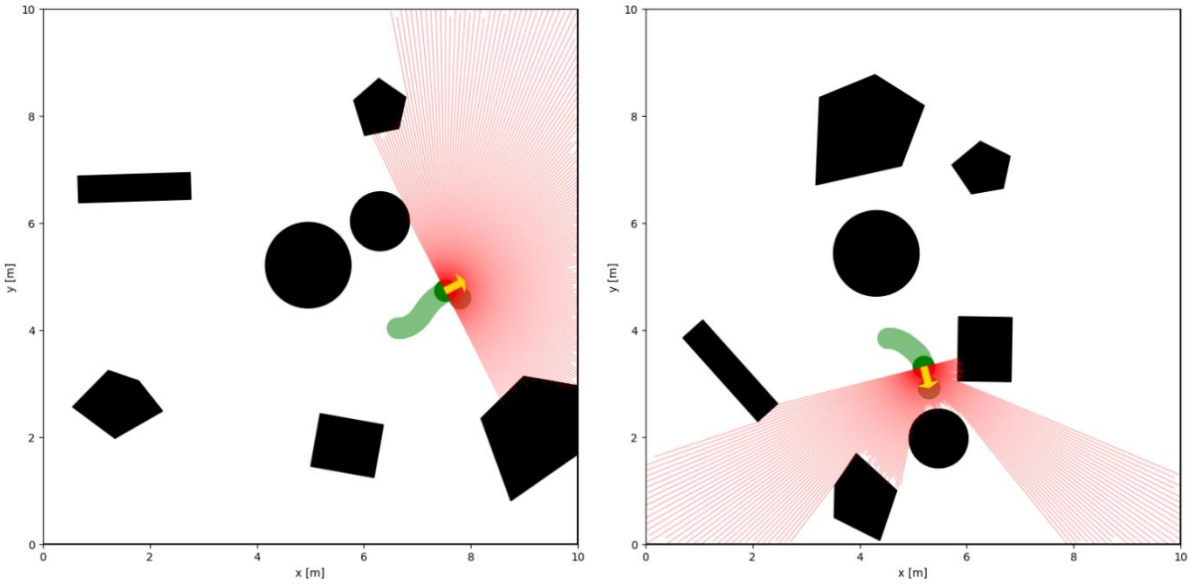
hành động phức tạp. Tổng thể, đồ thị minh chứng rõ ràng cho khả năng học hiệu quả và hội tụ nhanh của TD3 khi được triển khai trong hệ thống robot di động

Để thử nghiệm kết quả đạt được, trong đề tài này một số kịch bản được sử dụng gồm điểm đích được đặt gần với điểm xuất phát, giữa điểm đích không có vật cản, đây được coi là các kịch bản đơn giản. Kịch bản phức tạp gồm điểm đích nằm xa điểm đặt, giữa điểm đích và điểm đặt có vật cản di chuyển ngẫu nhiên. Quỹ đạo đạt được của robot trong một số trường hợp thử nghiệm đơn giản tại Hình 4.2. Thử nghiệm nâng cao trong một số trường hợp phức tạp được mô tả tại Hình 4.3.

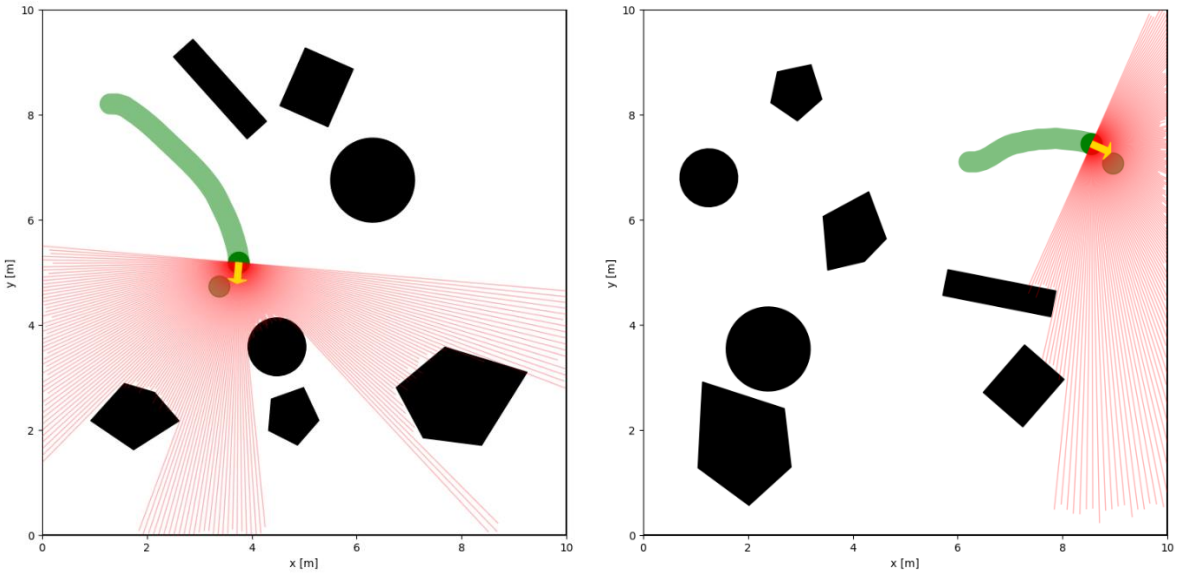
Kết quả thử nghiệm cho các trường hợp tại Hình 4.2 và Hình 4.3 cho thấy robot hoàn toàn đạt được vị trí mong muốn trong khi tránh va chạm với môi trường.

Ngoài ra còn một số kịch bản khiến robot va chạm với vật cản có thể kể đến như tại Hình 4.4 (a) và (b). Trong trường hợp (a) robot được khởi tạo quá gần vật cản, va chạm ngay lập tức xảy ra khi robot di chuyển một khoảng rất nhỏ. Trong trường hợp (b), vật cản động di chuyển hướng về phía điểm đích cùng lúc với robot, tại vị trí quá gần điểm đích, robot sẽ ưu tiên tiến đến đích gây va chạm với vật cản.

(a) (b)

