

**ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**



**RBE3017 - BÁO CÁO CUỐI KỲ
MÔN HỌC: LẬP TRÌNH ROBOT VỚI ROS**

SLAM CHO ROBOT DI ĐỘNG

Sinh viên:

Nguyễn Thị Ánh - 19020500

Trần Văn Chiến - 19020506

Nguyễn Thị Ngọc Mai - 19020577

Giảng viên hướng dẫn:

PGS.TS.Hoàng Văn Xiêm

ThS. Phan Hoàng Anh

HÀ NỘI - 2022

LỜI CẢM ƠN

Nhóm xin gửi lời cảm ơn đến PGS.TS.Hoàng Văn Xiêm, ThS. Phan Hoàng Anh và các thầy, cô công tác tại khoa Điện tử - Viễn thông, trường Đại học Công Nghệ - Đại học Quốc gia Hà Nội, đã tận tình hướng dẫn và giúp đỡ nhóm trong suốt quá trình tìm hiểu môn học để có thể hoàn thành bản báo cáo.

TÓM TẮT

Hệ điều hành Robot (Robot Operating System – ROS) đã trở thành một phần quan trọng trong lĩnh vực Robot hiện tại, có tiềm năng cực kỳ lớn giúp giải quyết các vấn đề của Robot. Một trong những lợi thế lớn nhất của ROS là nó cho phép tích hợp liền mạch các quá trình khác nhau hay còn gọi là các node trong hệ thống Robot, các node có thể được sử dụng trong các hệ thống khác mà không có sự chồng chéo nào.

Một trong những ứng dụng phổ biến nhất của ROS là SLAM (Simultaneous Localization and Mapping). Mục tiêu của SLAM trong Robot di động là xây dựng và cập nhật bản đồ của một môi trường chưa được khám phá với sự trợ giúp của các cảm biến có sẵn được gắn vào Robot.

Bài báo cáo này sẽ triển khai các package có sẵn trong ROS, bao gồm stack Navigation và Gmapping để lập bản đồ một ngôi nhà Robot chưa từng khám phá bằng cách sử dụng SLAM tự động dùng thuật toán cây ngẫu nhiên RRT (Rapidly Exploring Random Tree) để giúp Robot di chuyển che phủ tất cả các vùng môi trường chưa biết. Sau đó để cập đến một thuật toán định vị xác suất được gọi là Định vị Monte Carlo thích nghi (Adaptive Monte Carlo Localization – AMCL) sử dụng các hạt có trọng số để xấp xỉ vị trí và hướng của Robot từ đó lập được kế hoạch đường đi.

Từ khóa: *ROS, SLAM, Mobile Robot, Path Planning, Navigation Stack, Rapidly Exploring Random Tree, Adaptive Monte Carlo Localization, PiMouse.*

Mục lục

Lời cảm ơn	i
Tóm tắt	ii
Chương 1 Đặt vấn đề	1
Chương 2 Nền tảng lý thuyết	2
2.1 Phương pháp SLAM tự động sử dụng khám phá cây ngẫu nhiên RRT	2
2.1.1 Phát hiện điểm biên cục bộ và toàn cục	4
2.1.2 Mô đun Bộ lọc	6
2.1.3 Mô đun phân bổ nhiệm vụ Robot	7
2.2 Phương pháp AMCL	9
2.2.1 Nền tảng lý thuyết về bộ lọc hạt	9
2.2.2 Áp dụng phương pháp vào bài toán định vị robot	10
Chương 3 Phần cứng sử dụng	12
3.1 Raspberry Pi	12
3.2 Động cơ Step	13
3.3 Lidar	14
3.4 Phần cứng sử dụng	15
Chương 4 Thực nghiệm	16
4.1 Thuật toán SLAM sử dụng hector-SLAM	16

4.2	Thuật toán SLAM tự động sử dụng RRT và điều hướng	17
4.2.1	Tự động SLAM sử dụng RRT và Hector	17
4.2.2	Điều hướng	20
4.2.3	Định vị robot trên bản đồ đã dựng sử dụng Adaptive Monte Carlo Localization (AMCL)	21
4.2.4	Kết quả sau khi chạy mô phỏng trên Rviz	23
4.2.5	Điều hướng Robot	23
Chương 5	Kết quả và Đánh giá	28
Tài liệu tham khảo		30

Chương 1

Đặt vấn đề

Diều hướng robot hiệu quả đòi hỏi phải xác định trước bản đồ. Các thuật toán thăm dò tự động khác nhau, hướng Robot đến những không gian chưa được khám phá bằng cách phát hiện các điểm biên giới (Detecting frontiers). Trong đó, biên giới là ranh giới ngăn cách không gian chưa biết và không gian đã biết. Thông thường phát hiện biên giới sử dụng các công cụ xử lý ảnh giống như việc phát hiện cạnh, do đó giới hạn nó ở việc khám phá bản đồ 2D.

Bài viết này trình bày một chiến lược thăm dò mới sử dụng trên việc sử dụng nhiều cây ngẫu nhiên khám phá nhanh chóng RRT [3]. Thuật toán RRT được chọn bởi vì, hướng tới nhanh chóng đến các khu vực chưa được khám phá. Ngoài ra, sử dụng RRT cung cấp một vị tướng tiếp cận có thể được mở rộng đến không gian chiều cao hơn và các hệ thống đa Robot.

Chiến lược được đề xuất được thực hiện và thử nghiệm bằng cách sử dụng cách package được hỗ trợ trong hệ điều hành ROS. Sử dụng thuật toán RRT cho Robot để lập kế hoạch đường đi đến tất cả các điểm cuối xa có thể tiếp cận trong vùng lân cận của cảm biến hay còn gọi là các điểm biên giới (frontier points), làm cho bản đồ Robot ở các vùng mới được liên tục bằng cách sử dụng SLAM khi nó cố gắng đi đến điểm cuối xa mới của nó.

Áp dụng RRT cho Robot di động (Mobile robots) theo cách như vậy cho phép chúng ta tạo một Robot tự trị tự khám phá (self-exploring autonomous robot) mà không cần sự can thiệp của con người.

Chương 2

Nền tảng lý thuyết

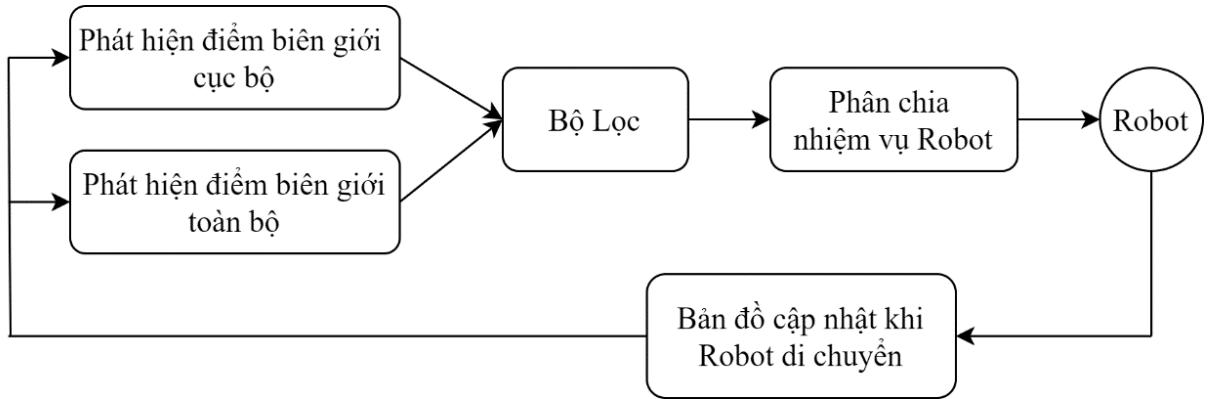
2.1 Phương pháp SLAM tự động sử dụng khám phá cây ngẫu nhiên RRT

Mục tiêu chính của việc khám phá môi trường tự động của Robot là đi đến những không gian không xác định, do đó mở rộng phần đã biết và tạo dựng được bản đồ. Việc thăm dò dựa trên các điểm biên giới thường được sử dụng, hướng Robot đến các cạnh biên giới của bản đồ hiện tại.

RRT là một thuật toán lập đường đi cho Robot bằng cách lău mẫu không gian bằng các điểm được tạo ngẫu nhiên. Các điểm ngẫu nhiên được sử dụng tạo cấu trúc cây, bao gồm các đỉnh và các cạnh. Trong phương pháp thăm dò dựa trên RRT, thay vì cấu trúc cây giúp robot điều hướng đường dẫn cấu trúc cây giúp tìm kiếm các điểm biên giới, các tìm kiếm này chạy độc lập từ các chuyển động của Robot. Sơ đồ của thuật toán thăm dò được thể hiện như 2.1.

Thuật toán được chia làm 3 mô đun: Phát hiện điểm biên giới (bao gồm biên giới cục bộ và toàn cục), Bộ lọc và Phân chia nhiệm vụ Robot. Các điểm biên giới được đi qua bộ lọc, mô đun bộ lọc có trách nhiệm phân cụm và lưu trữ chúng, ở đây sử dụng thuật toán phân cụm sự thay đổi trung bình.

Mô đun bộ lọc cũng xóa những điểm biên giới không hợp lệ và cũ. Sau khi qua Bộ lọc, các điểm biên giới được Mô đun Phân bổ tác vụ Robot gác cho Robot làm việc. Khi một điểm được gán cho Robot để khám phá, Robot di chuyển về



Hình 2.1: Sơ đồ thuật toán thăm dò RRT

phía điểm được chỉ định.

