TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

LUẬN VĂN THẠC SĨ

Xác định vị trí cho Robot trong nhà dựa trên thuật toán Scan Matching

PHẠM MỸ HẢO
Hao.pmca170382@sis.hust.edu.vn

Ngành Kỹ thuật máy tính

Giảng viên hướng dẫn: TS. Ngô Lam Trung

Chữ ký của GVHD

Viện: Công nghệ Thông tin và Truyền thông

HÀ NỘI, 06/2020

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập – Tự do – Hạnh phúc

BẢN XÁC NHẬN CHỈNH SỬA LUẬN VĂN THẠC SĨ

Họ và tên tác giả luận văn : Phạm Mỹ Hảo

Đề tài luận văn: Xác định vị trí cho Robot trong nhà dựa trên thuật toán Scan Matching

Chuyên ngành: Kỹ thuật máy tính

Mã số HV: CA170382

Tác giả, Người hướng dẫn khoa học và Hội đồng chấm luận văn xác nhận tác giả đã sửa chữa, bổ sung luận văn theo biên bản họp Hội đồng ngày 20/06/2020 với các nội dung sau:

- Chỉnh sửa các lỗi chính tả trong luận văn
- Bổ sung Danh mục từ viết tắt
- Chỉnh sửa tên tiêu đề các chương

Ngày tháng năm

Giáo viên hướng dẫn **Tác giả luận văn**

CHỦ TỊCH HỘI ĐỒNG

ĐỀ TÀI LUẬN VĂN

Xác định vị trí cho Robot trong nhà dựa trên thuật toán Scan Matching

Giáo viên hướng dẫn
Ký và ghi rõ họ tên

Tóm tắt nội dung luận văn

Trong khi các hệ thống robot cố định người ta thường thiết kế một không gian làm việc ít biến đổi và robot thực hiện các công việc lặp đi lặp lại trong môi trường xác định trước, thì đối với robot di động việc nhận biết được môi trường là một yếu tố quyết định tới các hành động khác. Xác định vị trí robot là việc tìm ra được trạng thái (bao gồm vị trí và định hướng) của robot trong môi trường của nó. Trong đó, có thể là xác định vị trí trong một bản đồ cho trước hoặc tìm vị trí tương đối sau khi di chuyển so với vị trí bắt đầu (trường hợp không biết trước bản đồ).

Đã có nhiều công nghệ được đề xuất để sử dụng cho các hệ thống xác định vị trí trong nhà ví dụ như Vision, Infrared, Wireless Local Area Network (WLAN), RFID, Bluetooth và Laser range finder. Qua so sánh các tiêu chí như: độ chính xác, phạm vi hoạt động, chi phí, độ phức tạp và môi trường, công nghệ Laser range finder có độ phức tạp thấp, chi phí rẻ, độ chính xác chấp nhận được, phù hợp để áp dụng và trở nên phổ biến đối với các thiết bị robot trong nhà.

Bên cạnh việc phát triển các công nghệ thì cũng đã có nhiều nghiên cứu trên thế giới với các thuật giải và phương pháp khác nhau để xác định vị trí của robot trong đó 04 nhóm giải pháp chính để định vị là phương pháp dẫn đường dự đoán (dead-reckoning), hệ thống dẫn đường cột mốc chủ động, hệ thống dẫn đường cột mốc thụ động, định vị sử dụng bản đồ cục bộ. Thuật toán quét và so khớp (Scan Matching) thuộc nhóm giải pháp sử dụng bản đồ cục bộ, có ưu điểm có thể áp dụng linh động không cần thiết lập trước các cột mốc, không bị ảnh hưởng bởi các sai số từ môi trường như gió, vật cản, bề mặt di chuyển không bằng phẳng hay sai lệch giữa thiết kế và thực tế của động cơ, sai lệch giữa các động cơ trong cùng một robot, …

Thuật toán ScanMatching có nhiều hướng tiếp cận trong đó nổi bật 2 hướng chính để tìm ra sự so khớp tương ứng, đó là thuật toán Lặp tương ứng kép (Iterative Dual Correspondence - IDC) và Điểm lặp gần nhất (Iterative Closest Point – ICP). Nếu như thuật toán ICP đưa ra các dự đoán về phép biến đổi để so khớp dữ liệu dựa trên tính toán giữa điểm tới điểm, thì thuật toán biến thể PL ICP (ICP with point-to-line metric) dựa trên tính toán từ Điểm tới đường thẳng. Qua

thử nghiệm thực tế, thuật toán PL ICP thực hiện ít vòng lặp, thời gian xử lý trung bình thấp cũng như độ chính xác vượt trội so với các thuật toán còn lại.

Để thực hiện đề tài này, ngoài việc nghiên cứu về các công nghệ, phương pháp cũng như chi tiết thuật toán Scan Matching, để có thể triển khai thực tế tôi đã tìm hiểu về hệ điều hành Robot ROS (Robot Operating System), cảm biến laser Hokuyo UTM-30LX và robot Kobuki. Đặc điểm nổi bật của ROS chính là xây dựng ứng dụng robotic trên nền tảng ROS sẽ giảm đi một lượng đáng kể các công việc lập trình, thiết lập hệ thống, tận dụng nguồn tài nguyên mã nguồn mở vô cùng phong phú của cộng đồng mà đa số đến từ những viện nghiên cứu và những trường đại học hàng đầu trên thế giới.

Trong đề tài này, tôi đã thực hiện triển khai thực tế thuật toán Scan Matching PL ICP trên hệ điều hành ROS sử dụng đế robot Kobuki để di chuyển, cảm biến laser Hokuyo UTM-30LX thu thập dữ liệu và board Nvidia Jetson TX1 Developer Kit được lắp tại tầng 1 của Kobuki để thu thập, trao đổi dữ liệu với chương trình chính chạy trên laptop thông qua mạng LAN. Sau đó, tôi thực hiện các thí nghiệm kiểm chứng với phần cứng đã được mô tả, trong điều kiện môi trường tiết diện thẳng, tiết diện gồ ghề, hành lang dài, môi trường trong nhà có phòng thông nhau thu được kết quả với sai số trung bình về vị trí là 42,6mm độ lệch chuẩn 12.7 mm và sai số về góc là $0,6^0$ độ lệch chuẩn $0,4^0$ trong thời gian thực. Với kết quả này, hoàn toàn có thể xây dựng một hệ thống robot trong nhà để thực hiện các chức năng như mang đồ vật, hút bụi, giám sát an ninh tòa nhà, robot di chuyển và hoạt động trong các môi trường con người không thể tiếp cận như khu vực cách ly, kho lạnh, hầm lò, …

HỌC VIÊN
Ký và ghi rõ họ tên

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG

DANH MỤC TỪ VIẾT TẮT

ROS	Ros Operating System	Hệ điều hành Robot
Wi-Fi	Wireless Fidelity	Hệ thống truy cập Internet không dây
WLAN	Wireless Local Area Network	Mạng cục bộ không dây
RFID	Radio Frequency Identification	Nhận dạng qua tần số vô tuyến
Imu	Inertial measurement unit	Đơn vị đo lường quán tính
IDC	Iterative Dual Correspindence	Thuật toán lặp tương ứng kép
ICP	Iterative Closest Points	Thuật toán điểm lặp gần nhất
PL-ICP	ICP with Point-to-Line metric	Thuật toán cải tiến Điểm tới Đường thẳng của Điểm lặp gần nhất

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

1.1 Lí do chọn đề tài

Kể từ khi máy tính cá nhân Personal Computer ra đời (những năm 1980) đến nay, chúng ta đã chứng kiến sự phát triển thần tốc của công nghệ thông tin, thâm nhập vào tất cả mọi ngóc nghách của cuộc sống, làm thay đổi bộ mặt của thế giới trong một thời gian ngắn. Tuy nhiên, cho đến bây giờ PC không còn là trung tâm của cuộc cách mạng công nghệ mà thay bằng những sản phẩm đột phá mới đáp ứng nhu cầu của con người. Một trong những sản phẩm tiềm năng thời đại hậu PC là Robot dịch vụ đặc biệt là Robot cá nhân phục vụ nhu cầu hàng ngày của con người. Khi những con robot này trở thành nhu cầu không thể thiếu của từng cá nhân, thâm nhập vào từng gia đình thì chúng sẽ tạo nên một thị trường khổng lồ làm thay đổi xã hội loài người như tác động của PC đã làm thay đổi xã hội thời gian qua.

Việc hoạt động ở môi trường trong nhà đặc biệt là hoạt động trong gia đình gặp nhiều trở ngại cho các robot cá nhân do môi trường có diện tích nhỏ và nhiều chướng ngại vật. Do đó việc xác định được vị trí chính xác của robot so với vị trí tương đối của các đồ vật trong nhà có ý nghĩa lớn trong việc phổ biến robot gia đình. Đề tài **Xác định vị trí cho Robot trong nhà dựa trên thuật toán Scan Matching** bằng cảm biến laser được ra đời để giải quyết các vấn đề trên. Các mục tiêu được đặt ra bao gồm: sử dụng hệ điều hành ROS (Robot Operating System) thu nhận dữ liệu từ cảm biến laser, sử dụng thuật toán Scan Matching để xác định vị trí tương đối của Robot so với vị trí gốc khi robot di chuyển.

Kết quả của đề tài sẽ được sử dụng để phát triển robot dịch vụ có khả năng di chuyển trong nhà dựa trên vị trí tương đối của robot và các đồ vật, ứng dụng trong robot hút bụi, robot dẫn đường, robot vận chuyển hàng ….

1.2 Mục đích nghiên cứu

Mục đích nghiên cứu của luận văn này là xây dựng và thử nghiệm thực tế thuật toán xác định vị trí của Robot dựa trên thuật toán Scan Matching, triển khai thử nghiệm thực tế ở điều kiện môi trường trong nhà. Cụ thể luận văn này có hai mục tiêu:

- Trước tiên, tìm hiểu về công nghệ và phương pháp định vị robot cũng như thuật toán Scan Matching.
- Sau đó, triển khai thực tế thuật toán đã đề xuất trên cảm biến laser, robot Kobuki và đánh giá kết quả đối với điều kiện trong nhà.

1.3 Đối tượng và phạm vi nghiên cứu

- Đối tượng nghiên cứu: Thuật toán Scan Matching để xác định vị trí của robot trong nhà và thử nghiệm thực tế
- Phạm vi nghiên cứu:
  - Nghiên cứu thuật toán xác định vị trí của robot trong nhà Scan Matching
  - Nghiên cứu và ứng dụng thuật toán này trên hệ điều hành Robot sử dụng cảm biến laser và robot Kobuki
  - Thử nghiệm và đánh giá phương pháp.

1.4 Phương pháp nghiên cứu

- Nghiên cứu các công nghệ xác định vị trí robot trong nhà để xác định ưu điểm và hạn chế của các công nghệ này.
- Nghiên cứu các thuật giải và phương pháp xác định vị trí
- Nghiên cứu thuật toán Scan Matching với một số phương pháp tiếp cận
- Nghiên cứu hệ điều hành robot ROS và các package liên quan
- Triển khai thực tế thuật toán Scan Matching PL-ICP trên cảm biến laser và robot Kobuki
- Thực hiện các thí nghiệm và đánh giá

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT XÁC ĐỊNH VỊ TRÍ

2.1 Công nghệ xác định vị trí

Trong nhiều năm qua, có nhiều công nghệ được đề xuất sử dụng cho các hệ thống định vị trong nhà. Các công nghệ phổ biến hiện nay gồm có Vision, Infrared, Ultrasound, Wireless Local Area Network (WLAN), RFID, Bluetooth, Sonar và Laser range finder.

2.1.1 Vision

Phương pháp này dựa trên việc xử lý và đánh giá dữ liệu video. Cụ thể thì việc định vị dựa vào video có thể được thực hiện bằng hai cách khác nhau:

- Hệ thống camera cố định (Fixed camera systems): Các camera được lặp đặt tại các vị trí cố định trong môi trường. Trong trường hợp này, mục tiêu là nhằm xác định đối tượng di động trong ảnh (cá nhân, đối tượng hoặc động vật) được thu lại bởi một hoặc nhiều camera và theo dõi các đặc trưng của chủ thể. Nếu đặc tính quan trọng nhất của chủ thể xuất hiện trong khung nhìn của camera, vị trí của nó sẽ được tính toán từ vị trí của các camera có liên quan. Vị trí của đối tượng được tính toán dựa vào vị trí của nó ở trong ảnh và sự phân bố các đặc trưng của nó trong không gian.
- Hệ thống camera di động (Mobile camera systems): Đối tượng di động được trang bị một camera và việc xác định vị trí được thực hiện bằng cách đặt các điểm mốc tại các vị trí biết trước (cả về hướng) hoặc bằng cách trích xuất các đặc trưng của môi trường xung quanh. Trong trường hợp đầu tiên, nếu mobile camera phát hiện hai hoặc nhiều điểm mốc, nó có thể nhận ra vị trí và hướng của nó. Trong trường hợp còn lại, quá trình xác định vị trí thông qua hai giai đoạn. Trong giai đoạn off-line, các ảnh của môi trường được chụp tại các vị trí được xác định trước để trích xuất các đặc trưng riêng và được lưu vào cơ sở dữ liệu. Trong giai đoạn on-line, camera sẽ chụp một ảnh, các đặc trưng của nó được trích xuất và so sánh với các đặc trưng được lưu trữ để đánh giá vị trí của camera. Trong cả hai trường hợp, thì mục đích đều nhằm đánh giá vị trí và hướng của mobile camera.