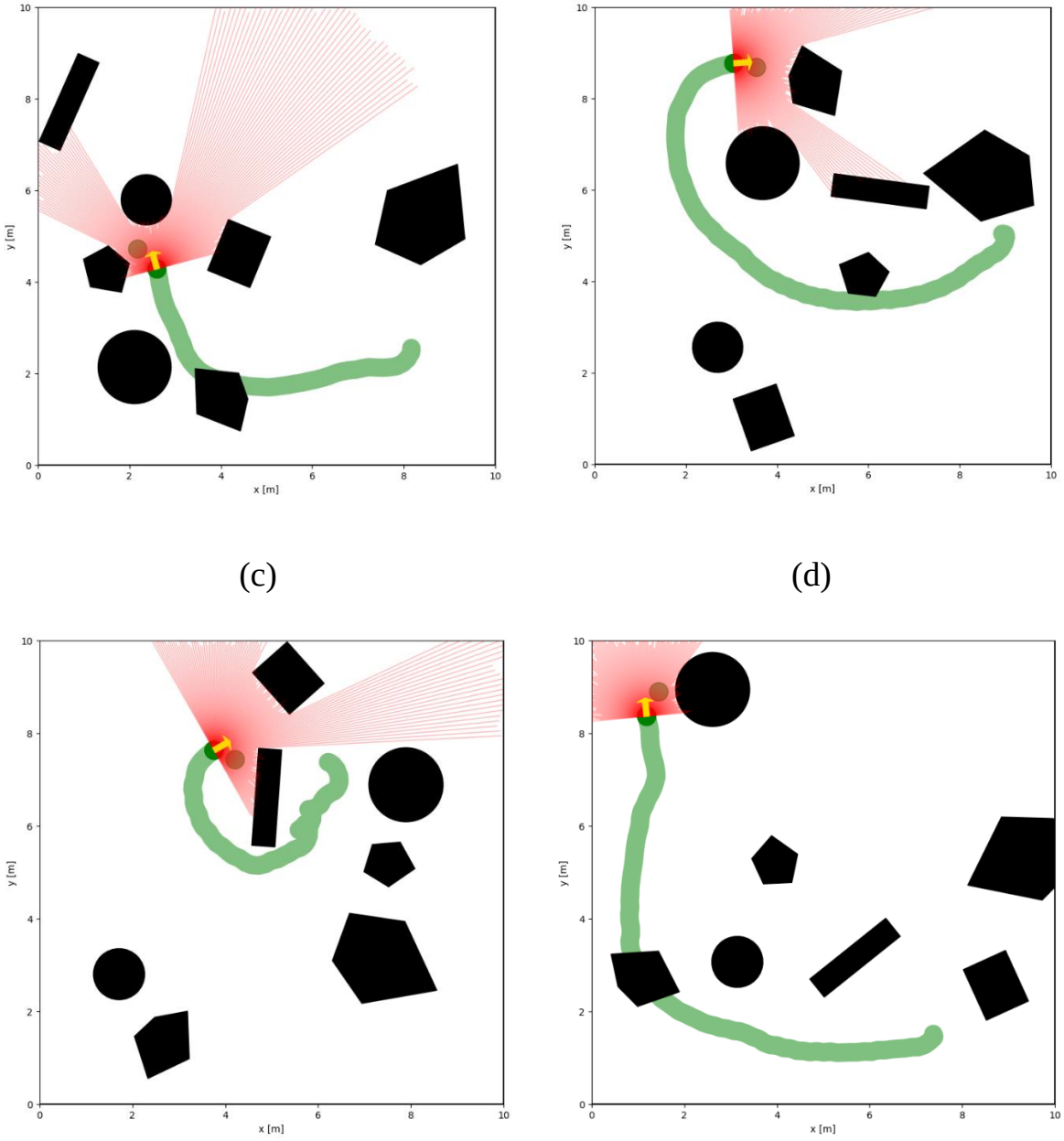


(c) (d)



Hình 4.2. Quỹ đạo robot đạt được trong một số kịch bản giữa điểm bắt đầu và kết thúc không có vật cản

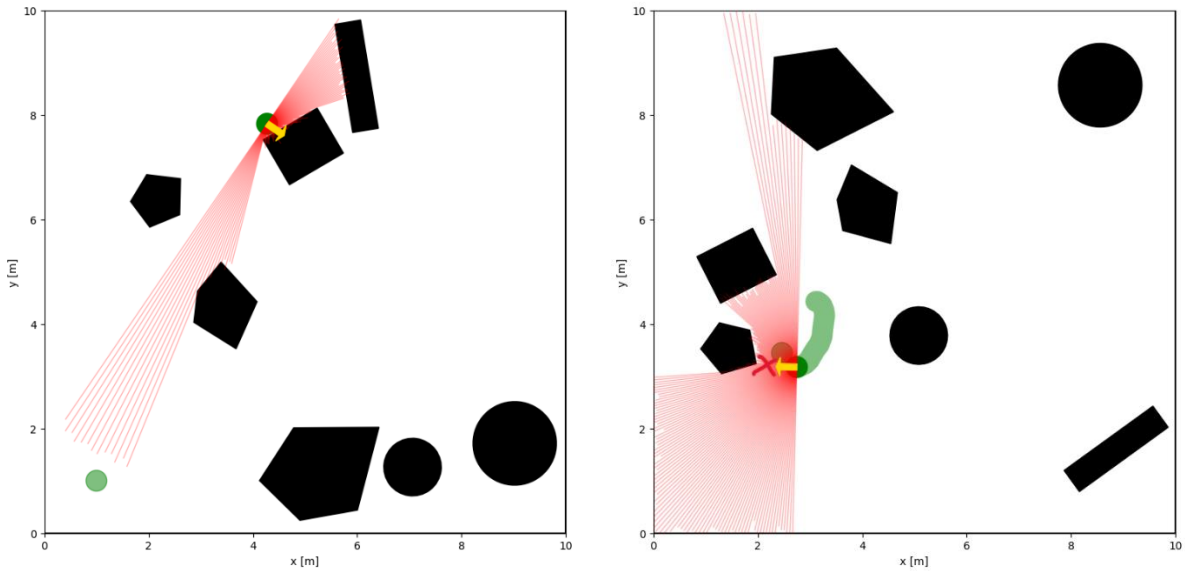
(a) (b)



(c) (d)

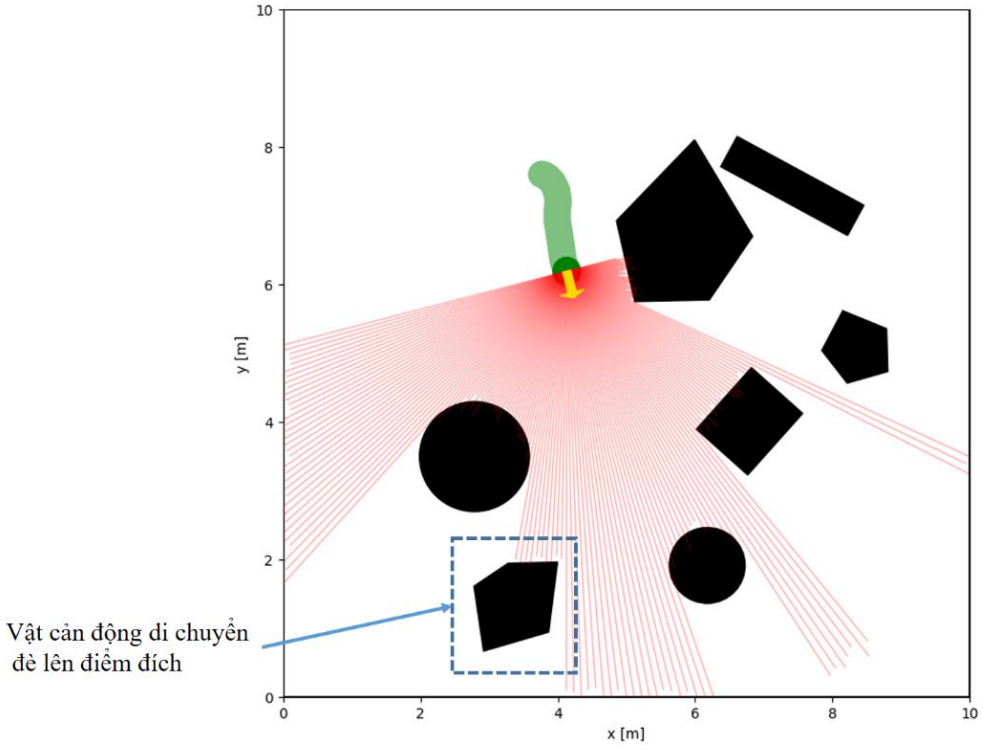
Hình 4.3. Quỹ đạo di chuyển của robot trong một số kịch bản vật cản phức tạp

(a) (b)



Hình 4.4. Một số trường hợp va chạm

Một số trường hợp đặc biệt robot không thể đến đích đó là khi điểm đích bị bao quanh hoặc bị ghi đè bởi vật cản động như được mô tả tại Hình 4.5.