Trong quá trình này, các cảm biến trên Robot (ví dụ như máy quét laser) quét và khám phá các khu vực lân cận trong phạm vi cảm biến. Một lợi thế của phương pháp này là có thể sử dụng nhiều cây phát triển độc lập để tăng tốc độ tìm kiếm các điểm biên giới.

Một số thuật ngữ sử dụng trong thuật toán:

Map X: Tập hợp toàn bộ không gian bao gồm: không gian chứa chướng ngại vật, không gian trống và không gian chưa biết (chưa được khám phá).

Occupancy grid: Bản đồ 2D được chia thành các ô vuông, mỗi ô được đánh số theo tọa độ của nó và mỗi ô có giá trị là 1 nếu bị có chướng ngại vật, 0 nếu trống, -1 là chưa được khám phá.

Free Space X_{free} : Tập hợp không gian trống, tức là không gian đã được khám phá và không bị chiếm đóng bởi chướng ngại vật.

Unknown region: Tập hợp không gian chưa được khám phá.

Vertices V: Tập hợp node hay đỉnh trong map, đối với thuật toán RRT các điểm được nối với nhau bởi các nhánh (các cạnh), mỗi điểm trên cây được gọi là một đỉnh, các đỉnh này được lưu vào V.

Edge E: Cạnh là cách nhánh nối liền hai đỉnh của cây, các cạnh được lưu vào E.

Graph G: Tập hợp các đỉnh và cạnh tạo thành Graph, trong RRT $G = (V, E)$

tạo thành một cấu trúc cây.

SampleFree: Hàm trả về một điểm ngẫu nhiên phân phối độc lập giống hệt nhau trên map X.

Nearest: Hàm có đầu vào là $G = (V, E)$, $x \in X_{\text{free}}$ và trả về một điểm $v \in V$ sao cho $\text{Nearest}(G = (V, E), x) = \operatorname{argmin}_{v \in V} \|x - v\|$.

Steer: Hàm có đầu vào là hai điểm x, y và trả về điểm z thỏa mãn $\|z - y\|$ đạt giá trị nhỏ nhất, $\|z - x\| \leq \eta$ với η là tốc độ tăng trưởng của cây.

Grid Check: Hàm có đầu vào là Map X và hai điểm. Nó trả về 0 nếu có vật cản giữa hai điểm, trả về -1 nếu là vùng không xác định, trả về 1 nếu các điểm nằm trong không gian trống đã biết.

PublishPoint: Hàm gửi các điểm biên tới Mô đun Bộ lọc.

2.1.1 Phát hiện điểm biên cục bộ và toàn cục

Mô đun phát hiện biên giới dựa trên RRT có nhiệm vụ tìm các điểm không nằm trong vùng xác định của bản đồ. Trong quá trình thực hiện, bản đồ được biểu diễn dưới dạng Occupancy Grid, ban đầu tất cả đều được đánh dấu là không xác định. Do vậy việc đọc giá trị các ô trên Grip sẽ giúp nhận diện được các điểm biên giới. Dưới đây, mô tả chi tiết hai phiên bản dò các điểm biên: biên cục bộ và biên toàn cục.

Phát hiện biên cục bộ (Local Frontier Detector)

Thuật toán Phát hiện biên cục bộ được thể hiện ở trong thuật toán 2.1. Thuật toán bắt đầu với việc khởi tạo một đỉnh $V = \{x_{\text{init}}\}$, và đặt $E = \emptyset$, tại mỗi lần lặp lại điểm ngẫu nhiên được lấy mẫu $x_{\text{rand}} \subset X_{\text{free}}$. Đỉnh đầu tiên của cây gần nhất với x_{rand} được tìm thấy, được gọi là $x_{\text{nearest}} \subset V$. Sau đó Hàm Steer tạo một điểm x_{new} . Hàm GridCheck kiểm tra nếu x_{new} nằm trong vùng chưa xác định, hoặc có bất kỳ điểm nào nằm trong đoạn nối x_{new} và x_{nearest} trong vùng chưa xác định thì x_{new} được coi là một điểm biên giới. Điểm x_{new} được gửi đến mô đun Lọc và cây được reset, các đỉnh và cạnh được xóa. Sau đó, vòng lặp tiếp tục tại vị trí hiện tại của Robot $V = x_{\text{current}} ; E = \{\emptyset\}$.

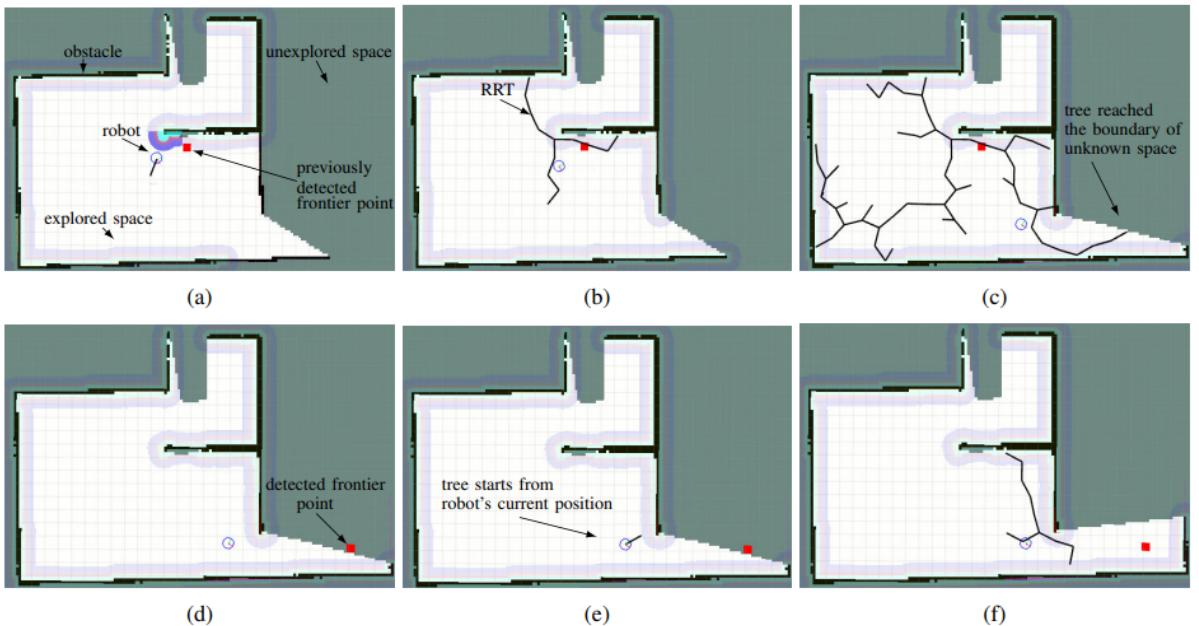
Algorithm 2.1: Phát hiện biên giới cục bộ

```
1  $V \leftarrow x_{\text{init}} ; E \leftarrow \emptyset;$ 
2 while True do
3    $x_{\text{rand}} \leftarrow \text{SampleFree};$ 
4    $x_{\text{nearest}} \leftarrow \text{Nearest}(G(V, E), x_{\text{rand}});$ 
5    $x_{\text{new}} \leftarrow \text{Steer}(x_{\text{nearest}}, x_{\text{rand}}, \eta);$ 
6   if GripCheck(map,  $x_{\text{nearest}}, x_{\text{new}}) = -1$  then
7     PublishPoint( $x_{\text{new}});$ 
8      $V \leftarrow x_{\text{current}} ; E \leftarrow \emptyset;$ 
9   else if GridCheck(map,  $x_{\text{nearest}}, x_{\text{new}}) = 1$  then
10     $V \leftarrow V \cup \{x_{\text{new}}\} ; E \leftarrow E \cup \{(x_{\text{nearest}}, x_{\text{new}})\};$ 
11  end if
12 end while
```

Trong trường hợp còn lại, không có trờ ngại nào tại x_{new} thì cây được mở rộng bằng cách thêm x_{new} như một đỉnh mới, một cạnh cũng được tạo ra giữa x_{new} và x_{nearest} . Đối với mỗi Robot chạy Mô đun dò biên giới địa phương, mỗi cây được tạo ra theo quá trình ở trên. Một khi cây đến một khu vực không xác định, một điểm biên giới được đánh dấu và cây được thiết lập lại. Quá trình này xảy ra trong thời gian Robot chuyển động, do đó cây phát triển từ một điểm ban đầu mới khi nó được đặt lại.

Dòng 8 trên thuật toán 2.1. Mô đun dò biên giới địa phương phát hiện nhanh chóng các điểm biên giới trong vùng lân cận của Robot bất kỳ lúc nào. Hình 2 cho thấy sự phát triển của một cây RRT địa phương và quá trình phát hiện một điểm biên giới.

Hình (a) cây bắt đầu từ vị trí hiện tại của Robot, trong (b) và (c) cây tiếp tục phát triển, trong (d) một đỉnh của cây nằm trong khu vực không xác định, được đánh dấu là một điểm biên giới và cây được thiết lập lại. Trong (e) và (f) vòng lặp lặp lại nơi cây phát triển lại từ vị trí hiện tại của Robot.



