NGHIÊN CỨU TỐI ƯU THUẬT TOÁN ĐIỀU KHIỂN ĐA ROBOT AGV TRÊN PHẦN MỀM ĐIỀU PHỐI TẬP TRUNG – ROBOT CONTROL SYSTEM

OPTIMIZATION RESEARCH ON MULTI-AGV CONTROL ALGORITHMS ON A CENTRALIZED – ROBOT CONTROL SYSTEM

Nguyễn Ngọc Kiên[1,*], Giang Quốc Hoàn[2], Nguyễn Văn Tình[1]
[1]Khoa Cơ khí chế tạo máy, Trường Cơ khí, Đại học Bách Khoa Hà Nội
[2]Tổng Công ty Cổ phần Bưu chính Viettel
**Email liên hệ: kien.nguyenngoc@hust.edu.vn*

TÓM TẮT

Bài báo nghiên cứu và xây dựng giải pháp tối ưu điều khiển đa robot AGV trên nền tảng điều phối tập trung Robot Control System (RCS). AGV là phương tiện tự hành dùng cho vận chuyển hàng hóa trong kho xưởng, đòi hỏi thuật toán điều hướng chính xác, phân công nhiệm vụ hợp lý và đảm bảo an toàn khi vận hành đồng thời nhiều robot. Nghiên cứu xây dựng ba thuật toán chính: Tìm đường đi ngắn và ít Robot nhất bằng cải tiến thuật toán A; tính toán phân bổ nhiệm vụ thích ứng theo trạng thái robot; và xây dựng thuật toán tránh va chạm dựa trên dự báo xung đột theo thời gian thực. Các thuật toán được lập trình, mô phỏng trên Python và tích hợp cơ chế truyền thông đa robot để đánh giá tính hiệu quả và khả năng triển khai thực tế.*

Từ khóa: *AGV; Điều khiển đa robot; RCS; A* nâng cấp; Phân bổ nhiệm vụ; Tránh va chạm; Mô phỏng Python.*

ABSTRACT

This paper presents an optimization study of multi-AGV control algorithms on a centralized Robot Control System (RCS). AGVs (Automated Guided Vehicles) are autonomous mobile units used for material handling in warehouses and factories, requiring efficient path planning, task assignment, and collision-free coordination. The research focuses on three main algorithmic approaches: Find the shortest path with the fewest Robots with the upgraded A, a dynamic task allocation method responsive to real-time robot status, and a time-prediction-based collision avoidance algorithm. All methods were simulated and evaluated using Python, with a multi-robot communication mechanism integrated to assess performance, scalability, and practical deployment potential.*

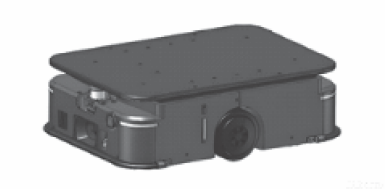
Keywords: *AGV; Multi-robot control; RCS; Enhanced A*; Task allocation; Collision avoidance; Python simulation.*

☞

1. NGHIÊN CỨU, PHÂN TÍCH HỆ THỐNG ĐIỀU KHIỂN XE TỰ HÀNH AGV

1.1. Robot tự hành AGV

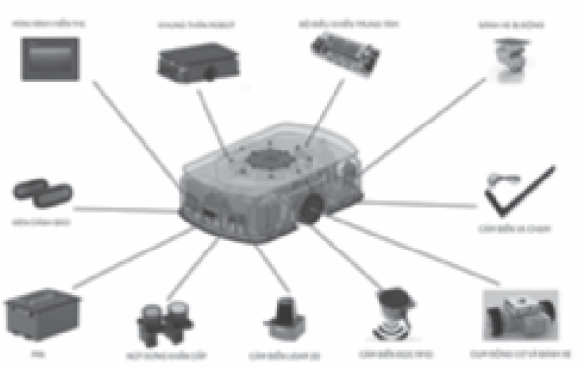
Xe tự hành AGV – Automation Guided Vehicle, là loại xe sử dụng công nghệ dẫn đường để tự động di chuyển nhằm mục đích vận chuyển hàng hóa, nguyên vật liệu đến những địa điểm đã được đánh dấu sẵn mà không cần sự can thiệp của con người. Xe tự hành AGV còn được gọi với nhiều tên khác như Robot AGV, Robot vận chuyển hàng tự động.