Hình 4.5. Vật cản chiếm hữu vị trí điểm đặt

Kết quả cho thấy bộ điều khiển điều hướng dựa trên phương pháp TD3 hoàn toàn đạt được mục tiêu trong môi trường vật cản di động, quá trình thử cho thấy robot có khả năng đạt được điểm đặt mà không cần quá trình thực hiện quy hoạch đường đi cục bộ và không cần thông tin map cục bộ như các phương pháp truyền thống.

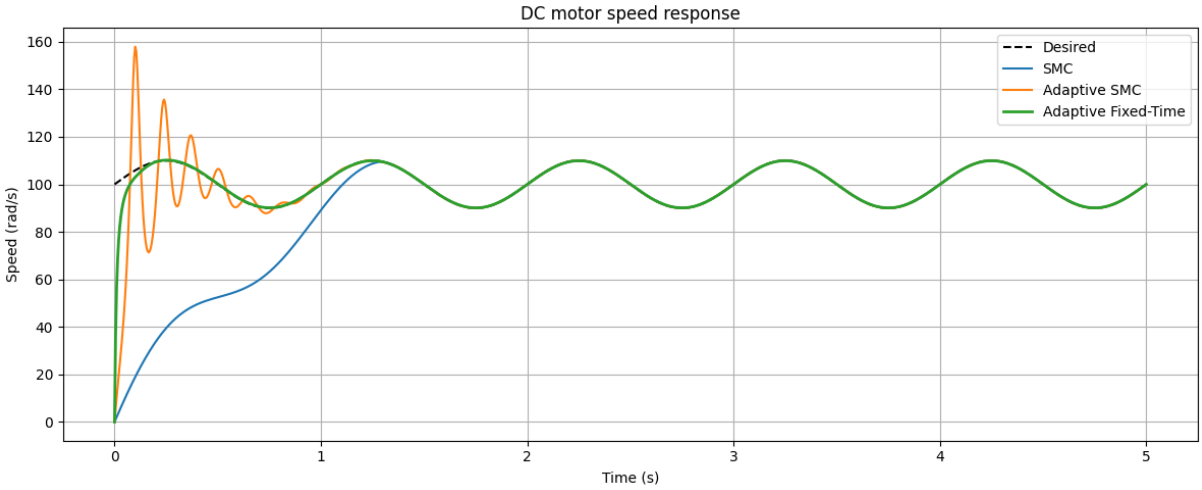
4.3. Kết quả mô phỏng bộ điều khiển động cơ sử dụng mạng RBF thích nghi

Với tham số bộ điều khiển được chọn tại Bảng 4-4, thời gian đáp ứng của bộ điều khiển được bao theo công thức (3.20):

$$T \le \frac{2}{k_1 2^{\frac{1+\alpha}{2}}(1-\alpha)} + \frac{2}{k_2 2^{\frac{1+\beta}{2}}(\beta-1)} = \frac{2}{10\times 2^{\frac{1+0.5}{2}}(1-0.5)} + \frac{2}{10\times 2^{\frac{1+1.5}{2}}(1.5-1)} \approx 0.4(s)$$

(4.1)

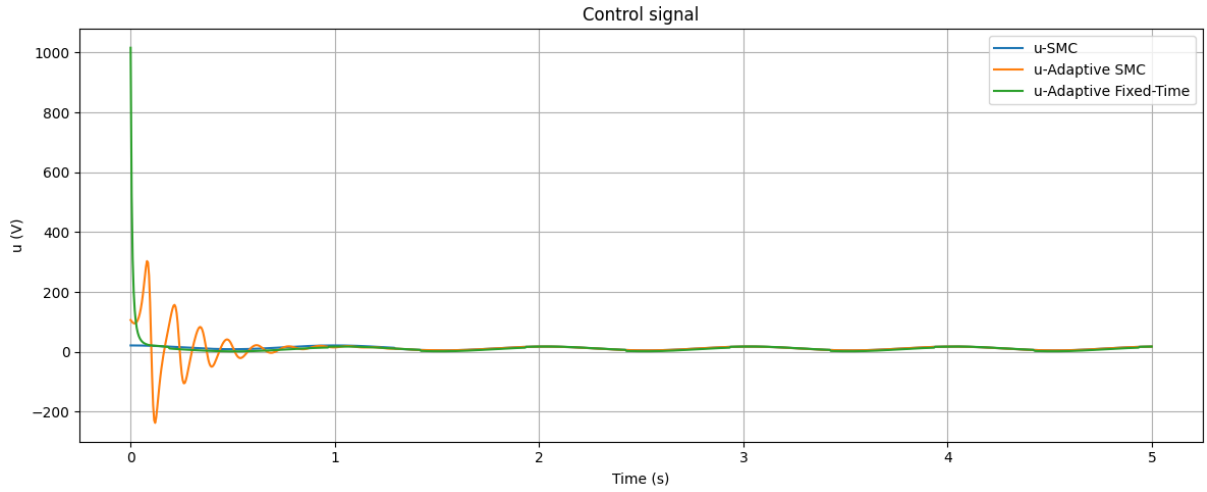
Kết quả đáp ứng vận tốc động cơ được biểu diễn tại Hình 4.6.



Hình 4.6. Mô phỏng đáp ứng động cơ của 3 phương pháp điều khiển trượt (SMC), bộ điều khiển trượt thích nghi (Adaptive SMC), bộ điều khiển thời gian định trước thích nghi (Adaptive Fixed-time)

Kết quả tại Bảng 4-6 thể hiện kết quả so sánh chất lượng điều khiển giữa ba phương pháp gồm: Sliding Mode Control (SMC), Adaptive Sliding Mode Control (Adaptive SMC) và Adaptive Fixed-time thông qua chỉ số sai số bình phương trung bình (RMSE). Theo kết quả thu được, phương pháp SMC truyền thống có

RMSE cao nhất, cho thấy mức sai số điều khiển lớn và khả năng thích ứng còn hạn chế trong môi trường nhiễu hoặc có đặc tính phi tuyến mạnh.



Hình 4.7. Biểu đồ tín hiệu điều khiển của 3 phương pháp SMC, Adaptive SMC và Adaptive Fixed-Time trong điều kiện thường

Đánh giá chất lượng 3 bộ điều khiển bằng phương pháp phân tích dữ liệu sai số toàn phương trung bình (RMSE) được mô tả tại Bảng 4-6.

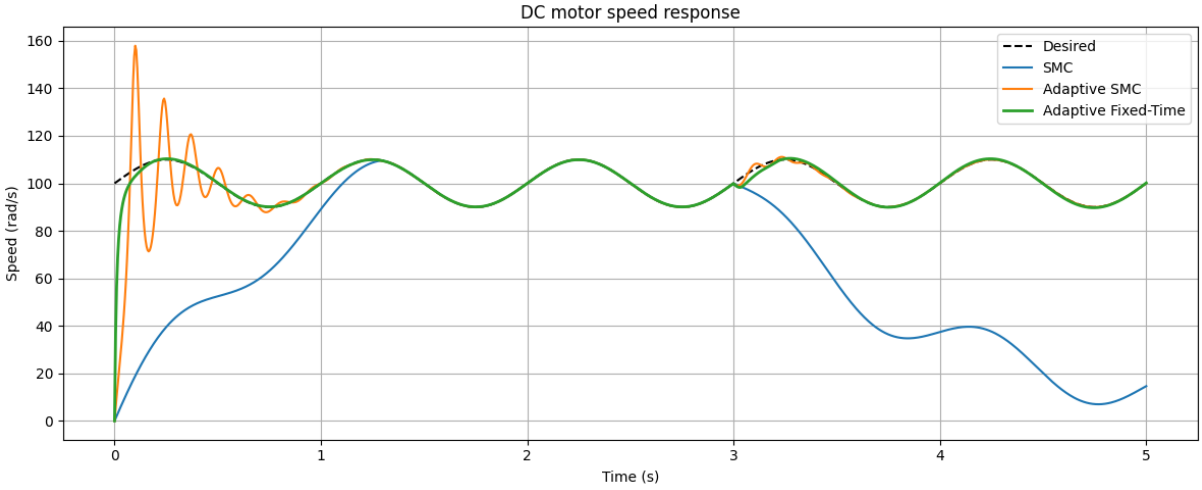
Bảng 4-6. Đánh giá chất lượng dựa trên tiêu chuẩn RMSE