Ngày nay, độ chính xác của các hệ thống định vị trong nhà bằng camera đạt tới 10^{-3}m – 10^{-1}m (đối với các hệ thống có độ chính xác cao). Ngoài ra, việc tăng tốc độ truyền dữ liệu và tăng khả năng tính toán, cùng với sự phát triển của thuật toán xử lý ảnh hiệu năng cao khiến cho công nghệ này trở nên cực kỳ hiệu quả. Một nhược điểm của công nghệ này là chi phí bỏ ra khá cao, nhưng nhờ các công nghệ mới, các giải pháp chi phí thấp đang ngày càng phát triển và xu thế đang hướng đến các hệ thống định vị sử dụng camera của điện thoại di động.

2.1.2 Infrared

Sóng hồng ngoại (Infrared radiation – IR) là một trong những công nghệ phổ biến nhất trong các công nghệ không dây được sử dụng để xác định vị trí đồ vật hoặc con người thông qua các bộ thu phát sóng hồng ngoại.

Một ví dụ của hệ thống định vị trong nhà sử dụng IR có thể thấy như là người dùng mang một cái túi xách, trong đó có chứa mã định danh (ID), nó phát ra các tín hiệu IR tại những khoảng thời gian đều nhau thông qua một bộ truyền tín hiệu hồng ngoại. Các bộ nhận tín hiệu được đặt tại những vị trí biết trước trong môi trường xung quanh, sau khi xác định được ID nó sẽ gửi tới một phần mềm xác định vị trí để tính toán vị trí của chiếc túi xách dựa vào khoảng cách giữa bộ phát và bộ thu tín hiệu.

Công nghệ này có một vài ưu điểm. Đầu tiên, chùm tín hiệu IR không thể xuyên tường cho nên chỉ có thể thu được các tín hiệu xuất phát từ bên trong phòng. Ngoài ra, công nghệ IR không bị ảnh hưởng bởi nhiễu của sóng điện từ và năng lượng của tín hiệu IR có thể được hiệu chỉnh dễ dàng cho phù hợp với từng khu vực cụ thể. Tuy nhiên, cũng có một vài nhược điểm. Các lỗi gây ra bởi hiện tượng đa đường multipath làm giảm nghiêm trọng độ chính xác của việc định vị. Thêm đó các hệ thống định vị trong nhà sử dụng IR thì có các thiết bị phần cứng đắt đỏ và tốn nhiều chi phí bảo trì. Hơn nữa, công nghệ IR yêu cầu một đường truyền thẳng (Line of Sight – LoS) giữa bộ thu và bộ phát để có thể hoạt động được một cách chính xác.

2.1.3 WLAN

Wireless Local Area Network (WLAN) có thể được sử dụng để tính toán vị trí của thiết bị di động trong mạng. Để tăng nhu cầu giao tiếp không dây nên trang

thiết bị của WLAN được trải rộng khắp trong các môi trường trong nhà. Vì lý do này, một trong những ưu điểm của kỹ thuật định vị sử dụng Wi-Fi là tính hiệu quả về chi phí do tính khả thi khi xác định vị trí của gần như tất cả các thiết bị có kết nối Wi-Fi mà không cần phải cài đặt thêm bất kỳ phần mềm nào. Một ưu điểm khác của việc sử dụng WLAN là không cần yêu cầu LoS (Line of Sight – truyền theo đường thẳng)

Trong mạng WLAN, một node thu/phát các tín hiệu RF từ/tới một router không dây, router này có thể được dùng để xác định chính xác vị trí của bất kỳ thiết bị Wi-Fi nào đã được kích hoạt. Phương pháp WLAN RSS có thể sử dụng ba cách cơ bản để xác định vị trí của thiết bị đích:

- Phương pháp Cell of Origin (CoO): hoạt động dựa vào việc biết vị trí của access point (AP), nơi mà các thiết bị được kết nối tới.
- Phương pháp Triangulation: vị trí của thiết bị mục tiêu được tính toán bằng việc lập lưới tam giác các thông tin cường độ tín hiệu nhận được tại các vị trí thu.
- Phương pháp Fingerprint: là phương pháp khả thi nhất dành cho định vị trong nhà sử dụng RSS và hoạt động bằng cách lập bản đồ cường độ tín hiệu quan sát được của các router cố định đặt ở môi trường trong nhà và được lưu vào một cơ sở dữ liệu (ví dụ: radio map). Thiết kế cơ bản của phương pháp này có thể được chia thành hai giai đoạn là offline và online. Trong giai đoạn offline, RSS được thu thập tại các vị trí lấy mẫu để dựng bản đồ radio đối với môi trường cụ thể. Trong giai đoạn online, vị trí vật lý của client có thể được tính toán bằng cách so sánh RSS đo được với các giá trị RSS đã được lưu trữ.

Độ chính xác của công nghệ Wi-Fi trong khoảng 20m cho đến 40m, nhưng nó có thể được cải thiện bằng các lắp đặt thêm nhiều các bộ định tuyến không dây (wireless routers) hoặc tích hợp thêm nhiều công nghệ khác, và những kết quả gần đây cho thấy rằng độ chính xác đạt được vào khoảng 3–5m [1].

Ngoài ra, có nhiều vấn đề mang tính thách thức trong công nghệ định vị sử dụng WLAN cũng rất quan trọng. Một trong số đó là vấn đề tiêu thụ năng lượng. Trên thực tế, do các thiết bị di động thường nhỏ và có những hạn chế về năng lượng,

một vấn đề khó khăn là làm sao để giảm thiểu điện năng tiêu thụ trong quá trình định vị. Một giới hạn khác của WLAN là sự hấp thụ tín hiệu bởi môi trường tĩnh như tường, sự di chuyển của đồ đạc và các cánh cửa.

2.1.4 RFID

Công nghệ RFID sử dụng một bộ đọc RFID được trang bị một hoặc nhiều antenna đọc và các bộ thu phát active hoặc passive. Các thẻ RFID active có một viên pin và có thể tự động truyền tín hiệu, trong khi đó các thẻ RFID passive thì không có pin và cần phải có nguồn ngoài để truyền tín hiệu. Thông thường, dữ liệu trong thẻ chỉ chứa một chuỗi số duy nhất, nhưng cũng có thể có thêm các thông tin khác (như thông tin vị trí) được lưu trữ trong đó. Lượng dữ liệu có thể được lưu trữ phụ thuộc vào kích thước bộ nhớ. Những đặc trưng của công nghệ này khiến nó trở thành ứng viên lý tưởng cho việc truy nguồn gốc của các sản phẩm [2] trong chuỗi cung ứng, nó cũng được sử dụng cho nhiều mục đích khác, bao gồm cả việc xác định vị trí trong nhà.

Việc xác định vị trí sử dụng RFID có thể được phân thành hai loại là xác định vị trí bộ đọc, và xác định vị trí thẻ (tag). Trong việc định vị bộ đọc, độ chính xác của hệ thống RFID phụ thuộc phần lớn vào mật độ của việc lắp đặt thẻ và phạm vi tối đa để có thể đọc được. Trong trường hợp khả thi, cần số lượng lớn thẻ RFID có chứa các thông tin về vị trí, có thể được lắp đặt để phủ khắp môi trường trong nhà. Một người cầm bộ đọc trên tay có thể đọc thông tin về vị trí của mình từ thẻ gần với bộ đọc nhất. Nhược điểm của phương pháp này là cần sử dụng một số lượng lớn thẻ RFID và phải chứa thông tin vị trí của thẻ. Ngược lại, thì việc xác định vị trí thẻ lại yêu cầu nhiều bộ đọc RFID trải rộng trong môi trường tại các vị trí được xác định trước. Có thể thấy rằng, phương pháp này tốn kém hơn và chi phí sẽ tăng mỗi khi cần tăng thêm các bộ đọc RFID.

Công nghệ RFID hoạt động mà không cần LoS bởi vì sóng radio có thể đâm xuyên qua các vật liệu rắn, nhưng cường độ của tín hiệu thì phụ thuộc vào mật độ của các đồ vật trong nhà, do vậy độ chính xác thường bị giới hạn. Thông thường phạm vi tần số được sử dụng trong RFID được phân loại như sau: Low Frequency (LF) tại 125–134 kHz; High Frequency (HF) tại 13.56 MHz; Ultra-High frequencies (UHF) tại 860–960 MHz. Ngoài khả năng hoạt động trong môi trường NLoS, các ưu điểm khác của công nghệ RFID là tốc độ truyền dữ liệu

cao, độ bảo mật cao, hiệu quả về chi phí và sự chắc chắn. Những hạn chế chính của công nghệ RFID sử dụng LF và HF có liên quan tới phạm vi đọc thẻ ngắn và chỉ có thể đọc được một ít thẻ cùng một lúc. Với công nghệ RFID sử dụng UHF, các nhược điểm của nó chủ yếu do sự hấp thụ hoặc sự phản xạ của sóng RF trên bề mặt chất lỏng hoặc kim loại.

2.1.5 Bluetooth

Bluetooth là một chuẩn không dây dành cho Wireless Personal Area Networks (WPANs) và hoạt động ở tần số 2.4 GHz. So với WLAN thì phạm vi tín hiệu ngắn hơn (10–15m). Mặt khác, Bluetooth được coi là một chuẩn "lighter", có mặt khắp mọi nơi bởi vì nó được nhúng trong hầu hết các thiết bị như là điện thoại di động, điện thoại cầm tay cá nhân (PDAs), máy tính xách tay, máy tính để bàn, v.v… Việc sử dụng công nghệ Bluetooth trong việc ghi lại vị trí cho phép tái sử dụng các thiết bị có tích hợp công nghệ Bluetooth, vì vậy việc thêm một người dùng mới vào trong hệ thống thì không cần phải cung cấp thêm bất kỳ thiết bị phần cứng nào nữa. Do Bluetooth là công nghệ chi phí thấp và năng lượng tiêu thụ thấp, nên nó rất hiệu quả để xây dựng các hệ thống định vị trong nhà. Ngoài ra, các thẻ Bluetooth là các bộ thu phát có kích thước nhỏ gọn. Giống như bất kỳ thiết bị Bluetooth nào khác, mỗi thẻ có một ID duy nhất, để có thể sử dụng cho việc xác định thẻ Bluetooth.

Cuối cùng, một công nghệ của Apple được gọi là iBeacons, sử dụng công nghệ không dây Bluetooth Low Energy (BLE) để cung cấp vị trí dựa trên các thông tin và các dịch vụ tới iPhones và các thiết bị iOS khác. Việc xác định vị trí trong nhà thông qua iBeacon cung cấp nhiều lợi ích đáng kể trong việc xây dựng các khu phức hợp mà không có một hạ tầng không dây cố định. Vị trí trong tòa nhà có thể được xác định bằng cách sử dụng iBeacon kết hợp với các giải pháp phần mềm. Sử dụng công nghệ BLE cho phép sử dụng pin trong nhiều tháng mà không cần phải dùng đến nguồn cấp ngoài.

Một trong những nhược điểm của công nghệ Bletooth trong định vị là mỗi khi tìm vị trí, nó phải chạy thủ tục để phát hiện các thiết bị; điều này làm tăng đáng kể thời gian chờ trong quá trình xác định vị trí (10–30s) và cũng tiêu tốn nhiều năng lượng. Vì lý do này, các thiết bị Bluetooth có thời gian chờ không phù hợp với các ứng dụng xác định vị trí thời gian thực. Một hạn chế khác của hệ thống

định vị sử dụng Bluetooth là nó chỉ có thể cung cấp độ chính xác trong khoảng 2–3m với độ trễ vào khoảng 20s. Ngoài ra, các hệ thống định vị sử dụng Bluetooth gặp phải các nhược điểm của kỹ thuật định vị sử dụng RF trong nhà tại các trường hợp phức tạp và thay đổi.

2.1.6 Laser Range Finder

Cảm biến đo khoảng cách bằng laser hiện nay được sử dụng rất phổ biến trong lĩnh vực đo lường, tự động hóa. Đặc điểm của tia laser là có tính định hướng (mức độ tập trung ánh sáng) tốt, tính đơn sắc tốt và độ chói cao, tính tương hợp (corelation) tốt.

Cảm biến đo khoảng cách bằng laser hay hoạt động dựa trên nguyên lý phản xạ của tia laser: cảm biến sẽ phát ra tia laser đồng thời có bộ phận thu nhận lại phản xạ khi tia laser gặp vật cản, từ đó tính toán được khoảng cách từ vị trí cảm biến đến vật cản. Cảm biến có ưu điểm có thể mang lại kết quả đo "siêu tốc" trong phạm vi từ 0,05 – 25 mét chỉ trong khoảng 0,5 giây, tối đa không quá 04s với độ chính xác cao, sai số tương đối nhỏ.

Dựa trên giá trị góc và khoảng cách giữa vật cản và cảm biến, có nhiều thuật toán xác định vị trí và dẫn đường cho robot theo bản đồ sử dụng laser range finder ở môi trường trong nhà.

2.1.7 Các công nghệ khác

Có nhiều công nghệ có thể thực hiện việc xác định vị trí ở trong nhà. Trong số các công nghệ đáng nhắc đến có ZigBee, FM radio và các công nghệ sử dụng cảm biến quán tính.

Định vị trong nhà sử dụng công nghệ ZigBee dựa trên việc hình thành một mạng ZigBee chứa nhiều nốt cảm biến tham chiếu tại các vị trí được biết trước và một nốt mục tiêu, hay còn được gọi là nốt chìm (sink node), thông tin vị trí của nốt này là chưa biết. Do các nốt có thể giao tiếp với nhau, cường độ tín hiệu được nhận bởi các cảm biến tham chiếu sẽ được sử dụng để tính toán vị trí. Trong mạng tồn tại các thuật toán định vị khác nhau dẫn đến nhận được các kết quả khác nhau và đạt được hiệu năng không giống nhau trong các môi trường thực nghiệm khác nhau [3].

Công nghệ sóng FM được sử dụng trong [4] để giải quyết các hạn chế của công nghệ Wi-Fi. Cụ thể, tác giả đề xuất sử dụng các tín hiệu radio quảng bá FM để tăng cường hoặc thậm chí thay thế các tín hiệu Wi-Fi.