Hình 2.2: Quá trình phát triển cây và phát hiện điểm biên giới địa phương

Phát hiện biên toàn cục (Global Frontier Detector)

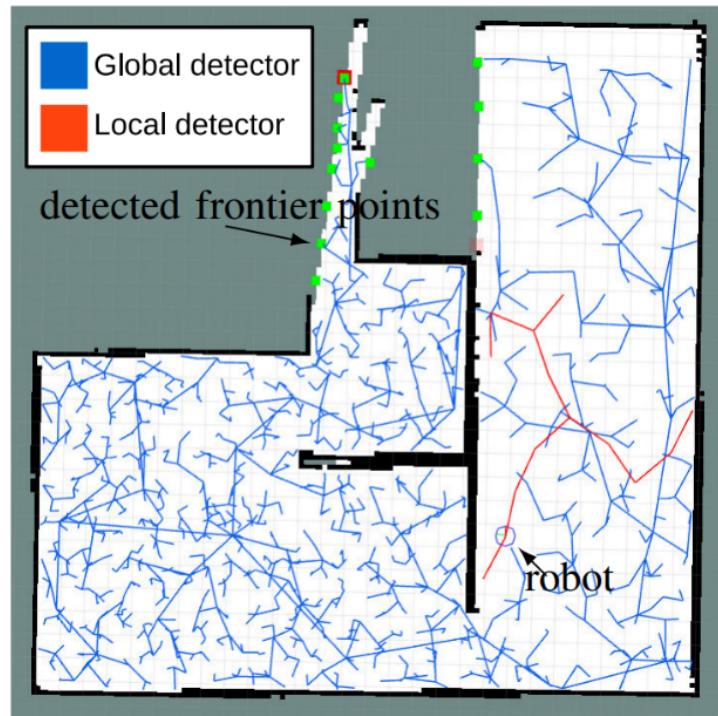
Thuật toán phát hiện biên toàn cục giống hệt với thuật toán phát hiện biên cục bộ đã được mô tả ở Thuật toán 2.1, ngoại trừ Dòng 8 được gỡ bỏ. Cây không thiết lập lại và tiếp tục phát triển trong toàn bộ thời gian thăm dò (tức là cho đến khi bản đồ được khám phá hoàn toàn), điều này làm thuật toán

Phát hiện biên giới toàn cần tương tự như RRT. Máy dò biên giới toàn cầu dùng để phát hiện các điểm biên giới thông qua toàn bộ bản đồ và ở những khu vực xa của Robot. Hình 2.3 minh họa việc phát triển các cây RRT thông qua bởi việc thăm dò biên giới cục bộ và toàn cục.

2.1.2 Mô đun Bộ lọc

Mô đun bộ lọc nhận các điểm biên giới được phát hiện từ tất cả các máy dò biên giới địa phương và toàn cục. Mô đun bộ lọc đầu tiên sẽ phân cụm các điểm biên giới và chỉ lưu trữ tâm của cụm, các điểm còn lại sẽ bị loại bỏ.

Sự phân cụm và loại bỏ là quá trình cần thiết để giảm số lượng các điểm biên giới do quá trình dò sẽ phát hiện ra vô số các điểm biên giới cực kỳ gần nhau. Nếu



Hình 2.3: Thăm dò biên toàn cục và cục bộ

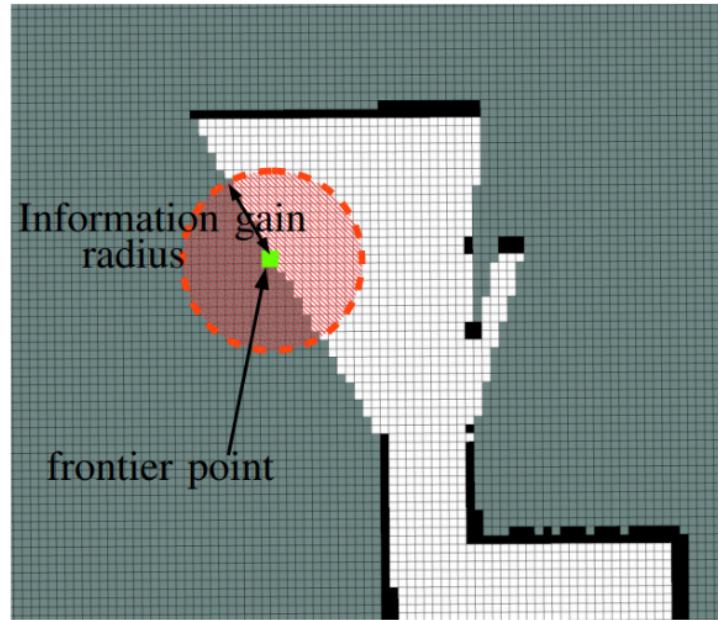
những điểm như vậy được gửi đến Mô đun phân bổ nhiệm vụ Robot sẽ tồn tài nguyên để tính toán mà không có thông tin bổ sung về bản đồ nhất thiết phải đạt được.

Mô đun bộ lọc cũng xóa các điểm biên giới không hợp lệ và cũ trong mỗi lần lặp lại. Trong đó, điểm biên giới cũ là các điểm biên giới được phát hiện trước đó nhưng giờ không còn nằm trong vùng không xác định của bản đồ. Điểm biên giới không hợp lệ là điểm biên giới mà Robot không thể tiếp cận một cách vật lý, tức là không có con đường nào tồn tại giữa vị trí của Robot và điểm biên giới đó.

2.1.3 Mô đun phân bổ nhiệm vụ Robot

Mô đun này nhận các điểm biên giới từ bước trên và gán chúng cho Robot. Thiết kế của mô đun Phân bổ nhiệm vụ Robot tương tự như]. Mô đun phân bổ nhiệm vụ robot chỉ định các điểm để Robot khám phá bằng cách xem xét các điều sau:

Navigation cost (N): được định nghĩa là khoảng cách dự kiến được di chuyển bởi một Robot để đến được điểm biên giới.



Hình 2.4: Vùng lợi ích thông tin của một điểm biên giới

Information gain (I): được định nghĩa là khu vực dự kiến sẽ được khám phá cho một điểm biên giới nhất định. Lợi ích thông tin được định lượng bằng cách đếm số các ô không xác định xung quanh một điểm biên giới trong một bán kính xác định. Bán kính này được gọi là thông tin bán kính đạt được, nên được đặt thành một giá trị bằng với phạm vi cảm biến nhận thức. Vùng đó sau đó được tính bằng cách nhân số ô với diện tích mỗi ô (diện tích mỗi ô được tính từ độ phân giải của bản đồ). Trong Hình 2.4 , lợi ích thông tin thu được khoảng 18.1 m^2 (có 181 ô vuông trong vùng không xác định xung quanh bán kính cho phép, mỗi ô vuông có diện tích 0.1 m^2)

Revenue from a frontier point (R): Đối với một điểm biên giới nhất định x_{fp} , và một vị trí hiện tại của Robot x_r , the revenue R thu được từ việc khám phá x_{fp} , được tính như sau:

$$R(x_{fp}) = \lambda h(x_{fp}, x_r) I(x_{fp}) - N(x_{fp})$$

$$h(x_{fp}, x_r) = \begin{cases} 1, & \text{if } \|x_r - x_{fp}\| > h_{rad} \\ h_{gain}, & h_{gain} > 1 \end{cases}$$

Sau đó, điểm có giá trị R cao nhất sẽ được Robot khám phá.

2.2 Phương pháp AMCL

Phương pháp AMCL (Adaptive Monte Carlo Localization) là phương pháp định vị sử dụng bộ lọc hạt dự đoán vị trí của robot trong bản đồ có sẵn. Ưu điểm của bộ lọc hạt so với các bộ lọc khác như bộ lọc Kalman là bộ lọc hạt có thể áp dụng cho hệ thống phi tuyến, cho phép ước lượng vị trí của robot mà không cần biết các vị trí trước đó, tương thích với nhiều mô hình nhiễu khác nhau và không bị ràng buộc bởi các điều kiện.

2.2.1 Nền tảng lý thuyết về bộ lọc hạt

[4]

Bộ lọc hạt được xây dựng dựa trên mô hình dự đoán và hiệu chỉnh, kết hợp thông tin không chắc chắn ở thời điểm hiện tại với dữ liệu từ môi trường để dự đoán được thông tin trong tương lai. Ý tưởng của bộ lọc hạt là đưa ra một phép xấp xỉ dựa trên định lý Bayes.

Các bước dự đoán và hiệu chỉnh được thể hiện như sau:

Bước 1: Dự đoán

Để ước lượng vị trí tại x_k với các phép đo z trước đó, ta cần tính được phép ước lượng tại thời điểm x_{k-1} với các phép đo z , đồng thời biết được xác suất để robot chuyển từ vị trí x_{k-1} sang x_k dựa trên mô hình của robot. Khi biết được hàm mật độ xác suất, ta tính xác suất xảy ra trong một vùng nhất định bằng cách tích phân chúng:

$$p(x_k | z_{1:k-1}) = \int p(x_k | x_{k-1}) p(x_{k-1} | z_{1:k-1}) dx_{k-1}$$

Bước 2: Hiệu chỉnh

Sử dụng định lý Bayes:

$$p(x_k | z_{1:k}) = \frac{p(z_k | x_k) p(x_k | z_{1:k-1})}{p(z_k | z_{1:k-1})}.$$

Dựa trên mô hình dự đoán và hiệu chỉnh, ta có thể ước lượng được vị trí dựa trên trọng số và sự sai khác giữa vị trí thực của robot với các hạt được phân bổ

trong môi trường. Cụ thể, trọng số sẽ thể hiện sự sai khác của các dữ liệu đo được:

$$p(x_k | z_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(x_k - x_k^i)$$

Trọng số giữa các vị trí của các hạt với vị trí của robot được tính dựa trên định lý Bayes, tỷ lệ thuận với phép đo và vị trí của robot. Sau đó tiến hành chuẩn hóa lại để đảm bảo tổng các trọng số đều bằng 1 bằng cách chia cho importance density q .

$$w_k^i \propto w_{k-1}^i \frac{p(z_k | x_k^i) p(x_k^i | x_{k-1}^i)}{q(x_k^i | x_{k-1}^i, z_k)}$$

2.2.2 Áp dụng phương pháp vào bài toán định vị robot

Giải thuật của Particle Filter được thể hiện dưới đây:

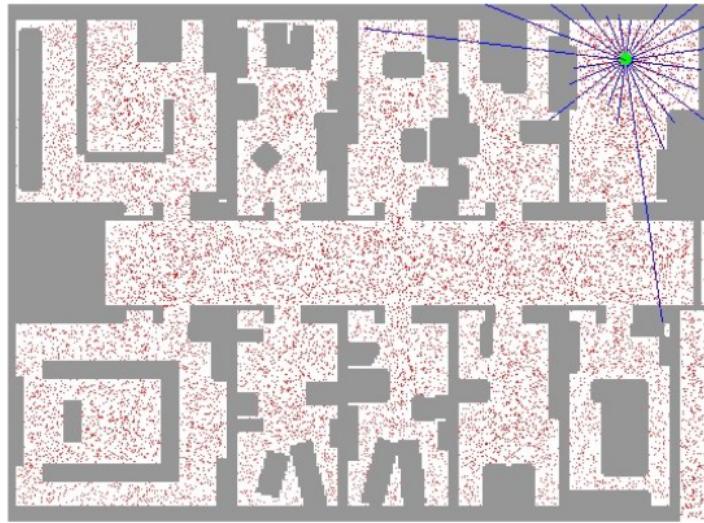
Algorithm 2.2: Thuật toán Particle filter

```

1  $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$ 
2 for  $m = 1$  to  $M$  do
3   sample  $x_t^{[m]} \sim p(x_t | u_t, x_{t-1}^{[m]})$ 
4    $w_t^{[m]} = p(z_t | x_t^{[m]})$ 
5    $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$ 
6 endfor
7 for  $m = 1$  to  $M$  do
8   draw  $i$  with probability  $\propto w_t^{[i]}$ 
9   add  $x_t^{[i]}$  to  $\mathcal{X}_t$ 
10 endfor
11 return  $\mathcal{X}_t$ 
```

Dầu tiên, cho các hạt phân bố ngẫu nhiên trong không gian bản đồ được cung cấp sẵn. Từ mỗi hạt, tính phép đo từ vị trí của hạt tới laser. So sánh vị trí từ hạt đến laser và vị trí của robot, tiến hành tính importance weight dựa trên các dữ liệu đo được bằng phân bố chuẩn Gauss và chuẩn hóa lại.

Sau mỗi bước, lấy mẫu lại các hạt dựa trên importance weight sao cho nhiều hạt được phân bổ ở những vị trí có trọng số cao hơn, vì tại những vị trí này có



Hình 2.5: Adaptive Monte Carlo Localization

khả năng cao gần với vị trí thực của robot. Sau khi lấy mẫu lại, vị trí ước lượng của robot sẽ là trung bình vị trí của các hạt được gắn với các trọng số.

Sau khi ước lượng được vị trí của robot, cho robot dịch chuyển theo mô hình động học và trở lại với bước dự đoán.

Phương pháp định vị Adaptive Monte Carlo (AMCL) là một giải thuật ước lượng tọa độ của robot từ cả thông tin odometry, cảm biến, bản đồ dựa trên bộ lọc hạt, được minh họa trong Hình 2.5

Chương 3

Phần cứng sử dụng

3.1 Raspberry Pi

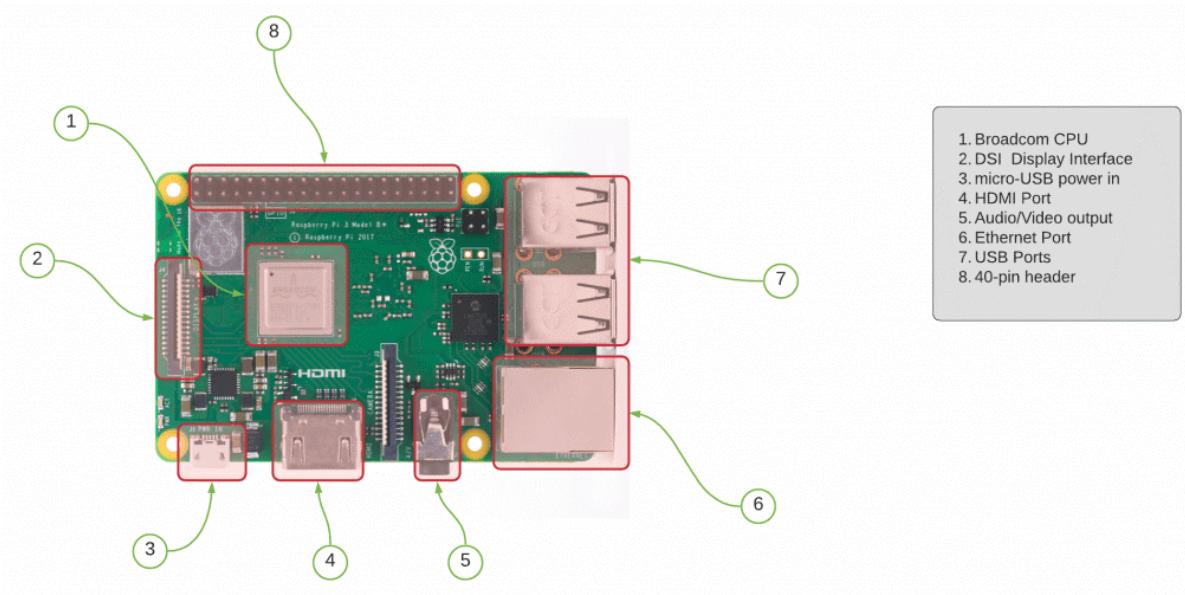
Raspberry Pi (Hình 3.1) là máy tính nhúng có kích thước nhỏ và chạy hệ điều hành Linux, được phát triển bởi Raspberry Pi Foundation – là tổ chức phi lợi nhuận với tiêu chí xây dựng hệ thống mà nhiều người có thể sử dụng được trong những công việc tùy biến khác nhau.

Đặc tính của Raspberry Pi xây dựng xoay quanh bộ xử lý SoC Broadcom BCM2835 (là chip xử lý mobile mạnh mẽ có kích thước nhỏ hay được dùng trong điện thoại di động) bao gồm CPU, GPU , bộ xử lý âm thanh /video , và các tính năng khác . . . tất cả được tích hợp bên trong chip có điện năng thấp này.

Một số ưu - nhược điểm của Raspberry Pi.

Ưu điểm:

- Giá rẻ
- Nhỏ gọn
- Siêu tiết kiệm điện
- GPU mạnh
- Phục vụ cho nhiều mục đích.



Hình 3.1: Raspberry Pi

- Khả năng hoạt động liên tục 24/7.

Nhược điểm:

- CPU cấu hình thấp
- Yêu cầu phải có kiến thức cơ bản về Linux, điện tử.

Ứng dụng của Raspberry Pi:

- Web server
- Mạng nội bộ IoT
- Điều khiển cảm biến
- Điều khiển Robot
- Điều khiển các thiết bị công nghiệp

3.2 Động cơ Step

Step Motor (Stepper Motor, Stepping Motor) được minh họa trong Hình 3.2, đều là những từ khóa chỉ về động cơ bước. Step Motor là loại động cơ chấp hành đặc



Hình 3.2: Động cơ Step

biệt, thường được sử dụng cho các hệ truyền động rời rạc. Step Motor thực chất là một động cơ đồng bộ dùng để biến đổi các tín hiệu điều khiển dưới dạng các xung điện rời rạc kế tiếp nhau thành các chuyển động góc quay hoặc các chuyển động của Rotor và có khả năng cố định Rotor vào những vị trí cần thiết.

Nếu góc bước của động cơ càng nhỏ thì số bước trên mỗi vòng quay của nó lại càng lớn và độ chính xác đối với vị trí thu được cũng càng lớn. Các góc bước của động cơ có thể lớn tới 90 độ và nhỏ nhất đến 0,72 độ. Tuy nhiên, các góc bước thường được sử dụng nhiều nhất là 1,8 độ và 2,5 độ hoặc 7,5 độ và 15 độ.

3.3 Lidar

Lidar (Hình 3.3) là tên viết tắt của cụm từ Light Detection And Ranging. Bằng cách phát đi đi một chùm tia laser rồi thu nhận lại tín hiệu phản hồi. Tốc độ ánh sáng đã biết trước, độ trễ phản hồi được ghi nhận, từ đó tính được khoảng cách giữa máy phát và vật thể một cách tương đối chính xác. Sự khác biệt về thời gian và bước sóng laser sau đó có thể được sử dụng để tạo mô hình số 3 chiều của đối tượng.

Một số ứng dụng quan trọng của Lidar:

- Ngành công nghiệp oto, Robot

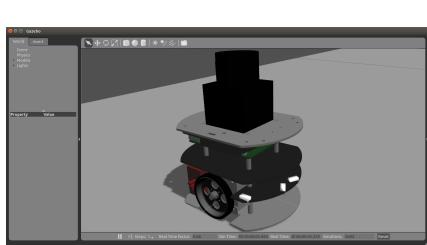


Hình 3.3: Lidar Hokouyo

- Nghiên cứu địa hình, vẽ bản đồ
- Lidar trên các thiết bị cầm tay

3.4 Phần cứng sử dụng

PiMouse được sử dụng trong Gazebo được mô phỏng trong Hình 3.4a và phần cứng thực tế sử dụng được minh họa trong Hình 3.4b.