Thông số	SMC	Adaptive SMC	Adaptive Fixed-time
RMSE	25.0879	10.2956	4.3563
Thời gian đáp ứng (s)	1.1	0.7	0.25

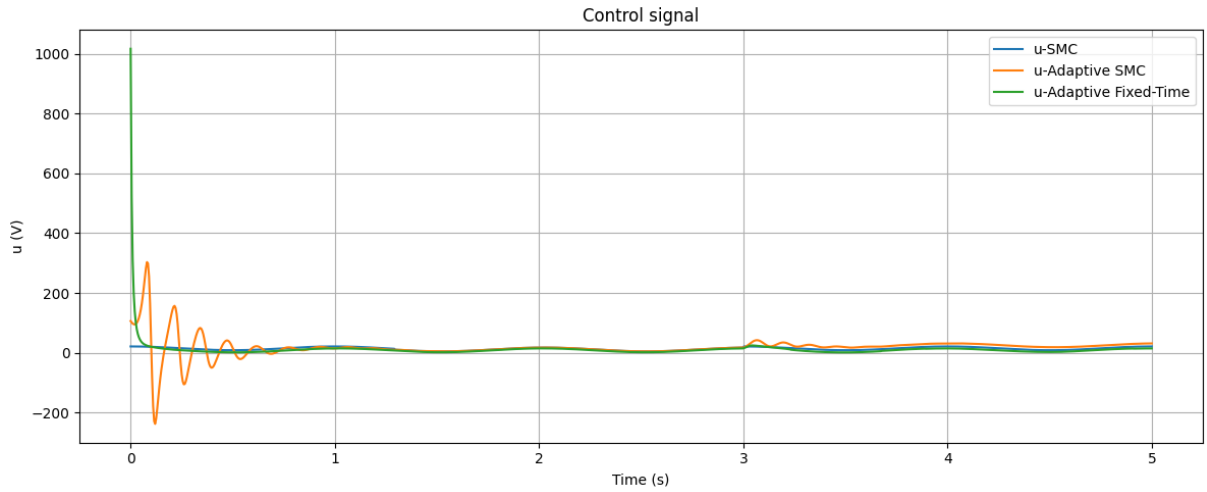
Khi áp dụng cơ chế thích nghi, phương pháp Adaptive SMC đã làm giảm đáng kể sai số điều khiển với RMSE chỉ còn 10.2956, tức giảm khoảng 58.9% so với SMC. Điều này cho thấy việc điều chỉnh tham số theo thời gian thực đã nâng cao hiệu quả ổn định hệ thống. Nổi bật nhất là phương pháp Adaptive Fixed-time với RMSE chỉ còn 4.3563, tiếp tục cải thiện hơn 57.7% so với Adaptive SMC và giảm tới 82.6% so với SMC. Ngoài ra, thời gian đáp ứng hệ thống của phương pháp Adative Fixed-time là 0.25(s) hoàn toàn được giới hạn trong thời gian định trước

0.4(s) tại phương trình (4.1). Kết quả này chứng minh rằng việc kết hợp giữa cơ chế thích nghi và đảm bảo hội tụ trong thời gian cố định không những tăng tính ổn định mà còn giúp hệ thống đạt được hiệu suất điều khiển tối ưu. Từ đó, có thể khẳng định rằng Adaptive Fixed-time Control là phương pháp hiệu quả và đáng tin cậy nhất trong ba phương án được so sánh.

Để thử nghiệm bộ điều khiển trong trường hợp chịu tải trong quá trình hoạt động, tại thời điểm $t = 3(s)$ một thay đổi về momen ngoại lực được thực hiện, giá trị momen ngoại lực thay đổi đột ngột từ $J = 0.1$ thành $J = 1.5$. Đáp ứng của hệ thống được thể hiện tại Hình 4.8.



Hình 4.8. Đáp ứng hệ thống với ngoại lực thay đổi đột ngột lên trục động cơ tại thời điểm $t = 3(s)$



Hình 4.9. Biểu đồ tín hiệu điều khiển của 3 phương pháp điều khiển SMC, Adaptive SMC và Adaptive Fixed-Time chịu nhiễu ngoại lực tác dụng lên trục động cơ tại $t = 3(s)$

Bảng 4-7. Đánh giá chất lượng RMSE với ngoại lực tác dụng

Thông số	SMC	Adaptive SMC	Adaptive Fixed-time
RMSE	46.8320	10.3511	4.3945

Tại Bảng 4-7 trình bày kết quả so sánh hiệu suất điều khiển của ba phương pháp SMC, Adaptive SMC và Adaptive Fixed-time khi hệ thống chịu tác động của ngoại lực nhiễu, thông qua chỉ số sai số bình phương trung bình (RMSE). Trong điều kiện có ngoại lực, bộ điều khiển SMC truyền thống ghi nhận RMSE cao nhất, đạt 46.8320, cho thấy khả năng chống nhiễu kém và mức sai số lớn trong việc duy trì ổn định hệ thống. Việc áp dụng cơ chế thích nghi trong Adaptive SMC đã giúp giảm đáng kể RMSE xuống còn 10.3511, tương ứng với mức giảm khoảng 77.9% so với SMC. Điều này khẳng định vai trò quan trọng của điều chỉnh tham số động trong việc nâng cao khả năng phản ứng với nhiễu. Đáng chú ý nhất, bộ điều khiển Adaptive Fixed-time tiếp tục thể hiện hiệu suất vượt trội với RMSE chỉ còn 4.3945 – thấp hơn 57.5% so với Adaptive SMC và thấp hơn 90.6% so với SMC. Sự kết hợp giữa tính thích nghi và đảm bảo hội tụ trong thời gian cố định giúp hệ thống

đạt được phản ứng nhanh, chính xác và ổn định ngay cả trong môi trường có tác động nhiễu mạnh. Tổng thể, Adaptive Fixed-time tiếp tục khẳng định ưu thế rõ rệt về độ chính xác và khả năng chống nhiễu so với hai phương pháp còn lại.

Qua đánh giá, phương pháp điều khiển thời gian hữu hạn thể hiện những ưu điểm vượt trội, luôn đảm bảo đáp ứng trong thời gian đặt trước không phụ thuộc vào thời điểm ban đầu, hoặc ảnh hưởng của nhiễu, cho thấy đây là phương pháp phù hợp có thể được sử dụng như bộ điều khiển cơ cấu chấp hành bậc thấp đảm bảo phản ứng nhanh, chính xác và bền vững cao khi kết hợp với mạng RBF.

Kết luận chương 4

Trong chương này, các công cụ mô phỏng đã được sử dụng để chứng minh hiệu quả của phương pháp điều khiển được đề xuất. Nhiều kịch bản được sử dụng để thử nghiệm khả năng của phương pháp điều khiển vị trí sử dụng học tăng cường, kết quả cho thấy khả năng đáp ứng mạnh trong các điều kiện môi trường phức tạp. Thử nghiệm nhiễu thay đổi cũng được thêm vào bộ điều khiển cơ cấu chấp hành, cho thấy khả năng đáp ứng với nhiễu mạnh đồng thời đảm bảo tốc độ phản hồi trong thời gian định trước.