Khi nói đến cảm biến quán tính tức là đề cập đến các cảm biến sử dụng quán tính để đo gia tốc tuyến tính hoặc vận tốc góc. Về vấn đề này, hệ thống điều hướng quán tính (Inertial Navigation System – INS) là một sự trợ giúp điều hướng sử dụng các cảm biến chuyển động và các cảm biến quay để tính toán liên tục vị trí, hướng và vận tốc của đối tượng đang di chuyển mà không cần thêm các tham chiếu nào khác [5]. Các điện thoại thế hệ mới được trang bị các cảm biến quán tính (accelerometers và gyroscopes) và nó góp phần làm cho các hệ thống định vị trong nhà với chi phí thấp trở nên khả thi hơn [6].

2.1.8 So sánh các công nghệ

Để có thể chọn được một công nghệ phù hợp (hoặc một sự kết hợp của các công nghệ) để xây dựng và cài đặt thành một hệ thống định vị trong nhà, việc so sánh các công nghệ là điều cần thiết.

Trong bảng sau, một vài tham số được chọn làm tiêu chí để so sánh như: độ chính xác, phạm vi hoạt động, chi phí thiết bị (bao gồm cả cảm biến và phần cứng xử lý tính toán), độ phức tạp và môi trường hoạt động. Giá trị của các tham số này hoàn toàn chỉ mang tính tượng trưng bởi vì các giá trị thực tế thì phụ thuộc vào nhiều yếu tố, cho nên chúng được ước lượng trong từng trường hợp khác nhau.

Bảng 2.1.1 Bảng so sánh các công nghệ định vị trong nhà [7]

		Parameters				
		Accuracy (m)	***Coverage (m)***	***Cost***	***Complexity***	***Typical Environment***
Technologies	***Vision***	10^{-3} – 10^{-1}	1 – 10	High	High	Indoor
	Infrared	10^{-2} – 1	1 – 5	Medium/High	Low	Indoor
	WLAN	1 – 10	20 – 50	Medium/Low	Low	Indoor/ Outdoor
	RFID	10^{-1} – 1	1 – 10	Low	Low	Indoor
	Bluetooth	1 – 10	1 – 30	Low	Low	Indoor/ Outdoor
	Laser range finder	10^{-2} – 10^{-1}	10^{-2} – 4	Low	Low	Indoor

Qua so sánh có thể thấy đối với môi trường trong nhà, công nghệ Laser range finder có độ phức tạp thấp, chi phí rẻ, độ chính xác chấp nhận được. Cũng chính vì những lí do này, công nghệ Laser range finder trở thành lựa chọn hàng đầu trong việc xác định vị trí cho các thiết bị robot trong nhà như robot hút bụi, robot cá nhân, ….

2.2 Phương pháp xác định vị trí

Hệ thống robot cố định truyền thống vả robot di động có một sự khác biệt quan trọng đó là robot di động không biết trước môi trường vận hành. Trong khi các hệ thống robot cố định người ta thường thiết kế một không gian làm việc cố định và robot thực hiện các công việc lặp đi lặp lại trong môi trường xác định trước, thì đối với robot di động việc nhận biết được môi trường là một yếu tố quyết định tới các hành động khác. Xác định vị trí robot là việc tìm ra được trạng thái (bao gồm vị trí và định hướng) của robot trong môi trường của nó. Trong đó, có thể là xác định vị trí trong một bản đồ cho trước hoặc tìm vị trí tương đối sau khi di chuyển so với vị trí bắt đầu (trường hợp không biết trước bản đồ).

Có nhiều nghiên cứu trên thế giới với các thuật giải và phương pháp khác nhau để xác định vị trí của robot trong đó 04 nhóm giải pháp chính:

2.2.1 Phương pháp dẫn đường dự đoán dead-reckoning

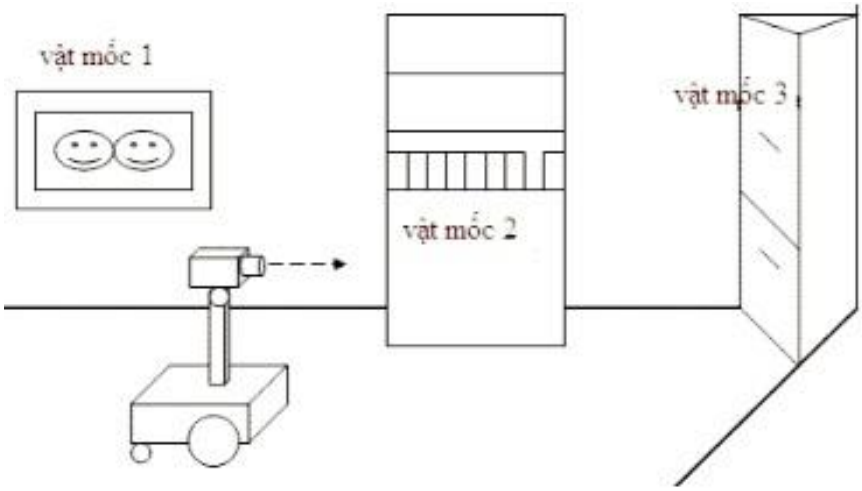
Dead-reckoning là phương pháp dẫn đường được sử dụng rộng rãi nhất đối với robot di động. Phương pháp này cho độ chính xác trong thời gian ngắn, giá thành thấp và tốc độ lấy mẫu rất cao. Tuy nhiên do nguyên tắc cơ bản của phương pháp dead-reckoning là tích luỹ thông tin về gia tốc chuyển động theo thời gian do đó dẫn tới sự tích luỹ sai số. Sự tích luỹ sai số theo hướng sẽ dẫn đến sai số vị trí lớn tăng tỉ lệ với khoảng cách chuyển động của robot.

Phương pháp dead-reakoning dựa trên phương trình đơn giản và thực hiện được một cách dễ dàng, sử dụng dữ liệu từ bộ mã hoá số vòng quay bánh xe. Dead-reckoning dựa trên nguyên tắc là chuyển đổi số vòng quay bánh xe thành độ dịch tuyến tính tương ứng của robot. Nguyên tắc này chỉ đúng với giá trị giới hạn. Có một vài lý do dẫn đến sự không chính xác trong việc chuyển từ số gia vòng quay bánh xe sang chuyển động tuyến tính. Tất cả các nguồn sai số này được chia thành 2 nhóm: sai số hệ thống và sai số không hệ thống. Để giảm sai số dead-reckoning cần phải tăng độ chính xác động học cũng như kích thước tới hạn.

2.2.2 Hệ thống dẫn đường cột mốc chủ động.

Hệ thống dẫn đường cột mốc chủ động cung cấp thông tin vị trí rất chính xác với quá trình xử lý tối thiểu. Hệ thống cho phép tốc độ lấy mẫu và độ tin cậy cao nhưng đi kèm với nó là giá thành cao trong việc thiết lập và duy trì. Cột mốc được đặt tại các vị trí chính xác sẽ cho phép xác định toạ độ chính xác của vật thể.

Phương pháp đo dùng trong hệ thống cột mốc chủ động là phép đo 3 cạnh tam giác hoặc phép đo 3 góc tam giác. Phép đo xác định vị trí vật thể dựa trên khoảng cách hoặc góc đo được tới cột mốc biết trước. Trong hệ thống dẫn đường sử dụng phép đo này thông thường có ít nhất là 3 trạm phát đặt tại các vị trí biết trước ngoài môi trường và 1 trạm nhận đặt trên robot. Hoặc ngược lại có 1 trạm phát đặt trên robot và các trạm nhận đặt ngoài môi trường. Sử dụng thông tin về thời gian truyền của chùm tia hệ thống sẽ tính toán khoảng cách hoặc góc giữa các trạm phát cố định và trạm nhận đặt trên robot để từ đó xác định vị trí của robot



Hình 2.2.1 Định vị sử dụng cột mốc

2.2.3 Hệ thống dẫn đường cột mốc thụ động.

Tương tự như phần trên nhưng đây là vật mốc nhân tạo và tự nhiên, đã xuất hiện sẵn trên bản đồ. Sử dụng các công nghệ nhận diện để đánh dấu đây là các cột mốc.

2.2.4 Định vị sử dụng bản đồ cục bộ

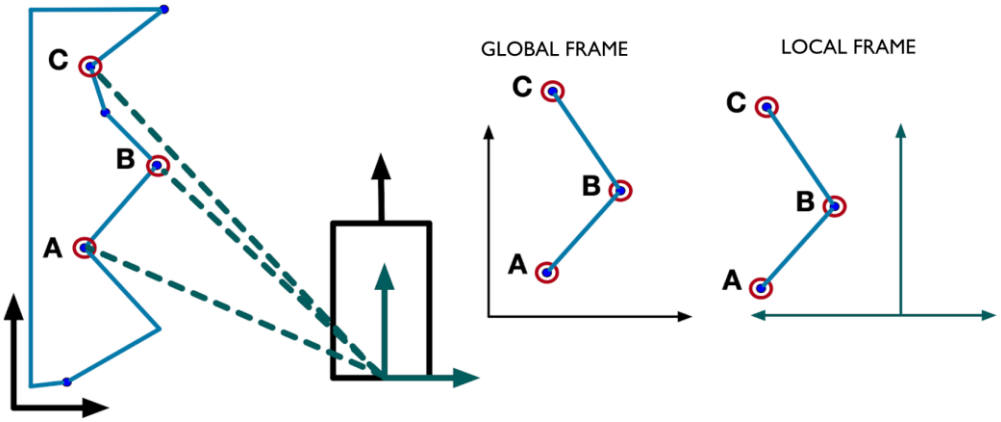
Robot sử dụng các cảm biến được trang bị để tạo ra một bản đồ cục bộ môi trường xung quanh. Dựa trên sự thay đổi bản đồ theo thời gian do sự di chuyển của robot, từ đó tính toán được vị trí thực tế của nó trong môi trường. Có nhiều thuật toán để xác định vị trí robot dựa trên bản đồ cục bộ, trong đó Thuật toán quét và so khớp (Scan Matching) là một trong những thuật toán cơ bản và phổ biến nhất

Hình 2.2.2 Bản đồ Phòng nghiên cứu Intel với dữ liệu cảm biến thô (bên trái) và sau khi Scan Matching dữ liệu đó (bên phải) [8]

2.3 Thuật toán Scan Matching

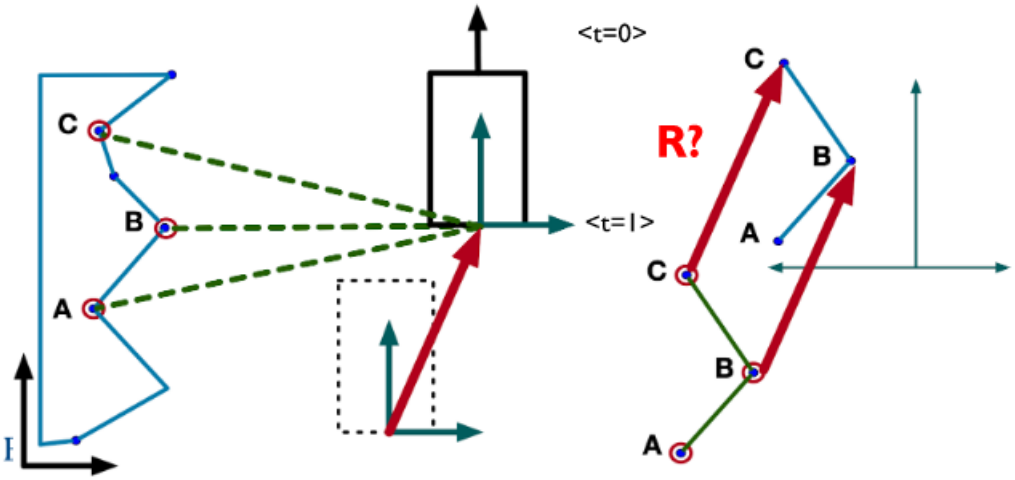
Đặt vấn đề:

Robot đang ở trong môi trường nào đó với 3 điểm mốc A, B, C. Tại thời điểm t=0, ta đo được khoảng cách từ robot đến từng điểm A, B, C và thể hiện trên tọa độ toàn bộ và tọa độ cục bộ tương ứng như sau:



Hình 2.3.1 Minh họa khoảng cách từ robot đến các điểm mốc trên hệ tọa độ

Tại thời điểm t=1, robot di chuyển một khoảng chưa rõ, khi này khoảng cách từ robot đến các điểm đánh dấu bị thay đổi. Chúng ta cần tìm một phép biến đổi R mà biến đổi 2 tập hợp điểm là gần nhất



Hình 2.3.2 Minh họa phép biến đổi giữa 2 lần di chuyển của robot so với các điểm mốc trên hệ tọa độ

Thách thức:

Nếu chúng ta biết chính xác các điểm mốc A, B, C thì việc tìm R là đơn giản. Tuy nhiên chúng ta lại không thể thực hiện các phép đo chính xác các điểm mốc này (hoặc xác định chính xác đặc điểm các cột mốc này).