(a) PiMouse mô phỏng trong ROS



(b) PiMouse thực tế

Hình 3.4: PiMouse

Chương 4

Thực nghiệm

- Toàn bộ source code cho mô phỏng có thể tìm thấy ở:
github.com/irobo197/raspimouse_sim.git
- Toàn bộ source code cho chạy thực có thể tìm thấy ở:
github.com/irobo197/pimouse.git

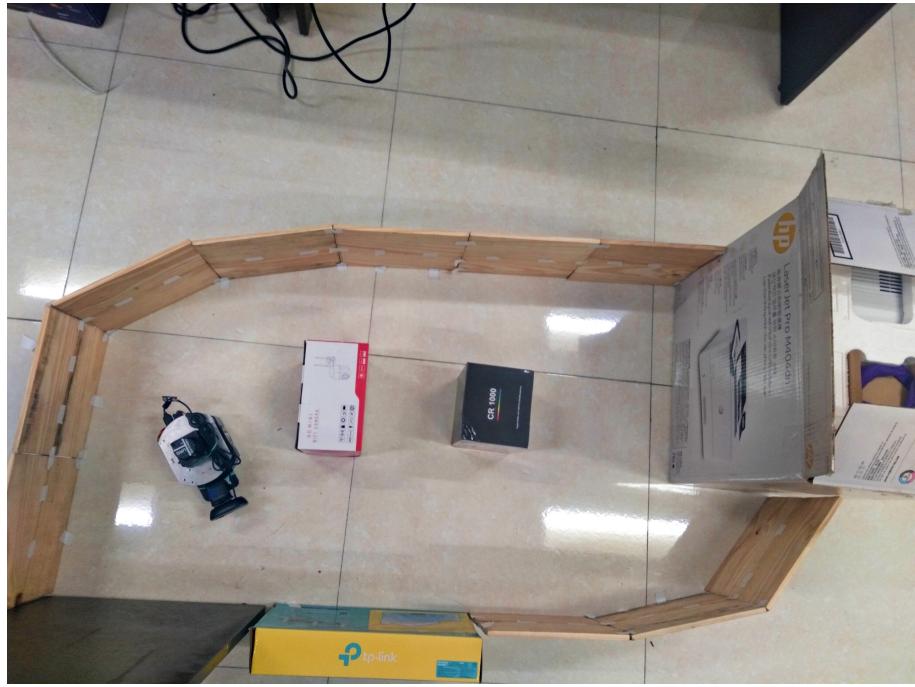
4.1 Thuật toán SLAM sử dụng hector-SLAM

Quá trình thực nghiệm được thực hiện trên Robot Raspberry Pi Mouse đã trình bày trong hình 3.4, với bộ nhớ 16GB, RAM 1GB, kết nối với client có cấu hình RAM 8GB, chip Intel core i5-10th-H. Môi trường có kích thước 1.8x1.5 được thể hiện trong hình 4.1.

Sử dụng lệnh sau để chạy quá trình SLAM với hector và Raspberry Pi Mouse:

1. Trên robot: rosrun pimouse_slam robot_side.launch
2. Trên client: rosrun pimouse_slam desktop_side.launch

Thuật toán được tính toán trên cơ sở client, sau khi đã nhận thông tin về laser có được từ topic `/scan` trên robot. Do khối lượng tính toán hector nếu đặt trên trực tiếp PiMouse, khi đó quá trình thực thi và cập nhật sẽ chậm, có thể kết quả sẽ không được như kỳ vọng, phần cứng tràn. Dựa trên thực tế yêu cầu, nhóm đã kết nối điều khiển PiMouse với máy tính cá nhân, thông qua TCP/IP. Khi đó,



Hình 4.1: Môi trường thử nghiệm

- Phía robot sẽ khởi tạo node điều khiển thông qua bàn phím, khởi tạo node chạy Lidar
- Phía máy tính cá nhân sẽ được sử dụng để chạy quá trình tính toán và tái tạo bản đồ trên Rviz

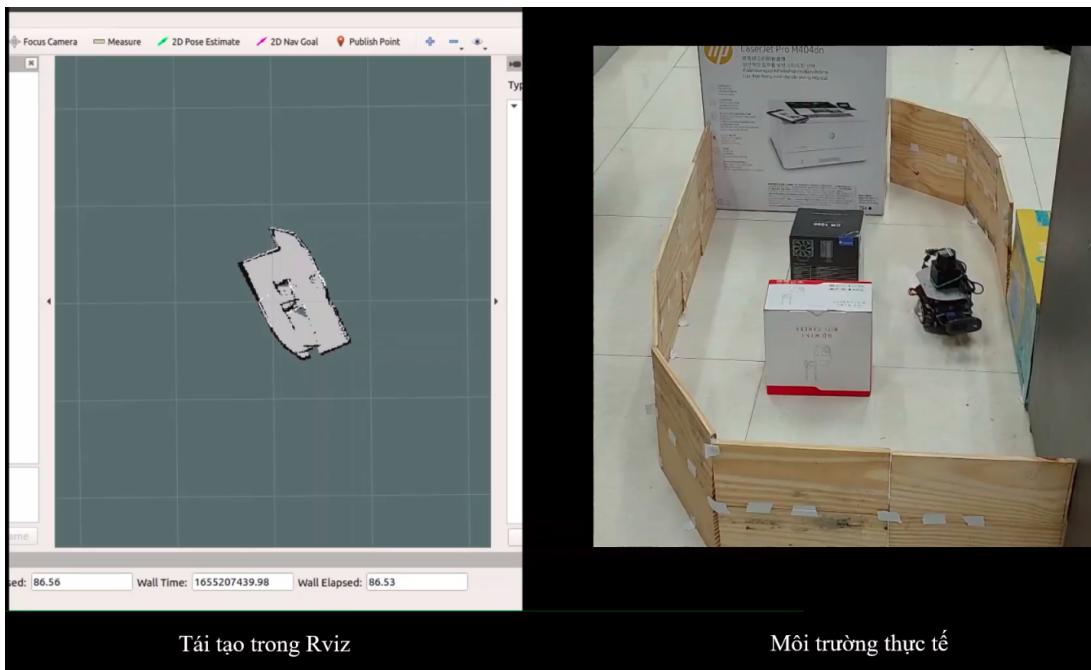
Quá trình thực hiện được thể hiện trong hình 4.2. Video đầy đủ có thể tìm thấy ở: <https://youtu.be/2DWmaTsI01s>

Hình 4.3 là bản đồ sau khi thực nghiệm với hector_slam trên RaspberryPi-Mouse

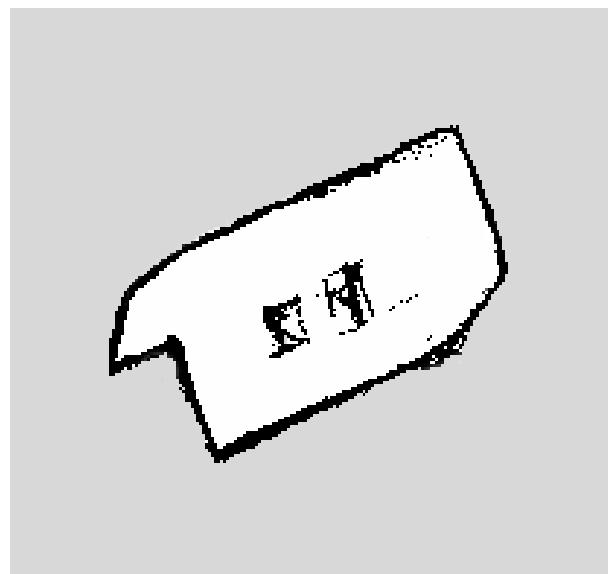
4.2 Thuật toán SLAM tự động sử dụng RRT và điều hướng

4.2.1 Tự động SLAM sử dụng RRT và Hector

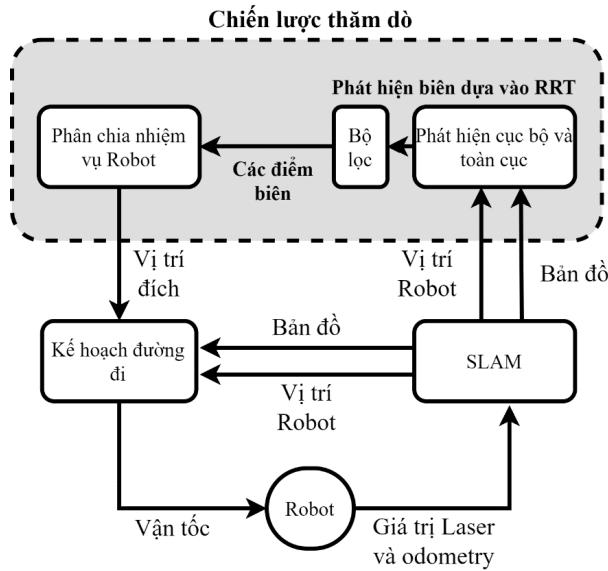
Quá trình được thực hiện mô phỏng thông qua mô hình và môi trường cấu trúc trong Gazebo. Hình 4.4 mô tả toàn bộ thuật toán khi được triển khai, bao gồm:



Hình 4.2: Thực nghiệm quá trình SLAM với phương pháp Hector-SLAM



Hình 4.3: Bản đồ tái tạo sau quá trình SLAM với phương pháp Hector-SLAM



Hình 4.4: Lưu đồ thuật toán SLAM tự động

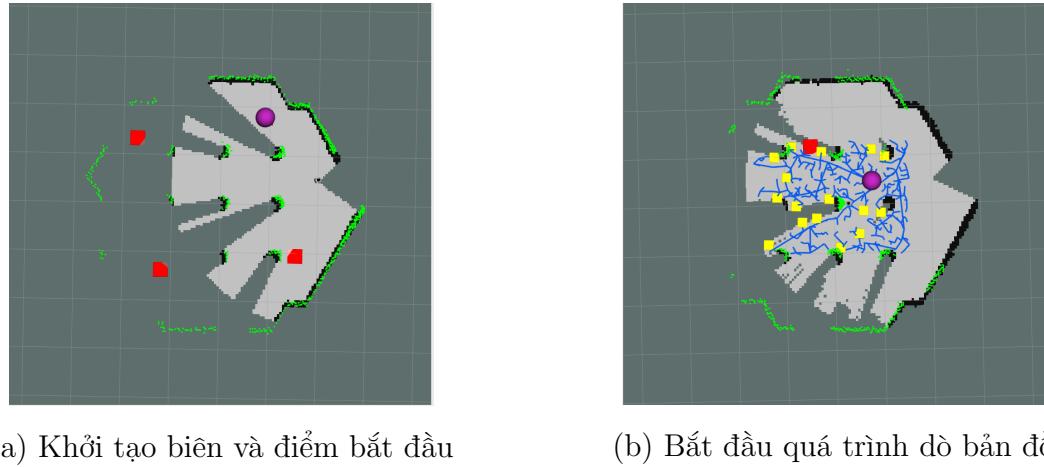
- Mô đun SLAM
- Mô đun lập kế hoạch đường dẫn
- Mô đun Phát hiện biên toàn cục và cục bộ
- Mô đun Bộ lọc, Mô đun Phân bổ nhiệm vụ Robot

Các package khác nhau của ROS được sử dụng để lập bản đồ và lập kế hoạch đường đi. Còn lại khôi thăm dò dựa trên RRT đã được mô tả như các phần trên.