Hình 1.1. Robot AGV nâng hàng

Xe tự hành AGV được phân chia thành các loại chính là: Xe tự hành dạng đẩy, dạng nâng, dạng kéo, dạng chở, trên này là thiết kế AGV dạng nâng hàng.

1.2. Cấu tạo của xe tự hành AGV

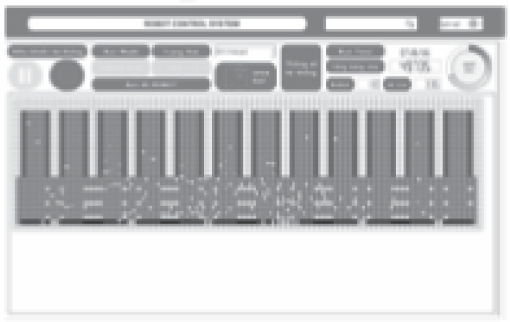


Hình 1.2. Cấu tạo chi tiết Robot tự hành AGV

Cấu tạo Robot AGV gồm các thành phần chính: Khung xe cơ khí chịu tải, phần mềm điều phối RCS, bộ điều khiển trung tâm, kết nối không dây qua RF/WiFi, giao diện người dùng quản lý – giám sát, pin và bộ sạc, thiết bị truyền nhận dữ liệu và cơ cấu gắn/kéo hàng.

2. HỆ THỐNG ĐIỀU KHIỂN ĐA ROBOT – ROBOT CONTROL SYSTEM

Bài báo tập trung nghiên cứu, tối ưu thuật toán trên phần mềm điều khiển RCS (Robot Control System) để quản lý, điều phối, và tối ưu hóa hoạt động của các Robot trong môi trường làm việc đa Robot. RCS có khả năng tương tác với các Robot, cảm biến và các thiết bị khác để thực hiện các nhiệm vụ một cách hiệu quả và linh hoạt.



Hình 2.1. Phần mềm điều phối Robot tập trung Robot Control System

2.1. Các thành phần của phần mềm RCS

Cấu trúc phần mềm điều phối đa robot (RCS) gồm các thành phần chính: Giao diện người dùng để giám sát và tương tác hệ thống; module điều khiển thực thi thuật toán phân công và quản lý chuyển động; module dẫn đường xử lý tìm đường và tránh chướng ngại vật; module giao tiếp truyền dữ liệu giữa RCS và robot qua các giao thức như TCP/IP/UDP, RF; cùng cơ sở dữ liệu lưu trữ trạng thái robot, nhiệm vụ và thông tin môi trường phục vụ tối ưu hóa vận hành.

2.2. Nghiên cứu, xây dựng thuật toán dẫn đường – RCS

* Thiết lập thuật toán A* để tính toán đường đi ít số lượng Robot nhất

A* là thuật toán tìm đường phổ biến, sử dụng hàm đánh giá để xác định đường đi tối ưu từ điểm khởi đầu đến điểm đích. A* hiệu quả trong môi trường tĩnh và cho phép Robot tìm được lộ trình nhanh nhất. Nghiên cứu đặt ra bài toán A* nâng cấp tìm đường đi ngắn có ít Robot nhất là giải thuật tìm kiếm trong đồ thị, tìm đường đi từ một đỉnh hiện tại đến đỉnh đích có sử dụng hàm để ước lượng khoảng cách hay còn gọi là hàm Heuristic kết hợp thêm việc xác định vị trí các Robot cùng trên bản đồ để tối ưu đường đi ngắn và ít Robot nhất.