Hướng tiếp cận:

Có nhiều hướng tiếp cận đối với thuật toán Quét và So khớp Scan Matching, tuy nhiên có 2 hướng tiếp cận chính để tìm ra sự so khớp tương ứng, cụ thể được giới thiệu dưới đây:

2.3.1 Thuật toán Iterative Dual Correspondence IDC

Thuật toán IDC coi phép biến đổi R sẽ bao gồm phép dịch chuyển *T* và phép xoay *w*. Thuật toán sử dụng 2 quy tắc để xác định sự so khớp giữa các điểm là Điểm gần nhất và So khớp phạm vi điểm:

- Quy tắc điểm gần nhất:
  - So khớp hai điểm mà chúng là gần nhất
  - Sử dụng đặc trưng tiêu biểu theo phép dịch chuyển *T* mà không sử dụng phép xoay *w*
- Quy tắc so khớp phạm vi điểm:
  - So khớp các điểm có cùng khoảng cách (phạm vi) từ các vị trí tương ứng của chúng và nằm trong một vòng quay được xác định trước của nhau
  - Sử dụng đặc trưng tiêu biểu theo phép xoay w mà không sử dụng phép dịch chuyển T (nội suy tuyến tính)

$$\hat{r} = \frac{r_1 r_2(\theta_2 - \theta_1)}{r_1(\hat{\theta} - \theta_1) + r_2(\theta_2 - \hat{\theta})} \quad (1)$$

  - Giả sử các vị trí ban đầu gần nhau (T là không đáng kể)

Dựa trên 2 quy tắc trên, thuật toán IDC tìm phép dịch chuyển T và phép xoay w:

1. Tìm vị trí gần nhất P_{cp} bằng quy tắc điểm gần nhất dựa trên dữ liệu Scan trước S_{ref} và Scan hiện tại S_{new}
2. Tìm vị trí P_{mrp} bằng quy tắc so khớp phạm vi điểm dựa trên dữ liệu S_{ref} và S_{new}
3. Tìm phép biến đổi (w_{cp}, T_{cp}) từ vị trí P_{cp} và (w_{mrp}, T_{mrp}) từ vị trí P_{mrp} bởi tối thiểu lỗi (giải pháp hình vuông nhỏ nhất)
4. Chọn phép biến đổi (w_{mrp}, T_{cp}) và tính sai số
5. Lặp lại các bước từ 1 đến 4 cho đến khi sai số <ngưỡng thiết lập

Như vậy để tìm ra phép biến đổi dịch chuyển và xoay, cần tối thiểu hóa bình phương khoảng cách giữa các điểm cần so khớp:

$$E_{dist}(w,T) = \sum_{i=1}^{n} \left| M_w P_i + T P_i' \right|^2 \quad (2)$$

Trong đó:

- *w* là góc xoay
- T là phép dịch chuyển
- P_i , P_i' là các điểm so khớp
- M_w là ma trận xoay cho phép xoay của góc *w*

Thuật toán sẽ chính xác hơn nếu các vòng lặp hội tụ (thực tế chứng minh với 15-20 vòng lặp là đủ)

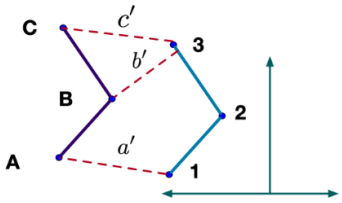
2.3.2 Thuật toán Iterative Closest Points ICP

Thuật toán ICP tìm kiếm lặp đi lặp lại để tìm phép biến đổi tốt nhất:

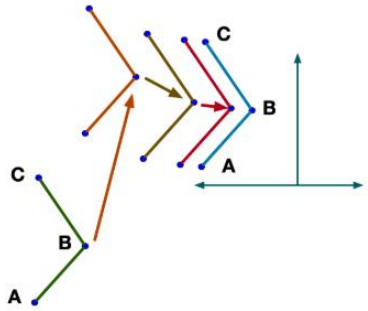
1. Thiết lập một "dự đoán" biến đổi của R (dự đoán hiện tại)
2. Với mỗi điểm ở lần scan mới (t=k+1), tìm điểm gần nhất trong tập hợp các điểm của lần scan trước (t=k) (**tìm kiếm tương ứng)**
3. Tạo một "dự đoán tốt hơn" của R (dự đoán tiếp theo)
4. Gán dự đoán hiện tại là giá trị của dự đoán tiếp theo, lặp lại các bước từ 2 đến 4 cho đến khi hội tụ

Thuật toán ICP lựa chọn biến đổi tốt hơn giữa 2 biến đổi dựa trên đo lường từ Điểm đến Điểm (point – to – point), cụ thể là so sánh hàm tổng khoảng cách giữa các điểm:

$$Score = |a'|^2 + |b'|^2 + |c'|^2 \quad (3)$$



Lặp lại thuật toán cho đến khi hội tụ để tìm ra kết quả



Hình 2.3.3 Minh họa lần lượt các bước lặp dự đoán để so khớp giữa 2 vị trí

Các hạn chế của thuật toán:

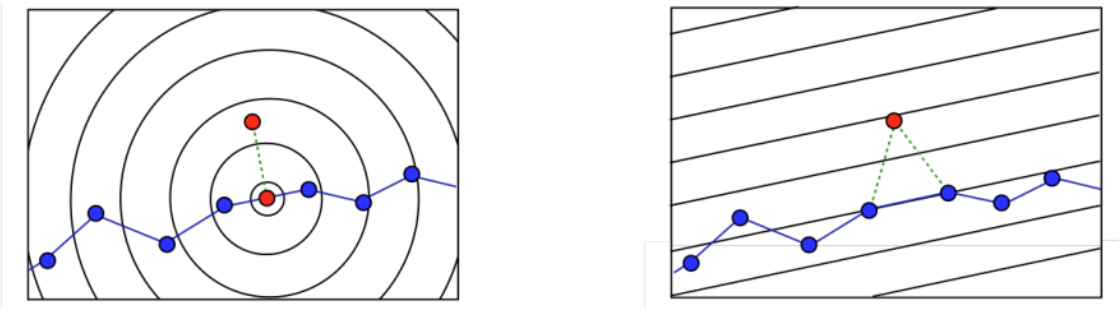
- Thuật toán hoạt động tốt nhất với các giả định sau nhưng lại ít xảy ra trong thực tế:
  - Cần tốc độ quét cực nhanh để có: sự chồng chéo đáng kể tồn tại giữa các lần quét liên tiếp và tồn tại các điểm có sự tương ứng cao
  - Giả định bề mặt nhẵn và đồng nhất
- Thiết lập dự đoán là quan trọng trong ICP để hội tụ
- Việc lặp lại tính toán giữa từng điểm tốn nhiều tài nguyên và thời gian

2.3.3 Thuật toán PL-ICP

Thuật toán PL-ICP là một cải tiến trong việc "tìm kiếm tương ứng" để mang lại hiệu quả hơn. Chuyển từ việc tính toán giữa Điểm tới điểm (point-to-point) sang tính toán giữa Điểm tới đường (**P**oint-to-**l**ine).

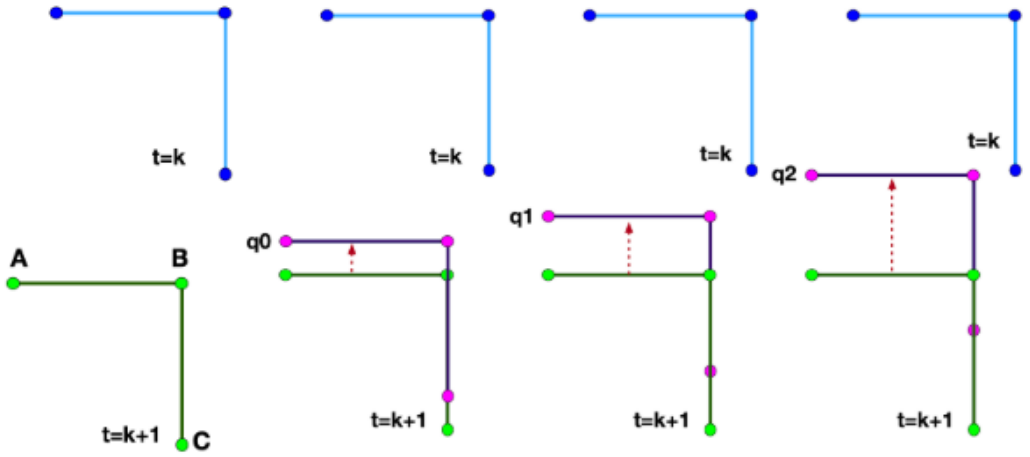
Với point-to-point, các đường viền của tập cấp là đường tròn đồng tâm và tập trung tại điểm chiếu. Với point-to-line, các đường viền của tập cấp dạng đường. Do đó, mức thiết lập bề mặt gần đúng tốt hơn. Điều này có nghĩa là chúng ta có

một xấp xỉ chính xác hơn của lỗi, giảm không gian biến đổi, kết quả **hội tụ nhanh hơn**



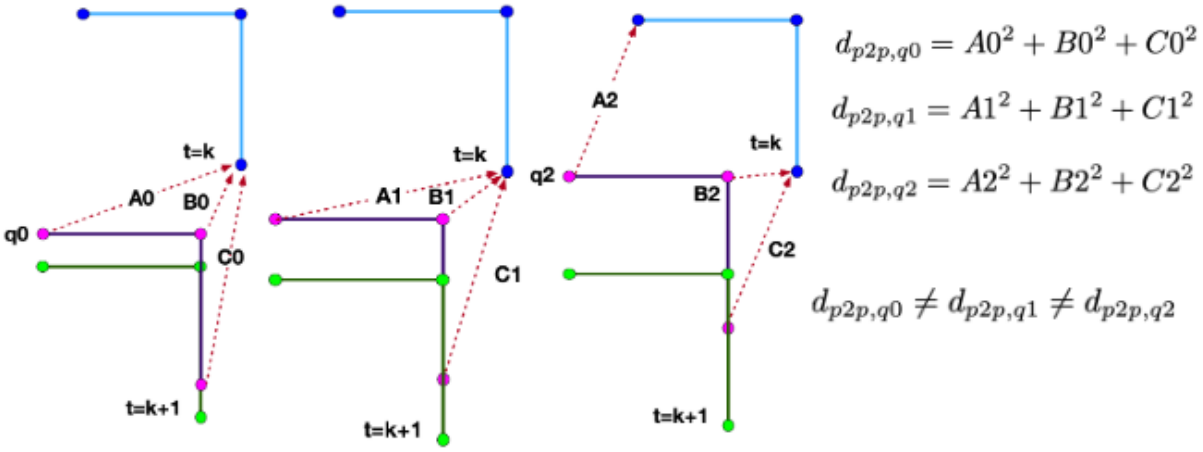
Hình 2.3.4 Dữ liệu điểm-điểm xấp xỉ khoảng cách đến bề mặt tốt hơn so với số liệu điểm-điểm được sử dụng trong ICP vanilla. [9]

Ta lấy ví dụ minh họa, để so khớp thời điểm t=k và thời điểm t=k+1, lần lượt có các dự đoán q0, q1, q2 cần tính toán sự tương ứng



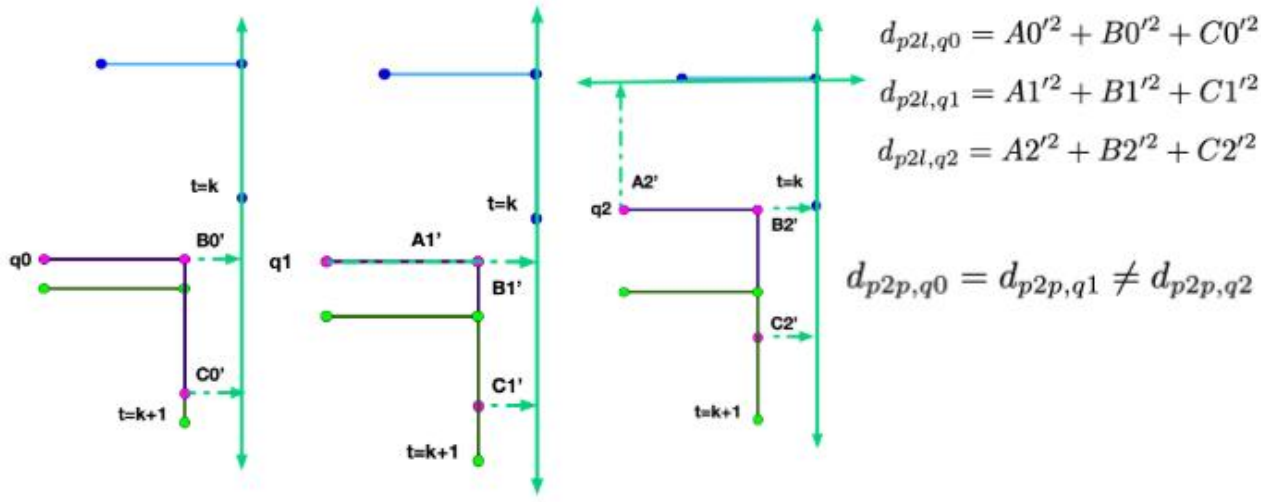
Hình 2.3.5 Minh họa các dự đoán

Đối với phương pháp tính toán point-to-point, ta phải lặp đủ số lần do tính toán các khoảng cách so khớp của các dự đoán q0,q1,q2 khác nhau



Hình 2.3.6 Minh họa phương pháp tính point-to-point

Với point-to-line, chúng ta do kết quả tính toán bằng nhau, số lần lặp được giảm bớt (lần lặp tiếp theo từ q0 sang q2, bỏ qua lần lặp q1 do kết quả giống nhau)



Hình 2.3.7 Minh họa phương pháp tính point-to-line

Tổng quan thuật toán:

- Dữ liệu đầu vào: Dữ liệu Scan tham khảo y_(t-1) thời điểm t-1, dữ liệu scan hiện tại y_t, thiết lập dự đoán của phép biến đổi q_0
- Đầu ra: Phép biến đổi q, là kết quả đại diện cho phép biến đổi giữa các lần Scan