Package Gmapping trong ROS được sử dụng để tạo bản đồ và định vị cho Robot. Gmapping là một thuật toán SLAM sử dụng Rao-Blackwellized particle filter [1]. Tuy nhiên, Gmapping phụ thuộc vào phần thông tin nhận được có cả từ odometry, vì thế, yêu cầu một phần cứng tốt để giảm thiểu khối lượng tính toán và xử lý nhiễu là cực kỳ cần thiết. Trong khi đó, thuật toán hector-slam được đề xuất [2] thể hiện tốt hơn trong bài toán SLAM khi không cần phụ thuộc vào odometry. Vì thế quá trình SLAM sẽ được hector phụ trách sau khi đã có điểm đích được quyết định bởi cây RRT, đã trình bày chi tiết trong Chương 2.

Hình 4.5 thể hiện quá trình SLAM với RRT, Hector-SLAM. Sử dụng môi trường cung cấp bởi turtlebot3 là *turtlebot3_world*, thực hiện xong quá trình SLAM

trong 10s.



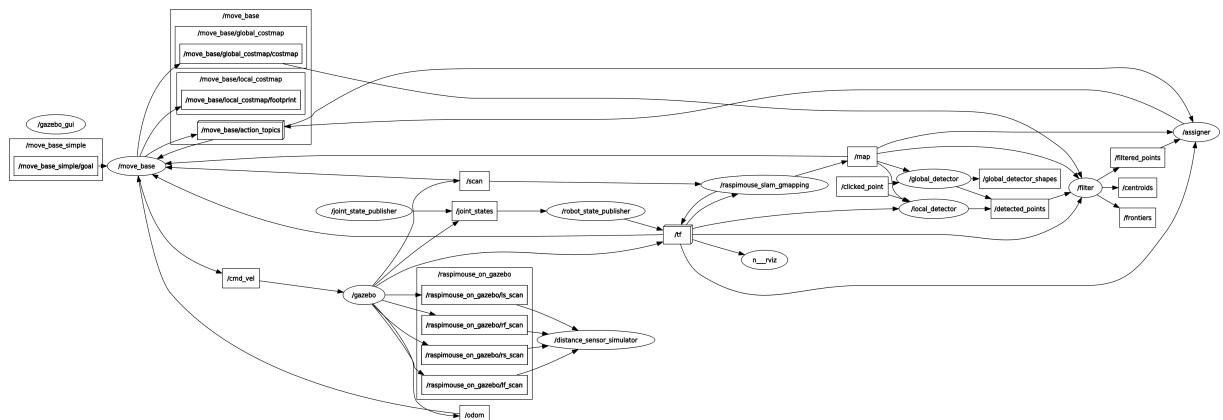
Hình 4.5: SLAM sử dụng RRT và Hector

Việc thực hiện kết nối, giao tiếp giữa các topic được thể hiện trong hình 4.6

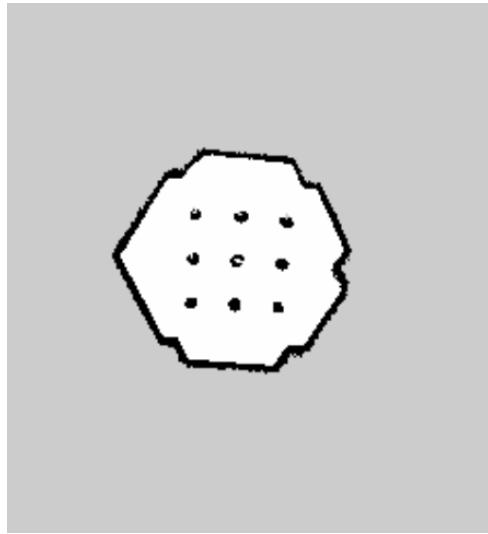
Hình 4.7 là bản đồ sau khi được tái tạo và sử dụng cho mục đích điều hướng:

4.2.2 Điều hướng

ROS Navigation stack được sử dụng để điều khiển và điều hướng Robot tới các điểm đích để khám phá (ở đây là các điểm biên được lựa chọn x_{fp} . Path planning dựa trên thuật toán A* [3] là một package có sẵn trong ROS navigation stack, được dùng để lập kế hoạch đường đi đến điểm đích theo vị trí hiện tại của Robot.



Hình 4.6: Lưu đồ quá trình thực hiện RRT và Hector SLAM



Hình 4.7: Bản đồ tái tạo lại sau quá trình thực hiện RRT và Hector SLAM

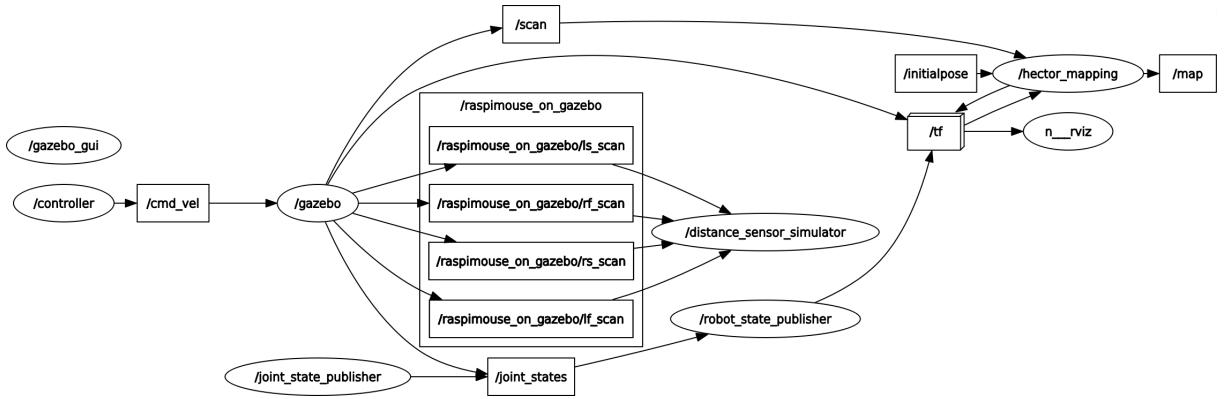
4.2.3 Định vị robot trên bản đồ đã dựng sử dụng Adaptive Monte Carlo Localization (AMCL)

Package *map_server* trong ROS Navigation Stack có cung cấp node *map_saver* cho phép truy xuất data từ ROS Service để lưu lại bản đồ. Command trên terminal lệnh: `rosrun map_server map_saver -f{ name }`, trong đó {name} được thay bằng tên của map, *-f* là thuộc tính cho phép lưu tên file, nếu không sử dụng *-f*, tên file sẽ được lưu tự động là "map". File được lưu về sẽ nằm trong workspace bao gồm file ảnh pgm và file mô tả ảnh yaml, được minh họa trong Hình 4.7

File ảnh pgm sẽ biểu diễn các không gian bị chiếm trong môi trường: pixel màu trắng là toàn bộ khoảng trống, vết đen là các vật cản (tường) và khoảng xám là vùng robot không đi qua. File yaml chứa các thông tin của bản đồ, bao gồm: tên file, độ phân giải,...

Command lệnh `rosrun rqt_graph rqt_graph` để kiểm tra luồng dữ liệu và sự liên kết giữa các node, đảm bảo các node subscribe và publish đúng với hệ thống (Hình 4.8).

Thuật toán AMCL là một hệ thống định vị dựa trên xác suất cho robot di chuyển trong môi trường 2D, sử dụng bộ lọc hạt *particle filter*. Package AMCL sử dụng thông tin từ laser, map và tf để dự đoán gần chính xác vị trí và hướng của robot.