* Thuật giải A*

Công thức mở rộng A*:

$$\text{Total Cost}(F) = G_{cost} + H_{cost} + W \times L_{cost} \quad (2\text{-}1)$$

Công thức này là một biến thể của hàm đánh giá A* cơ bản, được sử dụng trong các hệ thống Robot AGV, Xe tự lái và các bài toán lập kế hoạch di chuyển phức tạp, nơi không chỉ quan tâm đến quãng đường mà còn đến các yếu tố chi phí khác.

Bảng 2.1. Bảng thành phần A mở rộng*

Thành phần	Ý nghĩa trong Robot/Hệ thống Phức tạp
G_{cost}	Chi phí thực tế của quãng đường đã đi (Actual Path Cost)
H_{cost}	Chi phí ước lượng đến đích (Heuristic Cost): Ước tính chi phí (khoảng cách đường chéo và khoảng cách Manhattan) từ vị trí hiện tại đến đích.
L_{cost}	Chi phí Tránh Xung đột/Tránh Rủi ro (Collision/Penalty Cost): Đây là một chi phí phạt (Penalty) để đánh giá độ an toàn, tỉ lệ Robot trên đường.
W	Là hằng để điều chỉnh mức độ quan trọng của L_{cost} so với G_{cost} và H_{cost}. Nếu W lớn, hệ thống sẽ ưu tiên các đường đi an toàn (L_{cost} thấp) hơn là đường đi ngắn nhất (G_{cost}/H_{cost} thấp).

* Heuristic xấp xỉ

Khoảng cách Manhattan: Khoảng cách Manhattan là tổng các giá trị tuyệt đối của sự khác biệt giữa tọa độ x và y của ô hiện tại và ô đích. Sử dụng cho việc Robot sẽ đi ngược từ vị trí đổ trả hàng quay về đầu cấp lấy hàng.

- Giả sử: Vị trí hiện tại: (x_{curr}, y_{curr}), vị trí mục tiêu: (x_{goal}, y_{goal}).

- Heuristic được tính bằng:

$$h = |x_{curr} - x_{goal}| + |y_{curr} - y_{goal}| \quad (2\text{-}2)$$

- Sử dụng phương pháp tìm kiếm này khi Robot đi ngược từ vị trí đích về vị trí lấy hàng duy trì đường đi ngắn và ít nhánh rẽ.

Khoảng cách đường chéo: Giá trị tuyệt đối lớn nhất của sự khác biệt giữa tọa độ x và y của ô hiện tại và ô đích.

Công thức:

$$d_y = |y_{curr} - y_{goal}| \quad (2\text{-}3)$$

Heuristic đường chéo:

$$h = D*(dx + dy) + (D_2 - 2D)*\min(dx,dy) \quad (2\text{-}4)$$

- Thích hợp cho Robot di chuyển từ vị trí cấp hàng đến vị trí đích nếu khu vực cho phép di chuyển linh hoạt hơn.

- Dễ kết hợp với chi phí bổ sung: Robot ☞

cắt ngang đường ưu tiên, khu vực tắc nghẽn, chi phí do Robot nằm trên đường.

* Chi phí Robot nằm trên đường

Xác định một đường thẳng hình chữ L (trước theo trục X rồi trục Y, hoặc ngược lại) từ vị trí hiện tại đến mục tiêu. Đếm số Robot xuất hiện trên các ô thuộc đường này.

Thêm chi phí phạt vào hàm heuristic để thuật toán A* ưu tiên đường ít xung đột hơn.

- Cách tính số Robot nằm trên đường:

• Giả sử một đường đi gồm hai đoạn: Đoạn theo trục X + Đoạn theo trục Y.
• Vị trí hiện tại Robot: (x_{curr},y_{curr}).
• Vị trí mục tiêu: (x_{goal},y_{goal}).
• Tập vị trí của tất cả Robot khác: $R = \{(x_i,y_i)\}$.