<u>Cụ thể:</u>

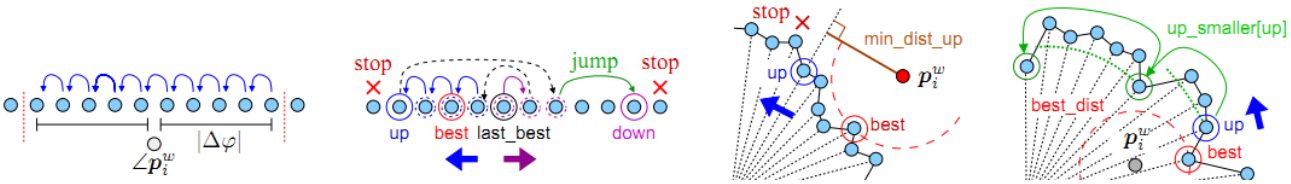
Tại thời điểm t=k

1. Sử dụng dự đoán trước đó q_k, biến đổi tọa độ của lần scan hiện tại vào cùng 1 khung hình với lần scan trước đó
2. Với mỗi điểm, tìm đường thẳng gần nhất (tìm kiếm hội tụ point-to-line)
3. Cập nhật phép biến đổi
    a. Xây dựng hàm point-to-line-error
    b. Tìm biến đổi q_(k+1) để tối thiểu hàm này

    Lặp lại cho đến khi hội tụ

Để việc tìm kiếm nhanh hội tụ hơn, chúng ta cố một số cải tiến [9]:

1. Tìm kiếm nội bộ với kết thúc sớm (Early stopping)
2. Sử dụng Jump table cho mỗi điểm scan



(a) Naive algorithm (b) Smart algorithm (c) Early stopping (d) Using jump tables

Hình 2.3.8 So sánh các cải tiến

Giả mã cho việc cải tiến tìm kiếm nhanh hội tụ hơn: [9]

```
// Out of the main loop, we remember the last match found.
int last_best = invalid;

for(each point p_i^w in scan y_t ) {

// Current best match, and its distance
   int best = invalid; double best_dist = ∞;
// Approximated index in scan y_{t-1} corresponding to point p_i^w
   int start_index = (∠p_i^w − φ0) · (nrays/2π);
// If last match was succesful, then start at that index + 1
   int we_start_at = (last_best != invalid) ? (last_best + 1) : start_index;
// Search is conducted in two directions: up and down
   int up = we_start_at+1, down = we_start_at;
// Distance of last point examined in the up (down) direction.
   double last_dist_up = ∞, last_dist_down = ∞;
// True if search is finished in the up (down) direction.
   bool up_stopped = false, down_stopped = false;

// Until the search is stopped in both directions...
   while ( ! (up_stopped && down_stopped) ) {
// Should we try to explore up or down?
      bool now_up = !up_stopped & (last_dist_up < last_dist_down);
// Now two symmetric chunks of code, the now_up and the !now_up
      if(now_up) {
// If we have finished the points to search, we stop.
         if(up >= nrays) { up_stopped = true; continue; }
// This is the distance from p_i^w to the up point.
         last_dist_up = ||p_i^w − p_up||^2;
// If it is less than the best point, up is our best guess so far.
         if( correspondence is acceptable && last_dist_up < best_dist)
            best = up, best_dist = last_dist_up;
         if (up > start_index) {
// If we are moving away from start_cell we can compute a
bound for early stopping. Currently our best point has distance
best_dist; we can compute the minimum distance to p_i^w for
points j > up (see figure 4(c)).
            double Δφ = φup − ∠p_i^w;
            double min_dist_up = sin(Δφ) ||p_i^w||;
            if( [ min_dist_up ]^2 > best_dist) {
         // If going up we can't make better than best_dist,
then we stop searching in the "up" direction
               up_stopped = true; continue;
            }
// If we are moving away, then we can implement the jump tables
optimization.
            up = // Next point to examine is...
            (ρup < ||p_i^w||) ? // is p_i^w longer?
              up_bigger[up] // then jump to a further point
            : up_smaller[up]; // else, to a closer one.
         } else
// If we are moving towards "start_cell", we can't do any ot the
previous optimizations and we just move to the next point.
            up++;
      } // if(now_up)
// This is the specular part of the previous chunk of code.
      if(!now_up) { ... }
   }
   // Set null correspondence if no point matched.
   ...
   // For the next point, we will start at best
   last_best = best;
}
```

Hình 2.3.9 Giả mã cho việc cải tiến

Triển khai thuật toán trên thực tế, và thực hiện các thí nghiệm kiểm chứng khác nhau, có kết quả so sánh trung bình giữa các thuật toán [9]:

Bảng 2.3.1 Kết quả so sánh trung bình giữa các thuật toán

	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5	Experiment 6	Avg accuracy
IDC	83.31	83.12	82.95	81.96	79.54	74.94	80.97
ICP	57.75	56.62	56.62	56.3	54	52.18	55.58
PL ICP	99.85	99.71	99.51	98.43	84.48	73.46	92.57

	avg. iterations	*avg. execution time*
ICP	34.7	0.083 s (12.0 Hz)
IDC	30.4	0.240 s (4.1 Hz)
PLICP	7.2	0.0018 s (539 Hz)

Ta thấy PL ICP có độ chính xác, tổng số vòng lặp và thời gian xử lý trung bình vượt trội so với thuật toán ICP và IDC. Như vậy PL ICP là thuật toán tối ưu nhất để ứng dụng trong thực tế.

CHƯƠNG 3. TRIỂN KHAI THỰC TẾ THUẬT TOÁN SCAN MATCHING

3.1 Hệ điều hành ROS

3.1.1 Giới thiệu tổng quan

Hệ điều hành robot – **Robot Operating System** là một nền tảng mã nguồn mở dùng cho các ứng dụng trên robot. Mặc dù "hệ điều hành" là một phần của tên của nó, ROS bản thân không phải là một hệ điều hành. Thực tế, bước đầu tiên và cũng là bắt buộc để bắt đầu với ROS, là cài đặt Hệ điều hành mà nó chạy trên: Ubuntu Linux. Tuy không phải là một hệ điều hành hoàn chỉnh, ROS có những khả năng cơ bản cho một hệ điều hành như thực hiện các tác vụ (task) song song; trao đổi, giao tiếp dữ liệu giữa các task bằng thông điệp (message) và quản lý dữ liệu.

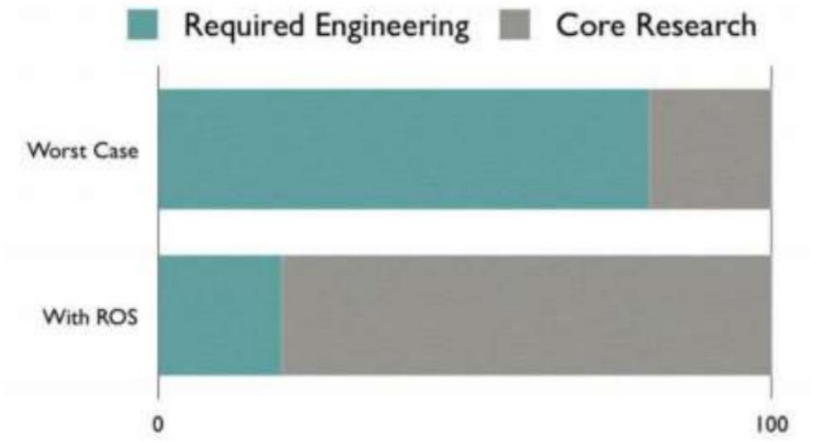
Để có thể ứng dụng trong lĩnh vực robotics, ROS đã phát triển các khái niệm, thư viện và công cụ chuyên biệt dành cho việc thu thập, xử lý dữ liệu, hiển thị và điều khiển. Ngoài ra ROS còn có thể kết hợp và tương tác với nhiều framework khác nhau, ….

ROS có nhiều định nghĩa về mỗi quan hệ giữa các thành phần trong hệ điều hành như stack, package, node, topi, meseage, service, … cũng như các khái niệm hệ tọa độ, phép chuyển đổi hệ tọa độ, graph, biểu diễn mối quan hệ giữa các thành phần trong hệ điều hành. Về khía cạnh trao đổi dữ liệu và giao tiếp, ROS có tích hợp sẵn một số chuẩn giao tiếp đồng bộ qua service, truyền dữ liệu bất đồng bộ qua topic và lưu trữ dữ liệu trên Parameter Service.

ROS cùng với các công cụ và thư viện hỗ trợ thường phát hành dưới dạng ROS Distribution tương tự Linux Distribution, cung cấp những bộ phần mềm ổn định cho người dùng sử dụng và phát triển thêm. ROS thu hút được rất nhiều sự quan tâm và đóng góp của cộng đồng mã nguồn mở trên khắp thế giới để xây dựng phát triển các dự án robotics, các công cụ và thư viện kèm theo. Đã có rất nhiều mô hình robot được xây dựng thành công với hệ điều hành ROS như ROS PR2 (Personal Robot), Turtlebot, Jaguar, Kobuki, …

Hiện nay ROS chạy trên hệ điều hành Ubuntu, Mac OS X và gần nhất vừa ra mắt phiên bản chạy trên hệ điều hành Windows

Đặc điểm nổi bật của ROS chính là xây dựng ứng dụng robotic trên nền tảng ROS sẽ giảm đi một lượng đáng kể các công việc lập trình, thiết lập hệ thống, tận dụng nguồn tài nguyên mã nguồn mở vô cùng phong phú của cộng đồng. ROS đã thống kê và đưa ra kết quả so sánh khối lượng công việc kĩ thuật cơ bản (required engineering) và khối lượng nghiên cứu khoa học nòng cốt (Core Research) như sau:



Hình 3.1.1 So sánh khối lượng công việc phải làm khi dùng và không dùng ROS [10]

Từ kết quả trên ta có thể thấy rõ ràng, với sự hiệu quả từ ROS thời gian dành cho các công việc kỹ thuật cơ bản sẽ được giảm xuống đáng kể, do đó tăng thời gian dành cho công việc nghiên cứu chuyên sâu, hàm lượng khoa học sẽ lớn hơn nhiều lần

Bên cạnh đó, vấn đề cốt lõi khiến ROS trở nên mạnh mẽ đó là nguồn tài nguyên được cộng đồng đóng góp rất lớn, hầu như được xây dựng phát triển từ những viện nghiên cứu và những trường đại học hàng đầu trên thế giới.

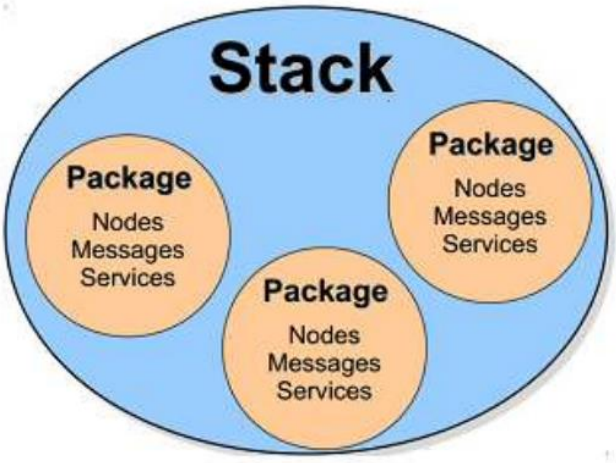
3.1.2 Cấu trúc ROS

ROS có 3 cấp khái niệm: Filesystem, Computation Graph và Community. Ngoài ra ROS còn có một số khái niệm cấp cao đặc trưng cho các ứng dụng robot như hệ tọa độ, phép chuyển đổi, thông điệp mô tả… Tất cả các khái niệm này đều được mô tả chi tiết tại [10], cụ thể như sau:

ROS Filesystem Level

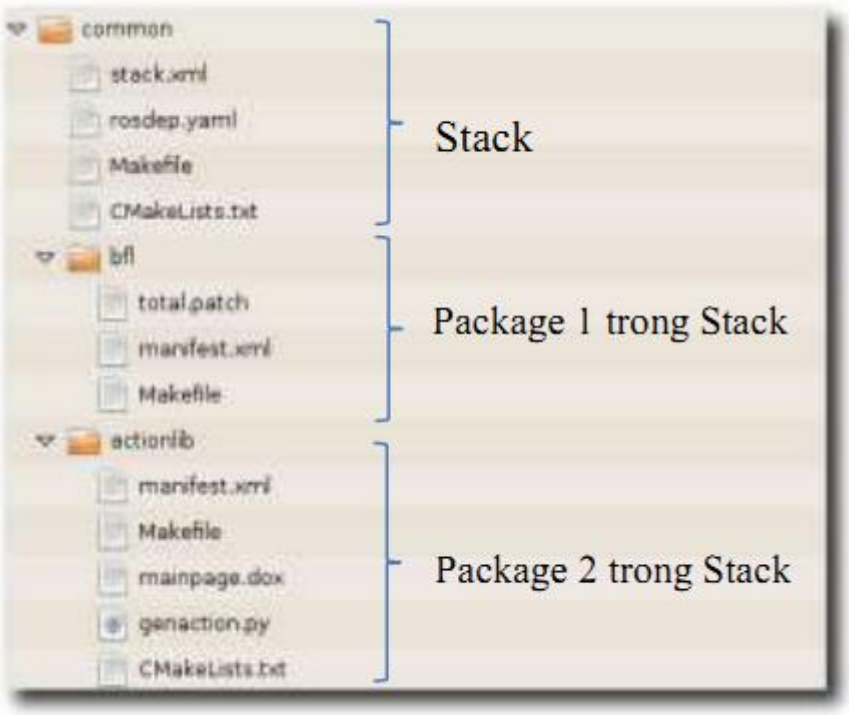
Filesystem là nguồn tài nguyên source code ROS được lưu trữ trên bộ nhớ hệ thống, nó bao gồm các khái niệm:

- Package: Gói ứng dụng là đơn vị cơ bản trong tổ chức phần mềm của ROS, một gói chứa source code cho một tác vụ thực thi một chức năng đặc thù, danh mục các mã nguồn kế thừa (dependency, là các filesystem ngang cấp được đơn vị này dựa trên), các file cấu hình (như file CMakeList.txt chứa các lệnh hướng dẫn biên dịch, yêu cầu tạo file thực thi .bin; hay chọn version cho các thư viện)…
- Manifest: là bảng kê khai thông tin mô tả một packages (manifest.xml), cung cấp các cơ sở dữ liệu về package đó, bao gồm điều kiện thực thi (license) và các dependency của package đó. Ngoài ra, manifest còn chứa những thông tin về đặc trưng của ngôn ngữ lập trình như cờ báo (flags) của trình biên dịch.
- Stacks: là tập hợp các packages phối hợp với nhau để thực hiện một chức năng cụ thể. Stack còn mô tả cách thức biên dịch ROS và thông tin về phiên bản ROS tương thích (gọi là distro, ví dụ như các phiên bản ROS hydro, groovy hay fuerte).