Hình 4.8: Sơ đồ các node trong hector

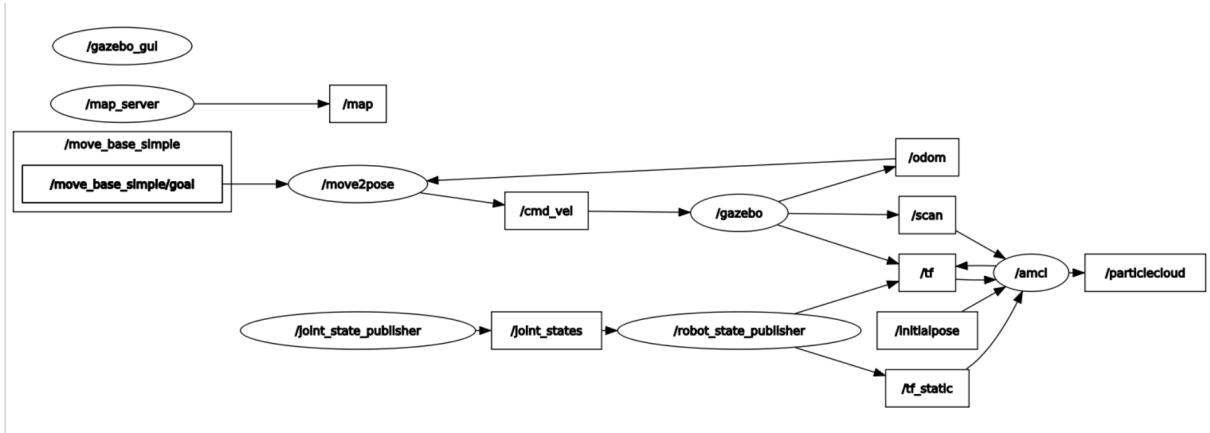
Sau khi launch Rviz, add LaserScan, Map và Pose Array để quan sát quá trình định vị của robot. Localization ở ROS được thể hiện thông qua các particles.

Sự phân bố của hàng loạt các particle cloud trên bản đồ biểu diễn những dự đoán khác nhau về vị trí của robot. Khi robot di chuyển, particle cloud sẽ được phân bổ lại dựa trên dữ liệu từ laser, loại bỏ những particles có khả năng thấp và tạo ra nhiều particles ở những vị trí có xác suất lớn hơn, từ đó cho được kết quả chính xác về vị trí và hướng của robot. Vì vậy, di chuyển càng nhiều, robot sẽ lấy được nhiều dữ liệu từ cảm biến và cho kết quả định vị chính xác hơn.

Đầu tiên, launch node amcl, sau đó, set up initial pose sử dụng 2D Pose Estimate. Khi cho robot di chuyển, node amcl đọc dữ liệu từ laser topic **/scan**, map topic **/map** và transform topic **/tf** để publish pose đến **/amcl_pose** và **/particlecloud** topic.

Trong file launch amcl, cài đặt lại một số cấu hình để phù hợp với hệ thống:

- <arg name="scan_topic", default="scan"/> → topic của laser scan
- <arg name="odom_frame_id", default="odom"/> → odom frame của hệ thống
- <arg name="base_frame_id", default="base_footprint"/> → base frame của hệ thống
- <arg name="initial_pose_x", default="0.0"/>
- <arg name="initial_pose_y", default="0.0"/>



Hình 4.9: Sơ đồ các node trong amcl

- <arg name="initial_pose_a", default="0.0"/>

Sử dụng **move_base** để điều hướng cho robot. Sau khi robot định vị được vị trí trên bản đồ, command lệnh *rosrun rqt_graph rqt_graph* để kiểm tra luồng dữ liệu của hệ thống (Hình 4.9)

4.2.4 Kết quả sau khi chạy mô phỏng trên Rviz

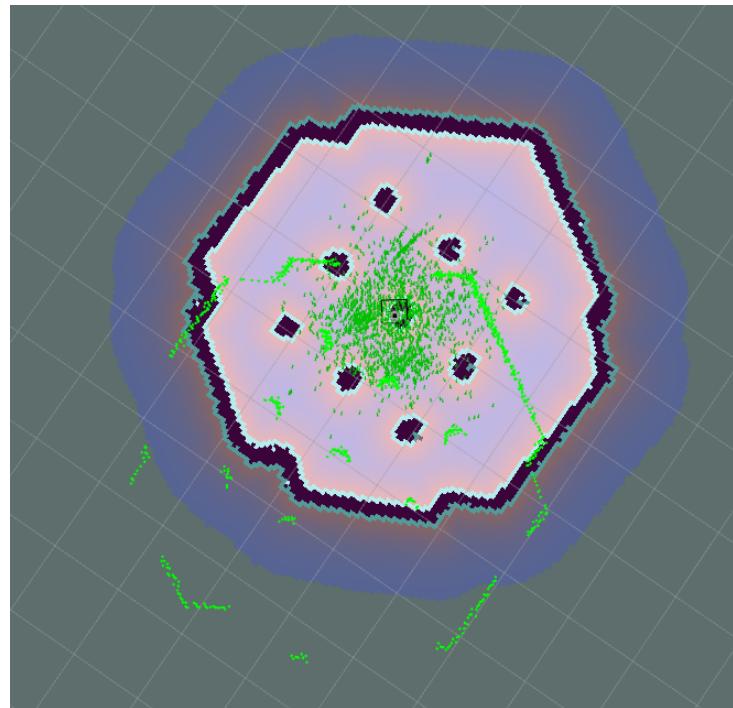
Khi đặt robot vào một vị trí bất kỳ, robot chưa thể định vị được chính xác vị trí đó trên bản đồ (Hình 4.10). Chỉ khi biết chính xác vị trí của mình thì robot mới có thể đưa các cập nhật mới vào bản đồ một cách hợp lý.

Để giúp robot cập nhật được thông tin về môi trường xung quanh, ta phải cho robot di chuyển để lấy được thông tin từ cảm biến, từ đó xác định được đúng vị trí của robot trên bản đồ. Quá trình đó được thể hiện trong Hình 4.11, Hình 4.12 và Hình 4.13.

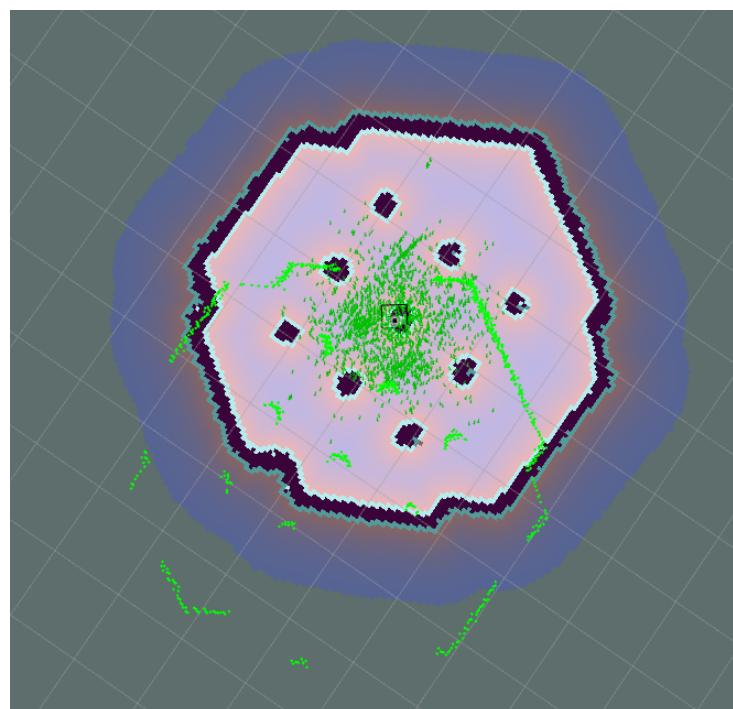
4.2.5 Điều hướng Robot

Giải thuật điều hướng cho Robot được thực hiện sử dụng thuật toán A*, cụ thể được thể hiện trong hình 4.14-4.16

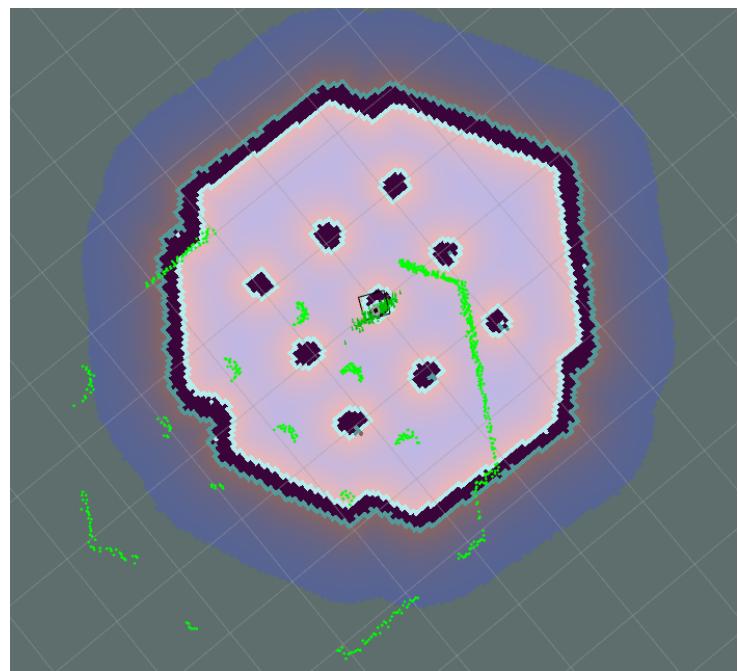
Sau khi đã ước tính vị trí, điều hướng được minh họa bằng cách sử dụng công cụ *2D Nav Goal* để cài đặt vị trí và hướng cho đích mới, thể hiện trong hình 4.14