- Bước 1 - Tập điểm trên đường X:

$$Path_X = \{(x,y_{curr}) \mid x \in [\min(x_{curr},x_{goal}),\max(x_{curr},x_{goal})]\} \quad (2\text{-}5)$$

- Bước 2 - Tập điểm trên đường Y:

$$Path_Y = \{(x_{goal},y) \mid y \in [\min(y_{curr},y_{goal}),\max(y_{curr},y_{goal})]\} \quad (2\text{-}6)$$

- Bước 3 - Đếm số Robot trên đường:

$$N_{robot} = \sum_{(x,y)\in Path_X \cup Path_Y} \mathbf{1}((x,y) \in R) \quad (2\text{-}7)$$

Trong đó: 1(x,y) là hàm chỉ báo (đúng = 1, sai = 0).

- Công thức thêm chi phí Robot vào hàm heuristic đường chéo:

$$dx = | x_{curr} - x_{goal} | \quad (2\text{-}8)$$

$$dy = | y_{curr} - y_{goal} | (2-7) \quad (2\text{-}9)$$

$$h_{diag} = D(dx + dy) + (D_2 - 2D)\min(dx, dy) \quad (2\text{-}10)$$

- Thêm chi phí Robot vào heuristic:

• Đặt C_r = hệ số phạt cho mỗi Robot chắn đường (1, 2 hoặc ≥ 5 tùy mức độ ưu tiên tránh xung đột).

$$h = h_{diag} + C_r \cdot N_{robot} \quad (2\text{-}11)$$

Trong đó:

$$h = D(dx + dy) + (D_2 - 2D)\min(dx, dy) + C_r \cdot N_{robot} \quad (2\text{-}12)$$

* Ý nghĩa của hàm chi phí

Nếu đường có nhiều Robot → chi phí tăng → A* tự động chọn đường vòng, giảm xung đột, tìm kiếm cung đường phù hợp cho Robot, đi trên đường vắng nhất.

2.3. Nghiên cứu thuật toán phân phối và điều khiển đa Robot

Hình 2.2. Thuật toán phân phối và điều khiển đa Robot

ISSN 2615 - 9910 (bản in), ISSN 2815 - 5505 (online)
TẠP CHÍ CƠ KHÍ VIỆT NAM, Số 337+338, tháng 1+2 năm 2026
cokhivietnam.vn

Thuật toán điều phối Robot trong RCS dựa trên trạng thái hoạt động và mức ưu tiên nhiệm vụ. Hệ thống backend liên tục nhận trạng thái robot và gán nhiệm vụ phù hợp với điều kiện thực tế.

- Robot ở bãi đỗ: Nếu số robot tại khu vực Induction chưa vượt giới hạn → điều động tới Induction; ngược lại tiếp tục chờ.

- Robot ở đầu cấp: Quét mã hàng → nếu mã hợp lệ, truy vấn WMS và điều hướng tới đúng Chute; nếu không hợp lệ thì tiếp tục chờ.

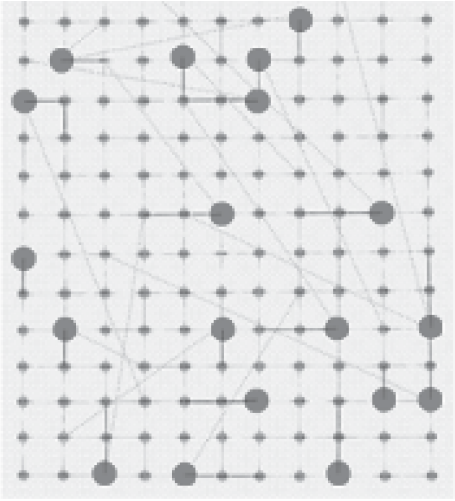
- Robot hoàn thành nhiệm vụ: Nếu pin yếu → chuyển tới trạm sạc; nếu pin đủ → quay lại Induction để nhận hàng mới.

- Robot đang sạc: Khi pin đạt mức tối thiểu → quay lại Parking chờ lệnh; nếu chưa đạt thì tiếp tục sạc.