KẾT LUẬN

Kết luận

Trong đề án này, một phương pháp điều khiển robot chịu nhiễu bất định được xây dựng dựa trên sự kết hợp phân tầng điều khiển gồm điều khiển vị trí sử dụng phương pháp học tăng cường và điều khiển cơ cấu chấp hành dựa trên phương pháp thích nghi mạng nơ ron RBF. So với các phương pháp điều khiển vị trí hiện tại, phương pháp điều khiển vị trí dựa trên học tăng cường được đề xuất không cần tín hiệu về bản đồ địa phương và lập quỹ đạo địa phương, điều này giảm khối lượng tính toán đồng thời vẫn đảm bảo chất lượng đáp ứng cần thiết về vị trí trong khi tránh vật cản từ môi trường. Trong đó bộ điều khiển cơ cấu chấp hành được thiết kế để đảm bảo đáp ứng trong thời gian hữu hạn và không biết trước về mô hình động lực học của robot. Lý thuyết toán học chứng minh khả năng ổn định của bộ điều khiển cơ cấu được đảm bảo đầy đủ qua lý thuyết ổn định Lyapunov đảm bảo hệ thống hội tụ trong thời gian hữu hạn trong mọi điều kiện trạng thái ban đầu và nhiễu ảnh hưởng.

Tính hiệu quả của phương pháp điều khiển vị trí và điều khiển cơ cấu chấp hành được chứng minh qua các mô phỏng với nhiều kịch bản khác nhau. Với bộ điều khiển vị trí, môi trường thử nghiệm được đặt trong kịch bản đơn giản và kịch bản vật cản di chuyển phức tạp ngẫu nhiên, kết quả đều cho thấy khả năng điều hướng vượt trội, tránh va chạm của phương pháp đề xuất. Với bộ điều khiển cơ cấu chấp hành, thử nghiệm điều khiển không có thông tin về mô hình động học và nhiễu trong quá trình hoạt động được thử nghiệm, các kết quả cho thấy thời gian đáp ứng luôn đảm bảo nhỏ hơn thời gian hữu hạn thiết kế. Thông số RMSE tại các trường hợp thử nghiệm cho thấy bộ điều khiển cơ cấu có khả năng bám sát tín hiệu đặt nhanh chóng và duy trì ổn định cao ngay khi chịu tác dụn của nhiễu.

Đồng thời, phương pháp chứng minh khả năng áp dụng cao của học tăng cường trong các bài toán điều khiển. Giải pháp được đưa ra hoàn toàn đáp ứng yêu cầu điều khiển về độ chính xác, khả năng tránh va chạm, khả năng đảm bảo

ổn định trong trường hợp chịu nhiễu hệ thống. Từ đó mở ra khả năng ứng dụng rộng rãi trong cuộc sống và sản xuất.

Hướng nghiên cứu và phát triển tiếp theo của luận văn

Trong tương lai, hệ thống được cải tiến để đảm bảo khả năng hoạt động trong điều kiện thời gian thực và đánh giá khả năng hoạt động thực tế của phương pháp điều khiển. Qua các quan sát, một số phương pháp điều khiển an toàn biên nên được tích hợp đảm bảo tránh va chạm hoàn toàn cho robot.

Ngoài ra phương pháp điều khiển sẽ được mở rộng để thử nghiệm trên các hệ thống đa robot, nhằm kiểm chứng khả năng của phương pháp học tăng cường trong các bài toán đa tác nhân ứng dụng trong nhiều bài toán kỹ thuật khó hiện nay.

DANH MỤC CÁC CÔNG TRÌNH KHOA HỌC ĐÃ CÔNG BỐ

Đã công bố:

1. Nguyen, Van-Truong, **Xuan-Thang Vu**, and Hai-Binh Giap. "Adaptive neural network hierarchical sliding-mode control for pendubot based genetic algorithm optimization." *Intelligent Systems and Networks: Selected Articles from ICISN 2022, Vietnam*. Singapore: Springer Nature Singapore, 2022. 574-580**.**

DANH MỤC TÀI LIỆU THAM KHẢO

[1] V. T. Nguyen, H.-A. Bui, A.-T. Nguyen, T.-L. . Bui, and D.-H. . Phan, "Development of navigation system for autonomous guided vehicle localization with measurement uncertainties," *Vietnam J. Sci. Technol.*, vol. 60, nº 3, pp. 513-526, 2022.

[2] Martins, F. N., Celeste, W. C., Carelli, R., Sarcinelli-Filho, M., & Bastos-Filho, T. F., "An adaptive dynamic controller for autonomous mobile robot trajectory tracking," *Control Engineering Practice,* vol. 16, nº 11, pp. 1354-1363, 2008.

[3] N. J. Nilsson, "Shakey the robot," *SRI International*, vol. 323, 1984.

[4] C. Samson, "Control of chained systems application to path following and time-varying point-stabilization of mobile robots," *IEEE Transactions on Automatic Control,* vol. 40, nº 1, pp. 64-77, 1995.

[5] L. S. P. L. J. &. O. M. Kavraki, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation,* vol. 12, nº 4, pp. 566-580, 1996.

[6] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal on Robotics and Automation,* vol. 2, nº 1, pp. 14-23, 1986.

[7] J. &. K. Y. Borenstein, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation,* vol. 7, nº 3, pp. 278-288, 1991.

[8] D. B. W. &. T. S. Fox, “The dynamic window approach to collision avoidance,” *IEEE Robotics & Automation Magazine,* vol. 4, nº 1, pp. 23-33, 1997.

[9] M. T. S. K. D. &. W. B. Montemerlo, “FastSLAM: A factored solution to the simultaneous localization and mapping problem,” em *AAAI Conference on Artificial Intelligence,* 2002.

[10] T. P. e. a. Lillicrap, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971,* 2015.

[11] S. F. C. D. T. &. A. P. Levine, “End-to-end training of deep visuomotor policies,” *Journal of Machine Learning Research,* vol. 17, nº 1, pp. 1334-1373, 2016.

[12] R. M. N. M. F. N. A. Ali, “A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods,” *IEEE Access,* vol. 9, pp. 134244-134270, 2021.

[13] S. Thrun, “Probabilistic robotics,” *Communications of the ACM,,* vol. 45, nº 3, pp. 52-57, 2002.

[14] A. K. A. M. I. a. A. J. H. M. S. Al Mahmud, “Advancements and challenges in mobile robot navigation: A comprehensive review of algorithms and potential for self-learning approaches,” *Journal of Intelligent & Robotic Systems,* vol. 110, nº 3, p. 120, 2024.

[15] O. a. B. A. a. T. E. a. S. M. a. B. D. a. C. S. a. G. C. a. M. A. a. D. M. a. S. A. a. S. P. a. C. F. a. B. R. Vermesan, “Internet of Robotic Things – Converging Sensing/Actuating, Hyperconnectivity, Artificial Intelligence

and IoT Platforms," *Internet of Things Intelligence Evolution,* pp. 97-155, 2017.

[16] A. L. a. N. S. Ahmad, "A systematic review on recent advances in autonomous mobile robot navigation," *Engineering Science and Technology, an International Journal,* vol. 40, p. 101343, 2023.

[17] A. Koubaa, Robot Operating System, Springer, 2017.

[18] e. a. V. Mnih, "Human-level control through deep reinforcement learning," *Nature,,* vol. 518, pp. 529-533, 2015.