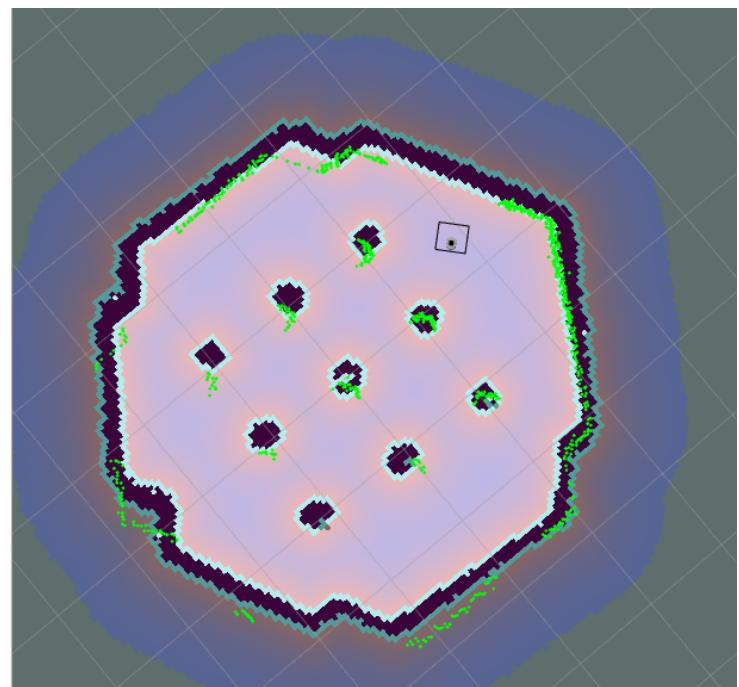
Hình 4.10: Tại thời điểm ban đầu, robot không tương thích với bản đồ



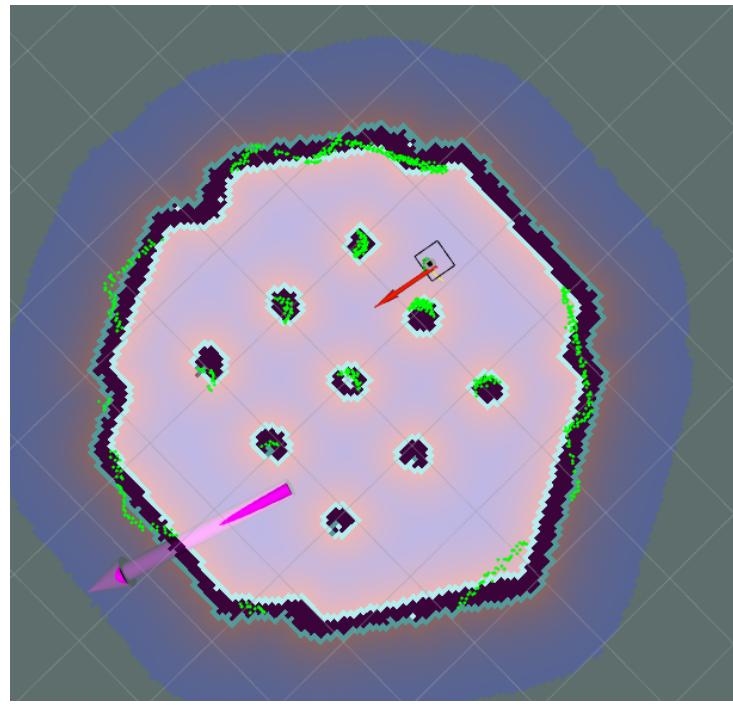
Hình 4.11: Robot di chuyển giúp các hạt tụ lại tại những khu vực gần đúng với vị trí thực của robot



Hình 4.12: Các point cloud dồn tập trung và thu hẹp lại vào những vị trí có khả năng cao là vị trí của robot



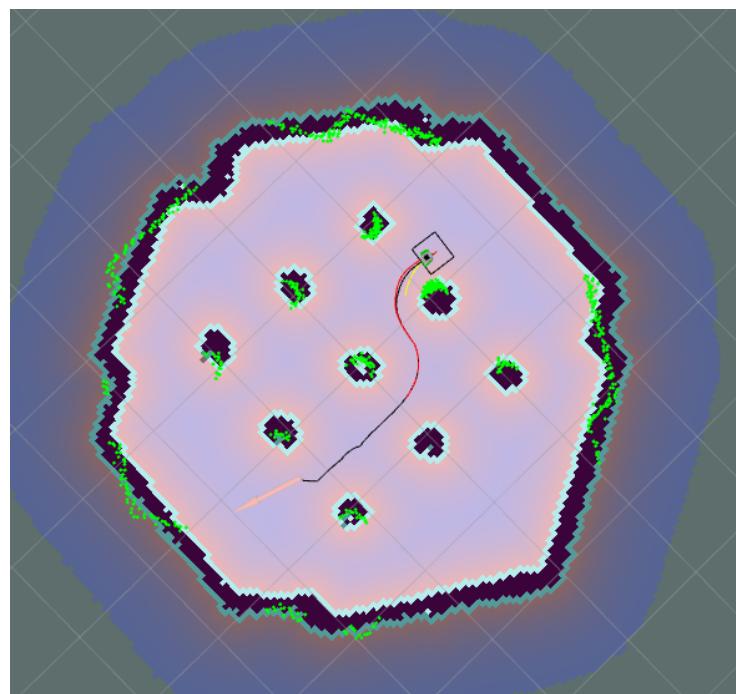
Hình 4.13: Point cloud chỉ đúng vị trí của robot trên bản đồ



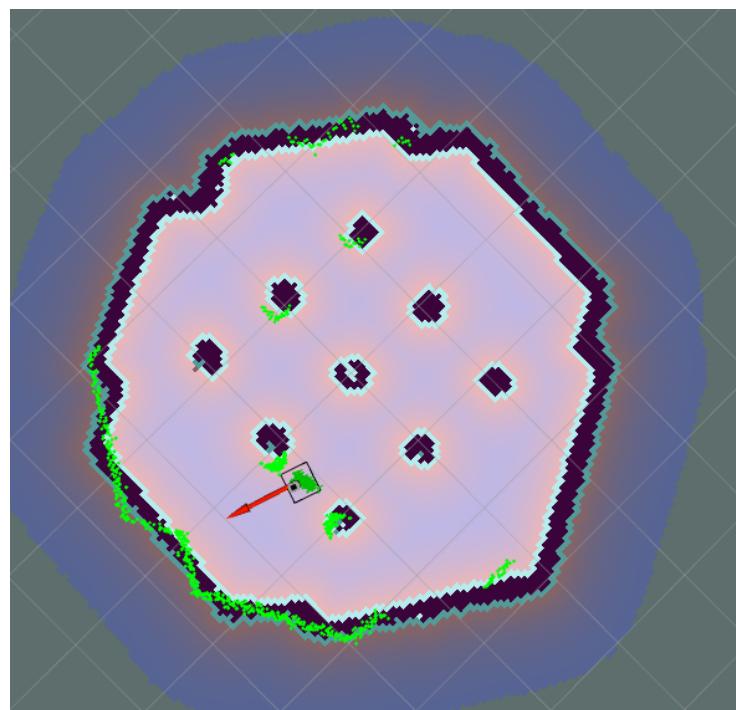
Hình 4.14: Tạo đích mới

Sau khi đã có điểm đích được xác định cả về vị trí và hướng, thuật toán A* lúc này sẽ khởi tạo đường đi, tránh các vật cản, đảm bảo đến đích với hạn chế sự cố va chạm nhất có thể, thể hiện trong hình 4.15

Robot chạm đích với đúng vị trí và hướng đã được xác định từ trước, sẵn sàng cho lệnh điều hướng tiếp theo, thể hiện trong hình 4.16



Hình 4.15: Lập kế hoạch đường đi



Hình 4.16: Chạm đích và chờ lệnh mới

Chương 5

Kết quả và Đánh giá

Đề tài trình bày quá trình:

1. Thực hiện hector SLAM được sử dụng trên thiết bị thật là Raspberry Pi Mouse
2. Thực hiện mô phỏng giải thuật tự động SLAM sử dụng nền tảng lý thuyết RRT và Hector
3. Thực hiện điều hướng sử dụng AMCL để định vị và giải thuật A* lập kế hoạch đường đi

Qua thực nghiệm, nhận xét:

1. Việc sử dụng hector SLAM là lựa chọn tốt hơn khi không có phần cứng thực sự tốt, thay vì chọn Gmapping.
2. Việc sử dụng hết các chương trình trong một tệp launch chạy trên robot là không khuyến khích, tải lượng tính toán nên được dịch qua một phần cứng mạnh mẽ hơn, ví dụ máy tính cá nhân.
3. Việc thực hiện điều hướng thực hoặc RRT trên môi trường thật, cần cài đặt được điểm gốc môi trường, từ đó sử dụng công cụ `/tf` hỗ trợ cho việc chuyển đổi, tính toán, ước tính tư thế.

Nhìn chung có nhiều cách tiếp cận với bài toán SLAM, và ứng dụng SLAM đang được chú ý nhiều. Đề tài tập trung thực hiện trải nghiệm trên sản phẩm thật, thực hành sử dụng các công cụ trong ROS: *Rviz*, *Gazebo*, *rqt_graph*, tiến hành trực quan hóa và phân tích hệ thống. Đề tài mang ý nghĩa giáo dục và có thể dựa trên đó làm prototype phát triển thêm trong tương lai.

Tài liệu tham khảo

- [1] Jos Elfring, Elena Torta, and René van de Molengraft. Particle filters: A hands-on tutorial. *Sensors*, 21(2), 2021.
- [2] Stefan Kohlbrecher, Oskar von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 155–160, 2011.
- [3] Hassan Umari and Shayok Mukhopadhyay. Autonomous robotic exploration based on multiple rapidly-exploring randomized trees. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1396–1402, 2017.
- [4] Yang Wang, Weimin Zhang, Fangxing Li, Yongliang Shi, Zhuo Chen, Fuyu Nie, Chi Zhu, and Qiang Huang. An improved adaptive monte carlo localization algorithm fused with ultra wideband sensor. In *2019 IEEE International Conference on Advanced Robotics and its Social Impacts (ARSO)*, pages 421–426, 2019.