Tóm tắt: Robot được điều hướng tới vị trí phù hợp theo trạng thái thực tế: Bãi đỗ → Đầu Cấp hàng → Vị trí đích → Trạm sạc, với quyết định phụ thuộc tải nhiệm vụ, và mức pin. Hệ thống luôn chọn điểm đến gần nhất để tối ưu thời gian và giảm xung đột.

2.4. Xây dựng thuật toán tránh va chạm sử dụng thuật toán tiên đoán thời gian

Thuật toán tránh va chạm cho hệ thống đa Robot trong môi trường dạng ma trận có thể được xây dựng dựa trên thuật toán tiên đoán về thời gian (Time-Predictive Collision Avoidance Algorithm). Thuật toán này sử dụng thông tin về vị trí, hướng di chuyển và tốc độ của mỗi Robot để dự đoán vị trí tương lai của chúng và tránh va chạm.



Hình 2.3. Thuật toán tránh va chạm

* Khởi tạo các biến và thông tin cần thiết

Bản đồ dạng ma trận (Grid Map): Bản đồ được chia thành các ô vuông, mỗi ô có tọa độ (x, y) và có thể ở trạng thái "trống" hoặc "bận".

Robot State:

- R_i đại diện cho Robot thứ i.
- $P_i(t) = (x_i(t),y_i(t))$: Vị trí của Robot i tại thời điểm t.
- V_i: Vận tốc của Robot i.
- $T_i(t)$: Hướng di chuyển của Robot i tại thời điểm t.

Thời gian dự đoán (Prediction Horizon) T: Khoảng thời gian mà hệ thống sẽ dự đoán vị trí của Robot để phát hiện và tránh va chạm.

Ma trận thời gian (Time Matrix): Lưu trữ thông tin về thời điểm mà mỗi ô sẽ bị chiếm bởi các Robot, M(x,y) = t có nghĩa là ô tại tọa độ (x,y)(x,y)(x,y) sẽ được chiếm vào thời điểm t.

Dự đoán vị trí của Robot trong thời gian tới: Tại mỗi bước thời gian t, dự đoán vị trí của mỗi Robot R_i trong khoảng thời gian dự đoán T: ☞

Sử dụng tốc độ V_i và hướng $T_i(t)$ của Robot để tính toán vị trí dự kiến trong các bước thời gian tiếp theo:

$$P_i(t+\Delta t) = P_i(t) + V_i \times T_i(t) \quad (2\text{-}13)$$

Cập nhật vị trí dự đoán của mỗi Robot trong ma trận thời gian, nếu R_i sẽ chiếm ô (x,y) vào thời điểm $t+\Delta t$, thì ta gán:

$$M(x,y) = t+\Delta t \quad (2\text{-}14)$$

* Kiểm tra va chạm trong thời gian dự đoán

Sau khi cập nhật vị trí dự kiến cho tất cả các Robot, kiểm tra từng Robot R_i để xác định xem vị trí tương lai của nó có trùng với vị trí của bất kỳ Robot nào khác trong ma trận thời gian.

Với mỗi bước thời gian t+k (trong khoảng T), kiểm tra xem vị trí $P_i(t+k)$ có trùng với vị trí dự kiến của các Robot khác tại cùng thời điểm hay không.

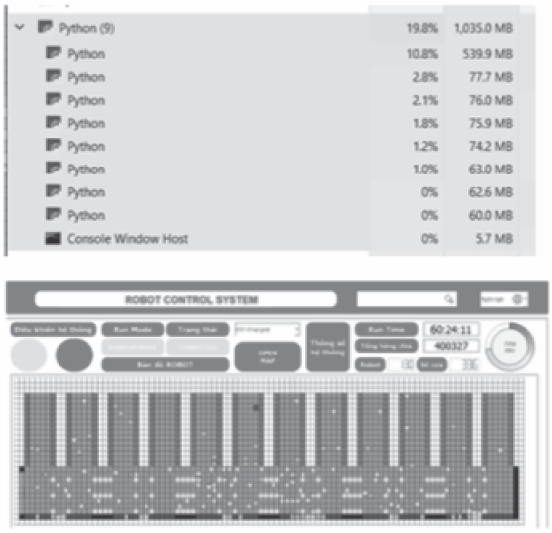
Nếu $M(x_i(t+k),y_i(t+k)) = t+k$, thì xảy ra xung đột vị trí, nghĩa là có khả năng va chạm.