[19] .. a. e. a. Y. W. H. Chen, "Multi-robot path planning based on a deep reinforcement learning DQN algorithm," *Neurocomputing,* vol. 452, pp. 464-476, 2021.

[20] S. H. a. G. Dissanayake, "Robot localization: An introduction," *Wiley Encyclopedia of Electrical and Electronics Engineering,* pp. 1-10, 1999.

[21] J. P. Q. T. N. G. Z. Z. a. T. W. Q. Li, "Multi-sensor fusion for navigation and mapping in autonomous vehicles: Accurate localization in urban environments," *Unmanned Systems,* vol. 8, nº 3, pp. 229-237, 2020.

[22] H. D.-W. a. T. Bailey, "Simultaneous localization and mapping: part I," *IEEE robotics & automation magazine,* vol. 13, nº 2, pp. 99-110, 2006.

[23] D. A. F. R. L. B. J. L. M. M. C. W. a. C. W. N. D. J. Bruemmer, "Shared understanding for collaborative control," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans,* vol. 35, nº 4, pp. 494-504, 2005.

[24] ResearchGate, "ResearchGate," Jul 2025. [Online]. Available: https://www.researchgate.net/figure/The-classification-of-robots-and-robotic-devices-2_fig4_235417999.

[25] "Evolution of Robots : 77-100 BC to 21st Century," [Online]. Available: http://blogs.infosys.com/digital-experience/emerging-technologies/evolution-of-robots.html.

[26] Z. L. Q. L. a. A. P. B. Wang, "Mobile robot path planning in dynamic environments through globally guided reinforcement learning," *IEEE Robotics and Automation Letters,* vol. 5, nº 4, pp. 6932-6939, 2020.

[27] J. R. a. X. L. R. Singh, "A review of deep reinforcement learning algorithms for mobile robot path planning," *Vehicles,* vol. 5, nº 4, pp. 1423-1451, 2023.

[28] J. K. Rosenblatt, "DAMN: A distributed architecture for mobile navigation," *Journal of Experimental & Theoretical Artificial Intelligence,* vol. 9, nº 2-3, pp. 339-360, 1997.

[29] R. a. G. D. a. B. M. Brachman, "Integrated AI Systems," *AI Magazine*, vol. 41, pp. 66-82, 2020.

[30] M. N. A. D. a. V. B. V. Ušinskis, "Sensor-fusion based navigation for autonomous mobile robo," *Sensors*, vol. 25, nº 4, p. 1248, 2025.

[31] Gurel, Canberk Suat and Rajendran Sathyam, Rajendra Mayavan and Guha, Akash, "ROS-based Path Planning for Turtlebot Robot using Rapidly Exploring Random Trees (RRT*)," 2018.

[32] T. Leung, “ROS Package cooperating OpenAI Gym and Baselines,” 24 1 2021. [Online]. Available: https://tiga.netlify.app/project/pyrobot-openai-gym-mujoco-gazebo/. [Acesso em 7 7 2025].

[33] K. Hulse, “linkedin,” 2025. [Online]. Available: https://www.linkedin.com/posts/kylehulse_robot-robots-robotics-activity-7288599309380468736-2p8B/. [Acesso em 7 7 2025].

[34] R. C. S. a. P. Cheeseman, “On the representation and estimation of spatial uncertainty,” *The international journal of Robotics Research,* vol. 5, nº 4, pp. 56-68, 1986.

[35] J. J. L. a. H. F. Durrant-Whyte, “Directed sonar sensing for mobile robot navigation,” *Springer Science & Business Media,* 2012.

[36] D. a. L. K. a. I. D.-Y. a. K. B. a. R. Y.-J. Tiep, “Design of Fuzzy-PID Controller for Path Tracking of Mobile Robot with Differential Drive,” *INTERNATIONAL JOURNAL of FUZZY LOGIC and INTELLIGENT SYSTEMS,* vol. 18, pp. 220-228, 2018.

[37] G. N. D. a. A. C. Kak, “Vision for mobile robot navigation: A survey,” *IEEE transactions on pattern analysis and machine intelligence,* vol. 24, nº 2, pp. 237-267, 2022.

[38] J. S. H. M. A. Ribeiro, “A Nonlinear Model Predictive Control Strategy for Trajectory Tracking of Omnidirectional Robots,” em *Lecture Notes in Electrical Engineering*, 2025.

[39] Y. S. S. Y. Z. G. a. T. W. H. Liu, “Deep reinforcement learning for mobile robot path planning,” *arXiv preprint arXiv:2404.06974,* 2024.

[40] J. R.-M. a. V. U.-C. M. Quinones-Ramirez, "Robot path planning using deep reinforcement learning," *arXiv preprint arXiv:2302.09120,* 2023.

[41] L. &. L. P. &. Q. S. &. Q. H. &. M. J. &. L. M. &. H. Y. &. M. E. Yang, "Path Planning Technique for Mobile Robots: A Review," *Machines,* 2023.

[42] G. T. a. V. T. M. Dorigo, "Swarm robotics: Past, present, and future [point of view]," *Proceedings of the IEEE,* vol. 109, nº 7, pp. 1152-1165, 2021.

[43] E. F. M. B. a. M. D. M. Brambilla, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence,* vol. 7, pp. 1-41, 2013.

[44] I. P. a. A. A. A. A. Chalvatzaras, "A survey on map-based localization techniques for autonomous vehicles," *IEEE Transactions on intelligent vehicles,* vol. 8, nº 2, pp. 1574-1596, 2022.

[45] H. M. E. S. a. H. Y. Y. Lu, "Real-time performance-focused localization techniques for autonomous vehicle: A review," *IEEE Transactions on Intelligent Transportation Systems,* vol. 23, nº 7, pp. 6082-6100, 2021.

[46] S. E. H. R. C. G. M. S. A. A. V. a. G. P. T. G. Reid, "Localization requirements for autonomous vehicles," *arXiv preprint arXiv:1906.01061,* 2019.

[47] Jose Luis Blanco, Mauro Bellone and Antonio Gimenez-Fernandez, "TP-Space RRT – Kinematic Path Planning of Non-Holonomic Any-Shape Vehicles," *Int J Adv Robot Syst,* pp. 12-55, 2015.

[48] R. S. S. R. a. M. M. F. Cappello, "Particle filter based multi-sensor data fusion techniques for RPAS navigation and guidance," *IEEE Metrology for Aerospace,* pp. 395-400, 2015.

[49] X. H. a. R. N. H. J. Ren, “Efficient deep reinforcement learning for optimal path planning,” *Electronics,* vol. 11, nº 21, p. 3628, 2022.

[50] J. F. Y. A. Y. a. H. N. J. Chung, “Learning team-based navigation: a review of deep reinforcement learning techniques for multi-agent pathfinding,” *Artificial Intelligence Review,* vol. 57, nº 2, p. 41, 2024.

[51] J. I. N.-V. C. S.-M. a. A. O.-d.-l.-P. D. P. Romero-Martí, “Navigation and path planning using reinforcement learning for a Roomba robot,” *2016 XVIII congreso Mexicano de robotica,* pp. 1-5, 2016.

[52] Yan, C., Chen, G., Li, Y., Sun, F., & Wu, Y., “Immune deep reinforcement learning-based path planning for mobile robot in unknown environment,” em *Applied Soft Computing*, 2023.

[53] A. P. N. U. &. V. A. Singh, “Speed Control of DC Motor using Pid Controller Based on Matlab,” em *Computer Engineering and Intelligent Systems*, 2013.

[54] J. P. G. &. P. D. Bharti, “Optimization of DC motor speed control using LQR technique,” em *Lecture notes in electrical engineering*, 2020.