Hình 3.1.2 Mối quan hệ giữa Stack và các Package

- Stack Manifests (.xml): cung cấp mô tả cơ sở dữ liệu về một stack, bao gồm điều kiện cho phép (license) và thông tin về các stack dependency khác.
- Dependency: là mô tả trong manifest của một package hay stack về các file system ngang cấp (stack hoặc package khác) mà stack hay package kế thừa.



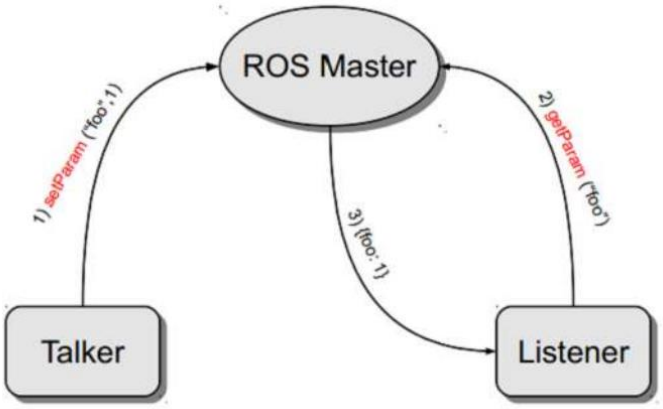
Hình 3.1.3 Ví dụ về quan hệ giữa Stack, Package và các file mô tả theo dạng thư mục

- Message (.msg): thông tin mô tả message, được lưu trữ trong file có dạng my_package/msg/MyMessageType.msg, định nghĩa các cấu trúc dữ liệu của messages được gửi trong ROS.
- Service (.srv): thông tin mô tả các services, được lưu trữ trong my_package/srv/MyServiceType.srv, định nghĩa cấu trúc dữ liệu cho các lệnh truy cập (request) và các phản hồi (response) của các services trong ROS.
- Launch (.launch): là các file .launch dùng để khởi tạo một tập hợp các node cùng lúc, đồng thời gán giá trị cho các parameter và gán các topic liên kết các node bằng lệnh roslaunch trong Command Terminal của Linux.

ROS Computation Graph

Computation Graph, tạm gọi là lược đồ tính toán, là mạng peer-to-peer các tác vụ khi thực thi của ROS, trong đó các dữ liệu được trao đổi và xử lý giữa các node. Các khái niệm cơ bản của computation Graph của ROS là các node, master, parameter sever, message, service, topics và bags.

- Node: là đơn vị thực hiện một tác vụ tính toán, điều khiển. Một node có thể được khởi tạo khi biên dịch thành công một package và có thể được khởi tạo nhiều node từ cùng một package. Một hệ thống điều khiển thường bao gồm nhiều node. Ví dụ một node điều khiển động cơ, một node thực hiện tác vụ định vị, một node vẽ quỹ đạo đường đi...
- Master: ROS Master cung cấp tên đăng ký và tra cứu đến phần còn lại của Computation Graph. Nếu không có Master, các nodes sẽ không tìm thấy nhau để trao đổi thông tin hay gọi services.
- Parameter Server: là một phần của Master, cho phép dữ liệu được lưu trữ trong một vị trí trung tâm và cho phép các node truy cứu tới.

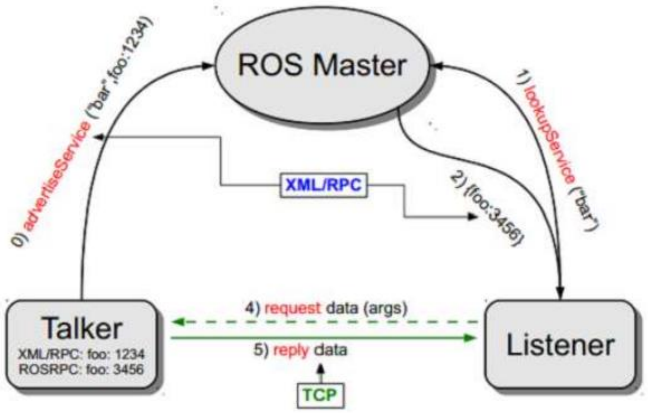


Hình 3.1.4 Mô tả cơ chế quản lý parameter trên Master

- Message: các nodes giao tiếp với nhau thông qua các messages. Một message trong computation graph là dữ liệu cụ thể có cấu trúc như trong khai báo của file .msg tương ứng. Các kiểu dữ liệu chuẩn như integer, floating, point, Boolean… và mảng (array) với kiểu chuẩn đều được hỗ trợ. Bên cạnh đó, message cũng có thể bao gồm các cấu trúc và các mảng lồng nhau như cấu trúc trong ngôn ngữ C. Các node khi nhận Message cần phải xác định queue đệm để xử lý dữ liệu nhận được.
- Topic: Messages được định tuyến thông qua một hệ thống vận chuyển (transport system), trong đó phân loại thành hai công việc chính: publish (đăng tin) và subcribe (đăng ký nhận tin). Một node gửi đi một message bằng việc đưa thông tin tới một topic (chủ đề). Một topic có tên và kiểu message xác định. Một node chỉ subscribe đến đúng topic có tên và kiểu

dữ liệu như đã khai báo. Một topic có thể có nhiều đối tượng đưa tin (publishers) và cũng có thể có nhiều đối tượng đăng ký nhận tin (subcribers). Mỗi node cũng có thể truyền tin trên nhiều topic khác nhau, cũng như có thể nhận tin từ nhiều topic khác nhau. Các nguồn truyền tin và các đối tượng nhận tin nhìn chung không cần phải biết về sự tồn tại của nhau. Ý tưởng xây dựng ROS ở đây là tách biệt nguồn tạo ra thông tin với bộ phận sử dụng thông tin đó. Topic được xem như là một kênh truyền các thông điệp được định kiểu. Mỗi kênh truyền này có một tên riêng, và node nào cũng có thể kết nối với kênh này để gửi/nhận thông điệp, miễn là thông điệp cùng kiểu với topic đó.

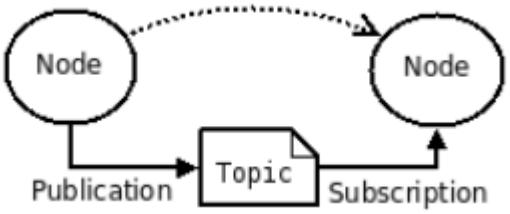
- Service: Mô hình truyền thông theo mẫu publish/subcribe như trình bày ở trên là một mô hình rất linh hoạt, tuy vậy, đặc điểm của nó là thông tin được truyền đa đối tượng, một chiều (many-to-many, one-way) đôi khi lại không phù hợp với các trường hợp cần tương tác theo kiểu request/reply (yêu cầu/đáp ứng), kiểu tương tác này thường gặp trong các hệ thống phân phối. Do vậy, cần có thêm một thành phần nữa trong ROS Graph, đó là service, nhằm thực hiện được các yêu cầu tương tác theo kiểu request/reply. Service là một cặp cấu trúc thông điệp: một thông điệp để gửi yêu cầu và một thông điệp dành cho đáp ứng. Một node cung ứng một service với một thuộc tính name, một client sử dụng service đó bằng cách gửi đi một thông điệp yêu cầu (request message) rồi đợi phản hồi. Trong thư viện client của ROS, phương thức tương tác này thường được cung cấp như một hàm được gọi từ xa.



Hình 3.1.5 Mô tả hoạt động của service

- Bag: là một định dạng để lưu trữ và phát lại dữ liệu từ ROS messages. Bags là một cơ chế quan trọng để lưu trữ dữ liệu.

Cách xây dựng hệ thống trong ROS cho phép nguồn cung cấp tin và đối tượng nhận tin có thể tách rời nhau, và mối liên hệ được thực hiện thông qua thuộc tính name. Name là thuộc tính đóng vai trò rất quan trọng trong ROS: các nodes, topics, services, và các parameters đều được đặt tên. Mỗi thư viện ROS Client đều hỗ trợ command-line để liên kết các tên này (remapping names), nhờ đó mà chương trình đã được biên dịch có thể cấu hình lại được khi chạy ngay cả khi hoạt động trong một cấu trúc Computation Graph khác.

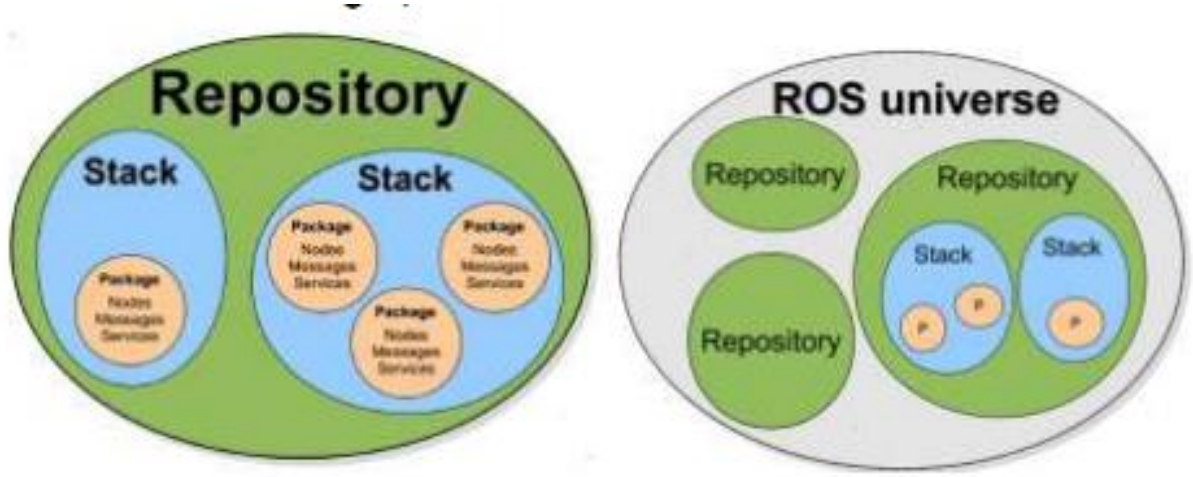


Hình 3.1.6 Mô hình giao tiếp cơ bản trong ROS

ROS Community level

ROS Community được định nghĩa là nguồn tài nguyên ROS mà các đơn vị nghiên cứu có thể trao đổi phần mềm và kiến thức. Các nguồn tài nguyên bao gồm:

- Distribution: ROS Distributions là bộ phiên bản các stack tương thích mà người dùng có thể cài đặt. Phiên bản mới nhất hiện nay là phiên bản thứ bảy Hydro Medusa. Luận văn sử dụng phiêm bản thứ sáu là Groovy Galapagos nhằm đảm bảo tương thích mới một số package như ccny_rgbd ước lượng tọa độ bằng thị giác (visual odometry).
- Repository: ROS là nguồn tài nguyên dựa trên cộng đồng mã nguồn mở, trong đó các viện nghiên cứu, trường đại học khác nhau có thể cùng phát triển đồng thời và công bố những mã nguồn trên mô hình robot của riêng họ.



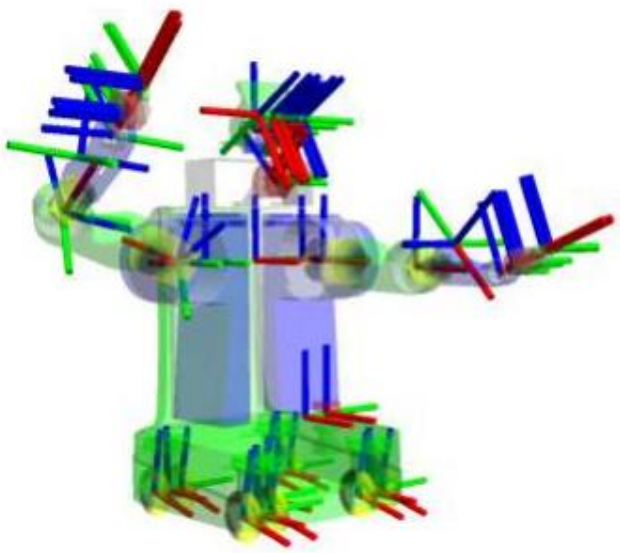
Hình 3.1.7 ROS repository và repository trong toàn tài nguyên ROS

- ROS wiki: là bách khoa mở lưu trữ các tài liệu, hướng dẫn về ROS. Bất kỳ ai cũng có thể đăng ký tài khoản để chia sẻ tài liệu, cập nhật, hay sử dụng, viết bài hướng dẫn và đặt câu hỏi… Đây là một kênh tham khảo quan trọng bậc nhất khi làm quen với ROS.