Giải pháp tránh va chạm: Nếu phát hiện xung đột vị trí, có thể thực hiện các bước sau để tránh va chạm: Giảm tốc độ, Thay đổi hướng di chuyển, Chờ trong một khoảng thời gian, Điều chỉnh thứ tự ưu tiên.

Cập nhật ma trận thời gian: Sau khi đã giải quyết các xung đột, cập nhật lại ma trận thời gian. Thuật toán tiếp tục lặp lại các bước trên trong thời gian thực để đảm bảo không có va chạm nào xảy ra khi các Robot di chuyển trong môi trường.

3. THỬ NGHIỆM MÔ PHỎNG VÀ ĐÁNH GIÁ HỆ THỐNG

Đánh giá hiệu suất và khả năng hoạt động liên tục chạy thử nghiệm phần mềm:



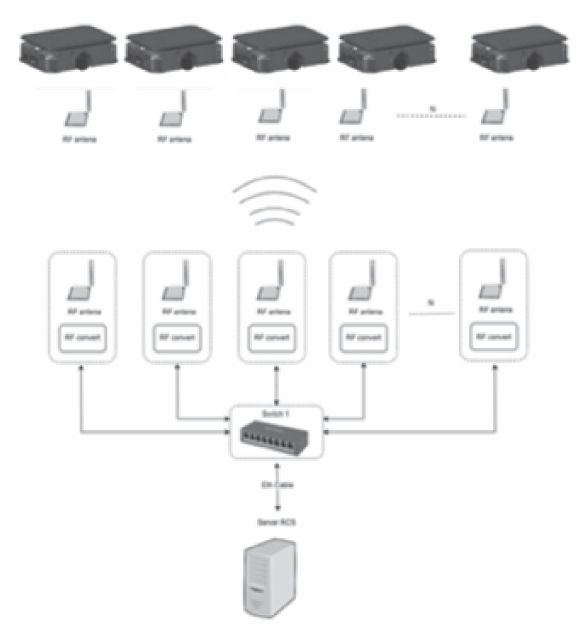
Hình 3.1. Kết quả thực hiện mô phỏng phần mềm RCS

Hệ thống hoạt động 60 tiếng liên tục, thực hiện hơn 400.000 nhiệm vụ, số lượng Robot được điều khiển trong hệ thống đạt 0 – 130 Robot tùy thuộc vào kích thước của bản đồ được khởi tạo. Các Robot đảm bảo việc di chuyển hàng hóa chính xác đến các địa chỉ đích, không bị va chạm trong quá trình di chuyển. Hệ thống không bị treo, các Robot vẫn phân bố đều năng suất được duy trì. Hiệu năng máy tính duy trì ở mức ổn định: Speed 4.0 Ghz, 39.5% CPU – I5 13400, Ram 1,03 GB, 9 Multi Core Process.

4. PHƯƠNG THỨC TRUYỀN THÔNG VỚI ĐA ROBOT AGV

Giao thức truyền thông giữa hệ thống điều khiển Robot (Robot Control System - RCS) và các Robot là một yếu tố quan trọng để đảm bảo các Robot hoạt động trơn tru và đồng bộ. Các giao thức này thường dựa trên các tiêu chuẩn truyền thông phổ biến trong

tự động hóa và điều khiển, bao gồm MQTT, WebSocket, TCP/IP, RESTful API, và RF (Radio Frequency).