[55] F. &. L. G. Wang, “Fixed-time control design for nonlinear uncertain systems via adaptive method,” *Systems & Control Letters*, 2020.

[56] V. V. X. G. H. Nguyen, “Adaptive Neural Network Hierarchical Sliding-Mode Control for Pendubot Based Genetic Algorithm Optimization,” em *Intelligent Systems and Networks. Lecture Notes in Networks and Systems*, Singapore, 2022.

[57] D. K. C. &. H. C. Kim, “Geometric path tracking and obstacle avoidance methods for an autonomous navigation of nonholonomic mobile robot,” *Journal of Institute of Control Robotics and Systems,* vol. 16, nº 8, pp. 771-779, 2010.

[58] precisionminidrives, “Small DC Brushed Motor,” [Online]. Available: https://precisionminidrives.com/product/31mm-small-dc-brushed-motor-57mm-type-model-nfp-r-31zy?srsltid=AfmBOoo79Fy9f4oOIG1YozoTkZYPJctMhDwB6Urows1w8eAgbni-_MmM. [Acesso em 05 04 2025].

[59] Matlab, “Mathworks,” Matlab, [Online]. Available: https://www.mathworks.com/help/control/ug/dc-motor-control.html. [Acesso em 7 7 2025].

[60] S. Salehi, “An efficient intrusive deep reinforcement learning framework for OpenFOAM,” *Meccanica,* 2024.

[61] E. Diederichs, “Reinforcement Learning - A Technical Introduction,” *Journal of Autonomous Intelligence,* vol. 2, p. 25, 2019.

[62] Statio, “Deep Q-Networks (DQNs) là gì?,” [Online]. Available: https://statio.vn/blog/deep-q-networks-dqns-la-gi-gioi-thieu-ve-mang-noron-sau-q-cach-hoat-ong-va-ung-dung-trong-reinforcement-learning.

[63] R. a. D. J. a. L. Y. a. H. A. A. a. C. H. Dong, “An enhanced deep deterministic policy gradient algorithm for intelligent control of robotic arms,” *Frontiers in Neuroinformatics,* vol. 17, p. 1096053, 2023.

[64] S. V. H. H. &. M. D. Fujimoto, “Addressing function approximation error in Actor-Critic methods,” *arXiv (Cornell University),* vol. 80, pp. 1587-1596, 2018.

[65] MINTROBOT, “Deep learning bible,” [Online]. Available: https://wikidocs.net/169332. [Acesso em 5 5 2025].

[66] A. Polyakov, “Nonlinear feedback design for Fixed-Time stabilization of linear control systems,” *IEEE Transactions on Automatic Control,* vol. 57, nº 8, pp. 2106-2110, 2011.

[67] ROS, “Setup and Configuration of the Navigation Stack on a Robot,” [Online]. Available: http://wiki.ros.org/navigation/Tutorials/RobotSetup. [Acesso em 7 9 2025].

[68] R. A. D. N. S. M. N. M. M. R. a. I. H. S. Nahavandi, “A comprehensive review on autonomous navigation,” *ACM Computing Surveys,* vol. 57, nº 9, pp. 1-67, 2025.

[69] Wikipedia, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Robot_navigation.

[70] C. Stachniss, “Robotic mapping and exploration,” *Springer Science & Business Media,* 2009.

[71] e. a. S. Thrun, “Stanley: The robot that won the DARPA Grand Challenge,” *Journal of field Robotics,* vol. 23, nº 9, pp. 661-692, 2006.