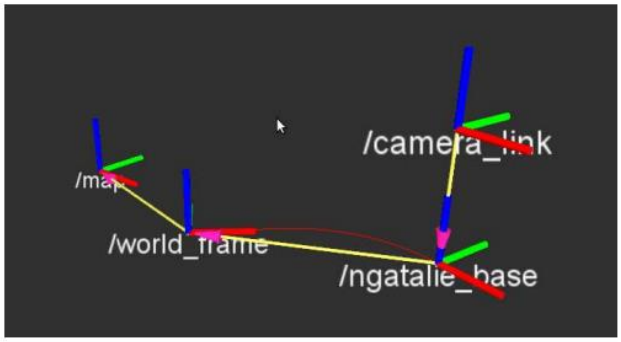
Các khái niệm cấp cao của ROS (Higher-level Concept)

Ngoài các khái niệm cơ bản dựa trên cấu trúc cơ bản của một Operating System. ROS còn quy chuẩn một số khái niệm liên quan đặc trưng tới các ứng dụng robot. Trong phạm vi luận văn, có một số khái niệm quan trọng cần quan tâm như cordinate frame (hệ tọa độ), tf (transform – phép chuyển đổi giữa các hệ tọa độ), và định dạng mô tả robot tổng quát (universal robot description format)…

Coordinate Frame/Transform: Khái niệm hệ tọa độ (frame) và phép chuyển đổi hệ tọa độ (tf) liên quan đến các phần tử không gian của robot và mối quan hệ giữa các phần tử này, giúp cho việc hiểu đúng các dữ liệu từ cảm biến và giám sát, điều khiển robot trong không gian. Các frame trong ROS tuân theo cấu trúc cây, nghĩa là mỗi frame có tối đa một frame mẹ, và có thể có các frame con. Mỗi một nhánh trên cây này tồn tại một tf chuyển đổi giữa frame mẹ và frame con. Package tf cung cấp các hàm dùng để chuyển hệ trục tọa độ, tính toán vị trí và quan hệ giữa các hệ trục tọa độ theo thời gian.

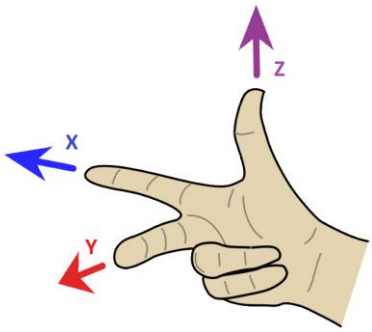


Hình 3.1.8 Các hệ tọa độ gắn với các phần tử chuyển động trên robot



Hình 3.1.9 Các hệ tọa độ của robot và chuyển động trong không gian

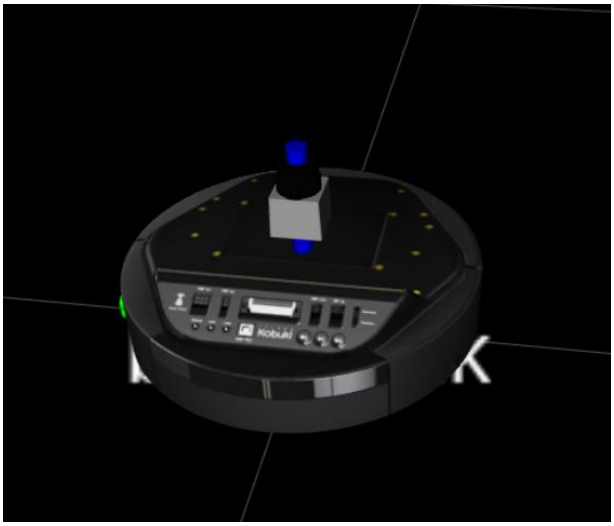
Các quy ước khung tọa độ của ROS tuân thủ quy tắc bàn tay phải, trong đó định hướng trục được quy định x là tiến lùi, y là trái phải, z là lên xuống:



Hình 3.1.10 Quy ước khung tọa độ của ROS tuân theo quy tắc bàn tay phải

Định dạng mô tả robot tổng quát – Universal Robot Description Format: Đây là ngôn ngữ mô tả các phần tử (part), các kết nối giữa các phần tử trên robot (joint). Các joint có thể được khai báo là cố định hoặc chuyển động. URDF là cần thiết cho các mô hình phức tạp có các phần tử như cánh tay robot, robot

omni,… URDF giúp cho việc mô phỏng trong quá trình nghiên cứu phát triển robot.



Hình 3.1.11 Mô hình robot Kobuki kèm cảm biến laser được mô tả bằng URDF

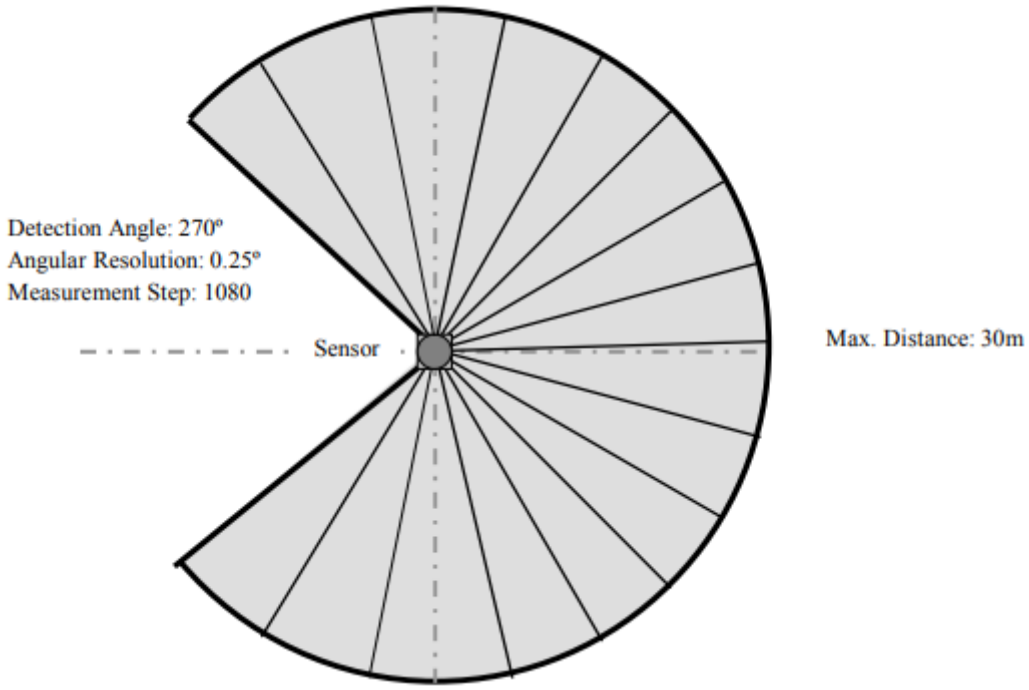
3.2 Cảm biến Laser ranger finder

3.2.1 Công nghệ và đặc tính

Bộ cảm biến là thiết bị điện tử cảm nhận những thay đổi từ môi trường bên ngoài và biến đổi thành các tín hiệu điện để điều khiển các thiết bị khác. Cảm biến laser có nguyên lí hoạt động đơn giản, gồm có nguồn phát tia laser và đầu thu. Tại đầu thu có một màng để đo góc phản xạ, tùy vào độ lớn của góc phản xạ mà tính được khoảng cách từ cảm biến đến vật cản gần nhất theo góc quay. Tùy vào yêu cầu mà thiết bị cảm biến laser có một hoặc nhiều bộ phát – thu từ đó có thể thu về dữ liệu hai chiều hoặc ba chiều với độ chính xác cao. Cảm biến laser ứng dụng trong robot, được ví chính như "đôi mắt" của robot, giúp nhận biết được các vật cản trong tầm quét.

Cảm biến được sử dụng là cảm biến Hokuyo UTM-30LX với thông số kĩ thuật:

- Góc mở 270^0
- Mỗi bước 0.25^0, tổng là 1080 bước
- Khoảng cách vật từ 0.1 đến 10m, sai số ± 30mm, từ 10m đến 30m sai số ± 50mm 1%,
- Tần số quét 25ms/lần,
- Trọng lượng 370g,
- Giao tiếp qua cổng USB 2.0.



Hình 3.2.1 Thông số kĩ thuật cảm biến UTM-30LX

3.2.2 Package urg_node

Package urg_node là package được đóng gói hoàn thiện trên ROS, được giới thiệu tại địa chỉ: http://wiki.ros.org/urg_node
Các API chính của package:

Published Topics:

`scan` (sensor_msgs/LaserScan): Quét dữ liệu từ cảm biến Laser
`diagnostics` (diagnostic_msgs/DiagnosticStatus): Thông tin trạng thái chẩn đoán

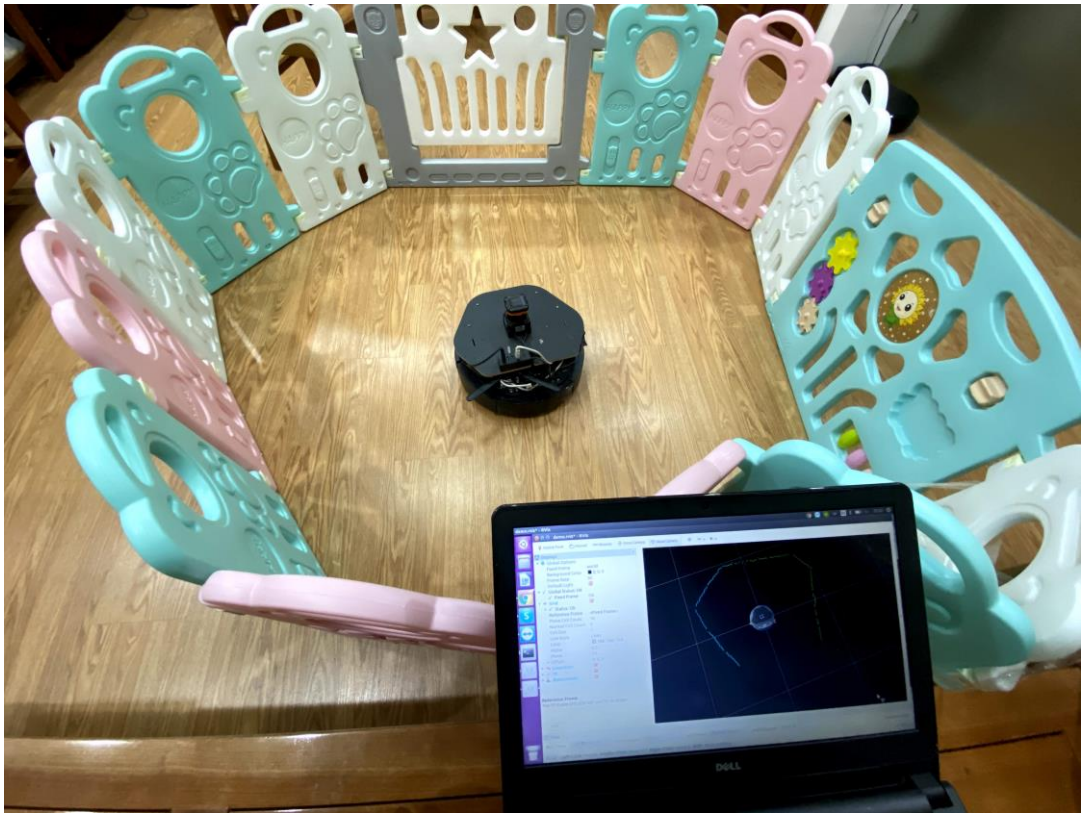
Services

`~self_test` (diagnostic_msgs/SelfTest): Bắt đầu bài tự kiểm tra của Hokuyo, chạy một loạt các thử nghiệm trên thiết bị. Laser dừng xuất bản quét trong quá trình thử nghiệm, mất khoảng một phút. Kết quả của bài kiểm tra là trong phản hồi được trả về bởi dịch vụ này. Khi kết thúc thử nghiệm, laser được đưa trở lại chế độ hoạt động bình thường.

Parameters

`~min_ang` (`double`, mặc định: -π/2): Góc của phép đo phạm vi đầu tiên tính bằng radian (phạm vi là [-π, π], mặc dù hầu hết các thiết bị có phạm vi khả thi nhỏ hơn).
`~max_ang` (`double`, mặc định: π/2): Góc của phép đo phạm vi cuối cùng tính bằng radian (phạm vi là [-π, π], mặc dù hầu hết các thiết bị có phạm vi khả thi nhỏ hơn).
`~intensity` (`bool`, mặc định: `false`): Có hay không hokuyo trả về giá trị cường độ.
~cluster (int, mặc định: 1): Số lượng các phép đo phạm vi liền kề để phân cụm thành một lần đọc; việc đọc ngắn nhất từ cụm được báo cáo.
`~skip` (`int`, mặc định: 0): Số lần quét để bỏ qua giữa mỗi lần quét được đo. Điều này kiểm soát tốc độ cập nhật. Đối với UTM-30LX, hokuyo sẽ quét ở tần số 40Hz, do đó, đặt "bỏ qua" thành 1 sẽ xuất bản ở tần số 20Hz.

~port (string, mặc định: /dev/ttyACM0): Cổng nơi thiết bị hokuyo có thể được tìm thấy.
~calibrate_time (bool, mặc định: true): Liệu node có nên hiệu chỉnh thời gian bù của hokuyo khi khởi động hay không. Nếu đúng, node sẽ trao đổi một loạt tin nhắn với thiết bị để xác định độ trễ thời gian trong kết nối USB. Bước hiệu chuẩn này là cần thiết để tạo ra dấu thời gian chính xác trên bản quét.
~frame_id (string, mặc định: laser): Khung trong đó quét laser sẽ được trả lại. Khung này phải ở trung tâm quang học của laser, với trục x dọc theo tia 0 độ và trục y dọc theo tia 90 độ.
~time_offset (double, mặc định: 0.0): Một đề nghị để thêm vào dấu thời gian trước khi xuất bản Phạm vi quét: -0,25 đến 0,25



Hình 3.2.2 Cảnh thật và dữ liệu thu về từ cảm biến

3.3 Robot Kobuki

3.3.1 Tổng quan Kobuki

Robot di động được sử dụng trong nghiên cứu là robot Kobuki. Kobuki là một robot di động được phát triển dành cho việc học tập. Robot có khả năng cung cấp nguồn cho các modules hoạt động bên ngoài như laptop, cảm biến hay các vi mạch tích hợp. Nó có khả năng di chuyển với độ chính xác cao. Việc tích hợp các cổng vào ra giúp người sử dụng có thể tích hợp các modules mới vào ngay robot, giảm chi phí phát triển các modules này.
Kobuki giao tiếp với máy tính qua RS232 hoặc USB. Tuy nhiên, cổng USB của Kobuki chính là một cổng COM ảo, vì vậy phần mềm điều khiển sẽ chỉ cần giao tiếp với cổng COM để điều khiển robot.