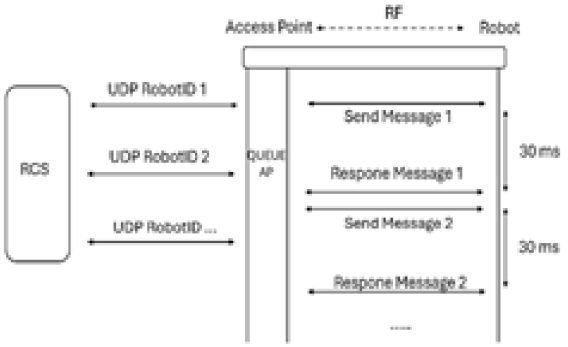
Hình 4.1. Sơ đồ kết nối RCS với AGV qua RF

* Về cơ chế thử nghiệm Lora (Long Range):

• LoRa: Là công nghệ RF có phạm vi phủ sóng xa, phù hợp cho các ứng dụng đòi hỏi kết nối ở khoảng cách lớn.

• Kiến trúc hoạt động: Robot và RCS kết nối với nhau qua gateway LoRa, giúp truyền thông từ xa và xuyên qua các môi trường có vật cản.

• Khung bản tin: Header - Chứa thông tin về bản tin, mã hóa và kiểm tra lỗi. Payload - Mã nhị phân, chứa các thông tin cơ bản về nhiệm vụ hoặc trạng thái của Robot.



Hình 4.2. Nguyên lý truyền bản tin RCS – AP – AGV

5. KẾT LUẬN

* Quy trình tích hợp:

- Đã đưa ra được mô hình toán học để mô phỏng hành vi của Robot trong môi trường nhà kho thông minh.

- Đã xây dựng và lập trình được các thuật toán điều khiển và dẫn đường được trong phần mềm RCS.

- Đã thực hiện các bài kiểm tra trong môi trường mô phỏng trước khi triển khai thực tế, nhằm điều chỉnh và tối ưu hóa thuật toán.

* Hiệu quả của 3 thuật toán đã được nghiên cứu áp dụng trong phần mềm điều phối – RCS:

- Tăng cường độ tin cậy: Các thuật toán điều khiển và dẫn đường giúp Robot hoạt động an toàn và hiệu quả, tiết kiệm chi phí năng lượng 5-10%, giảm được chi phí cảm biến Lidar trên Robot lên tới 20 Triệu đồng.

- Tối ưu hóa quy trình: Các thuật toán cho phép tối ưu hóa lộ trình và phân công nhiệm vụ, hiệu suất làm việc của cả hệ thống Robot AGV được nâng lên 20% đã được kiểm nghiệm tại nhà kho thông minh. ☞

- Linh hoạt và thích ứng: Hệ thống có khả năng phản hồi nhanh với những thay đổi trong môi trường, thay đổi vị trí các đích đến của hàng hóa trong kho hàng thông minh, duy trì hoạt động ổn định.❖

Ngày nhận bài: **02/12/2025**
Ngày phản biện: **13/12/2025**

Tài liệu tham khảo:

[1]. Yan Yang, Qiong Cai (2023), "*Research on the A-star Algorithm Based on Path Finding*". IEEE 3rd International Conference on Data Science and Computer Application (ICDSCA).

[2]. Valerio Digani, Lorenzo Sabattini, Cristian Secchi, Cesare Fantuzzi (2015), "*Ensemble Coordination Approach in Multi-AGV Systems Applied to Industrial Warehouses*". IEEE Transactions on Automation Science and Engineering (Volume: 12, Issue: 3, July 2015).

[3]. Yihao Liang, Mingrui Liu (2022), "*Multi-AGV Simulation System Based on Bipartite Graph, Space-Time A*, and Conflict Search*". IEEE China Automation Congress (CAC).

[4]. Bin-Bin Hu, Hai-Tao Zhang (2024), "*Multi-Robot Systems: Coordinated Fencing Control and Applications*". Springer Nature Singapore.

[5]. TS. Đỗ Trung Hải, ThS. Nguyễn Thị Tuyết Hoa, PGS.TS. Nguyễn Tuấn Minh (2023), "*Kỹ thuật điều khiển nhóm cho đàn Robot tự hành*". NXB. Khoa học và Kỹ thuật.