PHỤ LỤC

Code chương trình TD3

```
from pathlib import Path
import numpy as np
import torch
import torch.nn as nn
import torch.nn.functional as F
from numpy import inf
from torch.utils.tensorboard import SummaryWriter
from utils import get_max_bound


class Actor(nn.Module):
    def __init__(self, state_dim, action_dim):
        super(Actor, self).__init__()

        self.layer_1 = nn.Linear(state_dim, 400)
        torch.nn.init.kaiming_uniform_(self.layer_1.weight,
nonlinearity="leaky_relu")
        self.layer_2 = nn.Linear(400, 300)
        torch.nn.init.kaiming_uniform_(self.layer_2.weight,
nonlinearity="leaky_relu")
        self.layer_3 = nn.Linear(300, action_dim)
        self.tanh = nn.Tanh()

    def forward(self, s):
        s = F.leaky_relu(self.layer_1(s))
        s = F.leaky_relu(self.layer_2(s))
        a = self.tanh(self.layer_3(s))
        return a


class Critic(nn.Module):
    def __init__(self, state_dim, action_dim):
        super(Critic, self).__init__()

        self.layer_1 = nn.Linear(state_dim, 400)
        torch.nn.init.kaiming_uniform_(self.layer_1.weight,
nonlinearity="leaky_relu")
        self.layer_2_s = nn.Linear(400, 300)
        torch.nn.init.kaiming_uniform_(self.layer_2_s.weight,
nonlinearity="leaky_relu")
        self.layer_2_a = nn.Linear(action_dim, 300)
        torch.nn.init.kaiming_uniform_(self.layer_2_a.weight,
nonlinearity="leaky_relu")
        self.layer_3 = nn.Linear(300, 1)
```

```
        torch.nn.init.kaiming_uniform_(self.layer_3.weight,
nonlinearity="leaky_relu")

        self.layer_4 = nn.Linear(state_dim, 400)
        torch.nn.init.kaiming_uniform_(self.layer_1.weight,
nonlinearity="leaky_relu")
        self.layer_5_s = nn.Linear(400, 300)
        torch.nn.init.kaiming_uniform_(self.layer_5_s.weight,
nonlinearity="leaky_relu")
        self.layer_5_a = nn.Linear(action_dim, 300)
        torch.nn.init.kaiming_uniform_(self.layer_5_a.weight,
nonlinearity="leaky_relu")
        self.layer_6 = nn.Linear(300, 1)
        torch.nn.init.kaiming_uniform_(self.layer_6.weight,
nonlinearity="leaky_relu")

    def forward(self, s, a):
        s1 = F.leaky_relu(self.layer_1(s))
        self.layer_2_s(s1)
        self.layer_2_a(a)
        s11 = torch.mm(s1, self.layer_2_s.weight.data.t())
        s12 = torch.mm(a, self.layer_2_a.weight.data.t())
        s1 = F.leaky_relu(s11 + s12 + self.layer_2_a.bias.data)
        q1 = self.layer_3(s1)

        s2 = F.leaky_relu(self.layer_4(s))
        self.layer_5_s(s2)
        self.layer_5_a(a)
        s21 = torch.mm(s2, self.layer_5_s.weight.data.t())
        s22 = torch.mm(a, self.layer_5_a.weight.data.t())
        s2 = F.leaky_relu(s21 + s22 + self.layer_5_a.bias.data)
        q2 = self.layer_6(s2)
        return q1, q2


# TD3 network
class TD3(object):
    def __init__(
        self,
        state_dim,
        action_dim,
        max_action,
        device,
        lr=1e-4,
        save_every=0,
        load_model=False,
        save_directory=Path("cao_hoc/de_an/models/TD3/checkpoint"),
        model_name="TD3",
        load_directory=Path("robot_nav/models/TD3/checkpoint"),
```

```
        use_max_bound=False,
        bound_weight=0.25,
    ):
        self.device = device
        # Initialize the Actor network
        self.actor = Actor(state_dim, action_dim).to(self.device)
        self.actor_target = Actor(state_dim, action_dim).to(self.device)
        self.actor_target.load_state_dict(self.actor.state_dict())
        self.actor_optimizer =
torch.optim.Adam(params=self.actor.parameters(), lr=lr)

        # Initialize the Critic networks
        self.critic = Critic(state_dim, action_dim).to(self.device)
        self.critic_target = Critic(state_dim, action_dim).to(self.device)
        self.critic_target.load_state_dict(self.critic.state_dict())
        self.critic_optimizer =
torch.optim.Adam(params=self.critic.parameters(), lr=lr)

        self.action_dim = action_dim
        self.max_action = max_action
        self.state_dim = state_dim
        self.writer = SummaryWriter(comment=model_name)
        self.iter_count = 0
        if load_model:
            self.load(filename=model_name, directory=load_directory)
        self.save_every = save_every
        self.model_name = model_name
        self.save_directory = save_directory
        self.use_max_bound = use_max_bound
        self.bound_weight = bound_weight

    def get_action(self, obs, add_noise):
        if add_noise:
            return (
                self.act(obs) + np.random.normal(0, 0.2, size=self.action_dim)
            ).clip(-self.max_action, self.max_action)
        else:
            return self.act(obs)

    def act(self, state):
        state = torch.Tensor(state).to(self.device)
        return self.actor(state).cpu().data.numpy().flatten()

    # training cycle
    def train(
        self,
        replay_buffer,
        iterations,
        batch_size,
```

```
        discount=0.99,
        tau=0.005,
        policy_noise=0.2,
        noise_clip=0.5,
        policy_freq=2,
        max_lin_vel=0.5,
        max_ang_vel=1,
        goal_reward=100,
        distance_norm=10,
        time_step=0.3,
    ):
        av_Q = 0
        max_Q = -inf
        av_loss = 0
        for it in range(iterations):
            # sample a batch from the replay buffer
            (
                batch_states,
                batch_actions,
                batch_rewards,
                batch_dones,
                batch_next_states,
            ) = replay_buffer.sample_batch(batch_size)
            state = torch.Tensor(batch_states).to(self.device)
            next_state = torch.Tensor(batch_next_states).to(self.device)
            action = torch.Tensor(batch_actions).to(self.device)
            reward = torch.Tensor(batch_rewards).to(self.device)
            done = torch.Tensor(batch_dones).to(self.device)

            # Obtain the estimated action from the next state by using the
actor-target
            next_action = self.actor_target(next_state)

            # Add noise to the action
            noise = (
                torch.Tensor(batch_actions)
                .data.normal_(0, policy_noise)
                .to(self.device)
            )
            noise = noise.clamp(-noise_clip, noise_clip)
            next_action = (next_action + noise).clamp(-self.max_action,
self.max_action)

            # Calculate the Q values from the critic-target network for the
next state-action pair
            target_Q1, target_Q2 = self.critic_target(next_state, next_action)

            # Select the minimal Q value from the 2 calculated values
            target_Q = torch.min(target_Q1, target_Q2)
```

```
            av_Q += torch.mean(target_Q)
            max_Q = max(max_Q, torch.max(target_Q))
            # Calculate the final Q value from the target network parameters
by using Bellman equation
            target_Q = reward + ((1 - done) * discount * target_Q).detach()

            # Get the Q values of the basis networks with the current
parameters
            current_Q1, current_Q2 = self.critic(state, action)

            # Calculate the loss between the current Q value and the target Q
value
            loss = F.mse_loss(current_Q1, target_Q) + F.mse_loss(current_Q2,
target_Q)

            if self.use_max_bound:
                max_bound = get_max_bound(
                    next_state,
                    discount,
                    max_ang_vel,
                    max_lin_vel,
                    time_step,
                    distance_norm,
                    goal_reward,
                    reward,
                    done,
                    self.device,
                )
                max_excess_Q1 = F.relu(current_Q1 - max_bound)
                max_excess_Q2 = F.relu(current_Q2 - max_bound)
                max_bound_loss = (max_excess_Q1**2).mean() +
(max_excess_Q2**2).mean()
                # Add loss for Q values exceeding maximum possible upper bound
                loss += self.bound_weight * max_bound_loss

            # Perform the gradient descent
            self.critic_optimizer.zero_grad()
            loss.backward()
            self.critic_optimizer.step()

            if it % policy_freq == 0:
                # Maximize the actor output value by performing gradient
descent on negative Q values
                # (essentially perform gradient ascent)
                actor_grad, _ = self.critic(state, self.actor(state))
                actor_grad = -actor_grad.mean()
                self.actor_optimizer.zero_grad()
                actor_grad.backward()
                self.actor_optimizer.step()
```

```
                # Use soft update to update the actor-target network
parameters by
                # infusing small amount of current parameters
                for param, target_param in zip(
                    self.actor.parameters(), self.actor_target.parameters()
                ):
                    target_param.data.copy_(
                        tau * param.data + (1 - tau) * target_param.data
                    )
                # Use soft update to update the critic-target network
parameters by infusing
                # small amount of current parameters
                for param, target_param in zip(
                    self.critic.parameters(), self.critic_target.parameters()
                ):
                    target_param.data.copy_(
                        tau * param.data + (1 - tau) * target_param.data
                    )

            av_loss += loss
        self.iter_count += 1
        # Write new values for tensorboard
        self.writer.add_scalar("train/loss", av_loss / iterations,
self.iter_count)
        self.writer.add_scalar("train/avg_Q", av_Q / iterations,
self.iter_count)
        self.writer.add_scalar("train/max_Q", max_Q, self.iter_count)
        if self.save_every > 0 and self.iter_count % self.save_every == 0:
            self.save(filename=self.model_name, directory=self.save_directory)

    def save(self, filename, directory):
        Path(directory).mkdir(parents=True, exist_ok=True)
        torch.save(self.actor.state_dict(), "%s/%s_actor.pth" % (directory,
filename))
        torch.save(
            self.actor_target.state_dict(),
            "%s/%s_actor_target.pth" % (directory, filename),
        )
        torch.save(self.critic.state_dict(), "%s/%s_critic.pth" % (directory,
filename))
        torch.save(
            self.critic_target.state_dict(),
            "%s/%s_critic_target.pth" % (directory, filename),
        )

    def load(self, filename, directory):
        """
        Load the actor and critic networks (and their targets) from disk.
```

```
        self.actor.load_state_dict(
            torch.load("%s/%s_actor.pth" % (directory, filename))
        )
        self.actor_target.load_state_dict(
            torch.load("%s/%s_actor_target.pth" % (directory, filename))
        )
        self.critic.load_state_dict(
            torch.load("%s/%s_critic.pth" % (directory, filename))
        )
        self.critic_target.load_state_dict(
            torch.load("%s/%s_critic_target.pth" % (directory, filename))
        )
        print(f"Loaded weights from: {directory}")

    def prepare_state(self, latest_scan, distance, cos, sin, collision, goal,
action):
        latest_scan = np.array(latest_scan)

        inf_mask = np.isinf(latest_scan)
        latest_scan[inf_mask] = 7.0

        max_bins = self.state_dim - 5
        bin_size = int(np.ceil(len(latest_scan) / max_bins))

        # Initialize the list to store the minimum values of each bin
        min_values = []

        # Loop through the data and create bins
        for i in range(0, len(latest_scan), bin_size):
            # Get the current bin
            bin = latest_scan[i : i + min(bin_size, len(latest_scan) - i)]
            # Find the minimum value in the current bin and append it to the
min_values list
            min_values.append(min(bin) / 7)

        # Normalize to [0, 1] range
        distance /= 10
        lin_vel = action[0] * 2
        ang_vel = (action[1] + 1) / 2
        state = min_values + [distance, cos, sin] + [lin_vel, ang_vel]

        assert len(state) == self.state_dim
        terminal = 1 if collision or goal else 0

        return state, terminal
```