Các trình điều khiển giao tiếp với robot bằng cách sử dụng giao thức được xác định trước, lệnh sẽ được gửi tới robot và robot sẽ gửi trả một số thông tin phản hồi. Các lệnh được gửi dưới dạng luồng các byte dữ liệu được gọi là Bytestream.

Cấu trúc của Bytestream:

Một bytestream có thể được chia thành 4 trường: Headers, Length, Payload và Checksum.

Bảng 3.3.1 Cấu trúc bytestream

Tên	Header 0	Header 1	Length	Payload	Checksum
Kích thước	1 Byte	1 Byte	1 Byte	N Bytes	1 Byte
Mô tả	0xAA (Cố định)	0x55 (Cố định)	Kích thước của Payload tính theo byte	Mô tả phía dưới	Giá trị kiểm soát lỗi của gói tin

- *Header* : Hai byte của header, header 0 và 1, được cố định giá trị cho cả bytestreams, lệnh và data thông tin phản hồi. Header này được sử dụng để phát hiện các điểm khởi đầu của bytestream.
- *Length*: Length chỉ độ dài của sau byte bytestream duy nhất giữ. Kích thước mặc định của lĩnh vực này là 1 byte. Length có thể được sử dụng để phân biệt từng bytestreams. Giá trị nhỏ nhất là 3.
- *Payload*: Payload chứa dữ liệu thiết thực của bytestream. Cơ Cấu Payload được tạo nên bởi các Sub-Payload

Bảng 3.3.2 Cấu trúc payload

Payload				
Sub-Payload 0	Sub-Payload 1	Sub-Payload 2	...	Sub-Payload N-1

Cấu trúc của Sub-Payloads có thể được chia là 3 phần, Header, Length and Data.

Bảng 3.3.3 Cấu trúc Sub-Payloads

Tên	Header	Length	Data
Kích thước	1 Byte	1 Byte	N Byte(s)
Miêu tả	Định danh	Kích thước data	Được miêu tả phía dưới

- Checksum*:* được thực hiện theo cách các bytes trong bytestream được XOR với nhau, ngoại trừ 2 bytes header. Bên dưới là ví dụ hàm Checksum thực hiện theo cách mô tả bên trên

Ví dụ:

```
unsigned int packet_size(buffer.size());
unsigned char cs(0);
for (unsigned int i = 2; i < packet_size; i++)
{
```

```
cs ^= buffer[i];
}
return cs ? false : true;
```

Các lệnh điều khiển robot Kobuki

Bảng 3.3.4 Định danh lệnh

ID	Name	Mô tả
1	Base Control	Điều khiển bánh xe
2	*Reserved*	
3	Sound	Phát âm thanh cố định
4	Sound Sequence	Phát âm thanh được định nghĩa
5	*Reserved*	
6	*Reserved*	
7	*Reserved*	
8	Set Power	Điều khiển cấp nguồn ngoài
9	Request Extra	Yêu cầu thêm thông tin
10	*Reserved*	
11	*Reserved*	
12	General Purpose Output	Điều khiển các cổng ra

Điều khiển cơ bản

Điều khiển động cơ bánh xe để di chuyển robot. Robot sẽ di chuyển theo vòng cung, với bán kính là **<Radius>** mm, và vận tốc **< Speed>** mm / s.

Bảng 3.3.5 Cấu trúc điều khiển cơ bản

	Tên	Kích thước	Giá trị	Giá trị Hex	Mô tả
Header	Định danh	1	1	0x01	Cố định
Length	Kích thước data	1	4	0x04	Cố định
Data	Speed	2			mm / s
	Radius	2			mm

Các lệnh phản hồi từ robot Kobuki

Kobuki gửi các gói tin phản hồi với tần số 50 Hz, khi nó được bật nguồn. Các gói tin đó bao gồm:

Bảng 3.3.6 Lệnh phản hồi

ID	Tên	Miêu tả
1	Basic Sensor Data	Dữ liệu cơ bản
2	*Reserved*	
3	Docking IR	Tín hiệu hồng ngoại
4	Inertial Sensor	Dữ liệu con quay hồi chuyển
5	Cliff	Dữ liệu PSD
6	Current	Dữ liệu về 2 bánh xe
7	*Reserved*	
8	*Reserved*	
9	*Reserved*	
10	Hardware Version	Phiên bản phần cứng
11	Firmware Version	Phiên bản phần mềm
12	*Reserved*	
13	Raw data of 3-axis gyro	Dữ liệu ADC dạng thô của con quay hồi chuyển – IMU Data
14	*Reserved*	
15	Reserved	
16	General Purpose Input	Giá trị của 25 chân đầu vào
17	*Reserved*	
18	*Reserved*	
19	Unique Device IDentifier(UDID)	Định danh cho robot
20	*Reserved*	

Bảng dữ liệu cơ bản

Bảng 3.3.7 Dữ liệu phản hồi cơ bản

	Tên	Kích thước	Giá trị	Hex	Miêu tả
Header	Định danh phản hồi	1	1	0x01	Cố định
Length	Kích thước trường dữ liệu	1	15	0x0F	Cố định
Data	Timestamp	2			Thời gian chạy tính bằng ms (0 – 65535)
	Bumper	1			Cờ này được sét khi cảm biến va chạm

	Tên	Kích thước	Giá trị	Hex	Miêu tả
					có dữ liệu 0x01 cho bumper phải 0x02 cho bumper giữa 0x04 cho bumper trái
	Wheel drop	1			Cờ này được sét khi bánh xe bị kéo xuống 0x01 cho bánh phải 0x02 cho bánh trái
	Cliff	1			Cờ được sét khi robot gặp đá 0x01 cho sensor bên phải 0x02 cho sensor ở giữa 0x04 cho sensor bên trái
	Left encoder	2			Số vòng quay nhận từ encoder với giá trị từ 0 tới 65535
	Right encoder	2			
	Left PWM	1			Giá trị PWM
	Right PWM	1			
	Button	1			Cờ được sét khi nút được bấm 0x01 cho nút 0 0x02 cho nút 1 0x04 cho nút 2
	Charger	1			0: DISCHARGING 2: DOCKING_CHARGED 6: DOCKING_CHARGING 18:ADAPTER_CHARGED 22: ADAPTER_CHARGING
	Battery	1			Điện áp của pin với đơn vị là 0.1V. Giá trị lớn nhất đạt được là 167
	Over current flags	1			Cờ được sét khi bánh bị quá tải 0x01 cho bánh trái 0x02 cho bánh phải

3.3.2 Package kobuki

Package kobuki được đóng gói hoàn chỉnh trên ROS. Được giới thiệu tại địa chỉ: http://wiki.ros.org/kobuki . Package bao gồm nhiều package khác:

- kobuki_auto_docking : Tự động tìm đế sạc cho Kobuki
- kobuki_bumper2pc : Xuất bản các bộ đệm và các sự kiện cảm biến va đập trên pointcloud để Navistack có thể sử dụng chúng
- kobuki_controller_tutorial : Liên quan đến hướng dẫn điều khiển
- kobuki_description : Cung cấp mô tả mô hình của Kobuki để mô phỏng và trực quan hóa. Các tập tin trong gói này được phân tích cú pháp và sử dụng bởi nhiều thành phần khác.
- kobuki_keyop : Điều khiển Kobuki di chuyển bằng bàn phím

- kobuki_node : Trình bao bọc ROS node cho trình điều khiển kobuki
- kobuki_random_walker : Điều khiển Kobuki di chuyển ngẫu nhiên
- kobuki_safety_controller : Đồng hồ cảm biến va đập và bánh xe để cho phép vận hành an toàn
- kobuki_testsuite : Bộ công cụ để kiểm tra kỹ lưỡng phần cứng của Kobuki

Trong luận văn này, chúng tôi sử dụng các package chính sau:

- kobuki_description : Mô phỏng lại phần cứng robot kèm cảm biến laser phục vụ việc giả lập trên rviz
- kobuki_node: sử dụng published topic `~sensors/imu_data` phục vụ cho thuật toán Laser_scan_matcher
- kobuki_random_walker : Điều khiển Kobuki di chuyển ngẫu nhiên ứng dụng cho một số thí nghiệm
- kobuki_keyop : Điều khiển Kobuki di chuyển bằng bàn phím ứng dụng cho một số thí nghiệm

3.4 Thực nghiệm và đánh giá

3.4.1 Xây dựng hệ thống phần cứng và lập trình phần mềm

3.4.1.1. Phần cứng

Đã xây dựng robot Kobuki với các module phần cứng lắp đặt hoàn thiện:

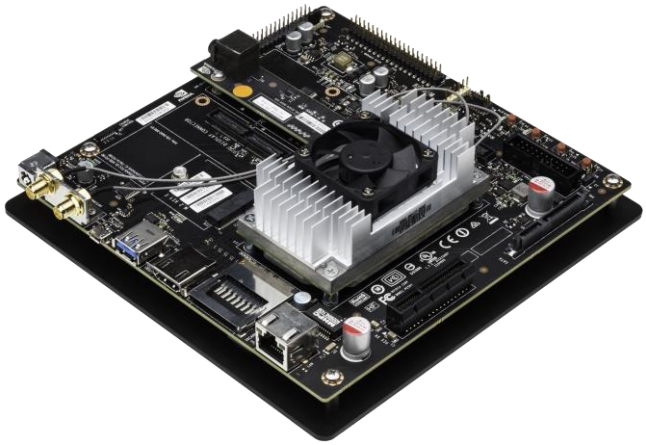
Hình 3.4.1 Mô hình robot hoàn thiện

Bao gồm chi tiết các phần:

- Robot Kobuki: Đế robot Kobuki được lắp thêm 2 tầng

Hình 3.4.2 Robot Kobuki

- Nvidia Jetson TX1 Developer Kit: Kobuki cung cấp nguồn điện 12V-5A kết hợp mạch chuyển đổi để thành nguồn điện 19V cho Board được lắp ở tầng 1 của robot.



Hình 3.4.3

- Cảm biến laser Hokuyo UTM-30LX: Lắp tại tầng 2 của robot Kobuki. Cảm biến sử dụng nguồn điện 5V–1A lấy từ robot Kobuki

Hình 3.4.4 Cảm biến laser Hokuyo UTM-30LX

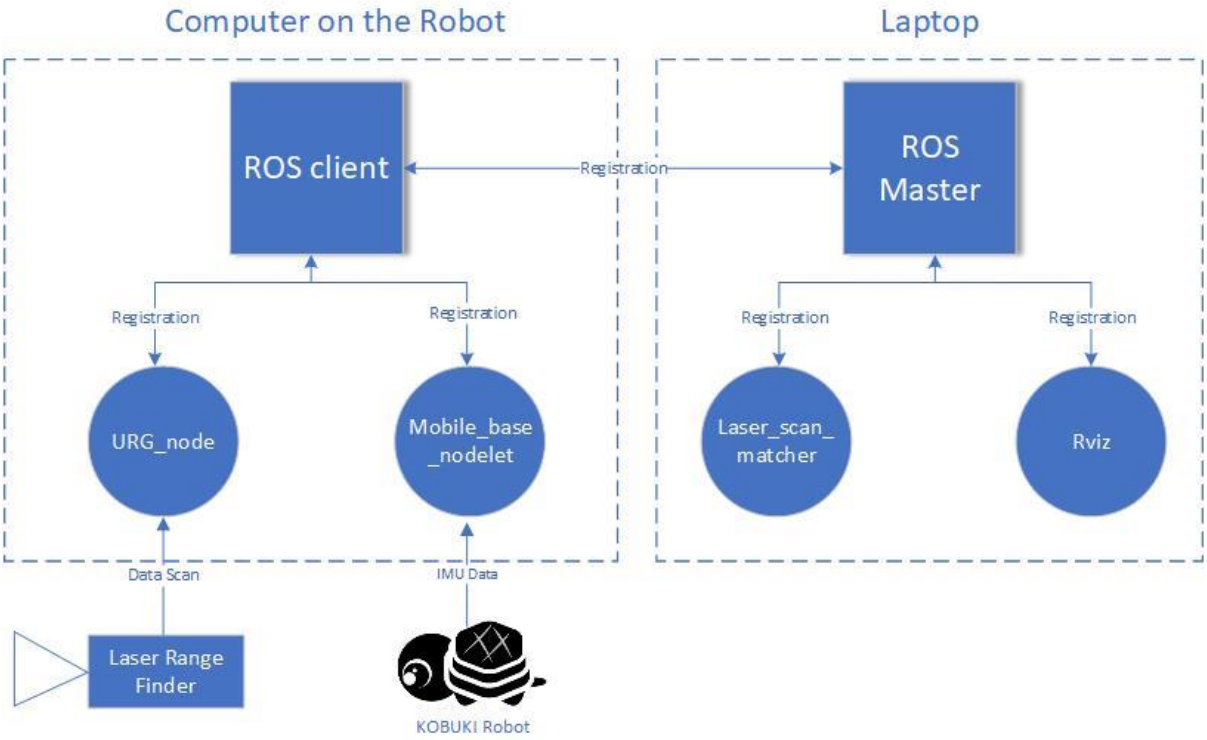
3.4.1.2. Lập trình phần mềm

Hệ thống sử dụng một số package trên hệ điều hành ROS, cụ thể:

- package urg_node: để thu nhận dữ liệu laser từ cảm biến UTM-30LX

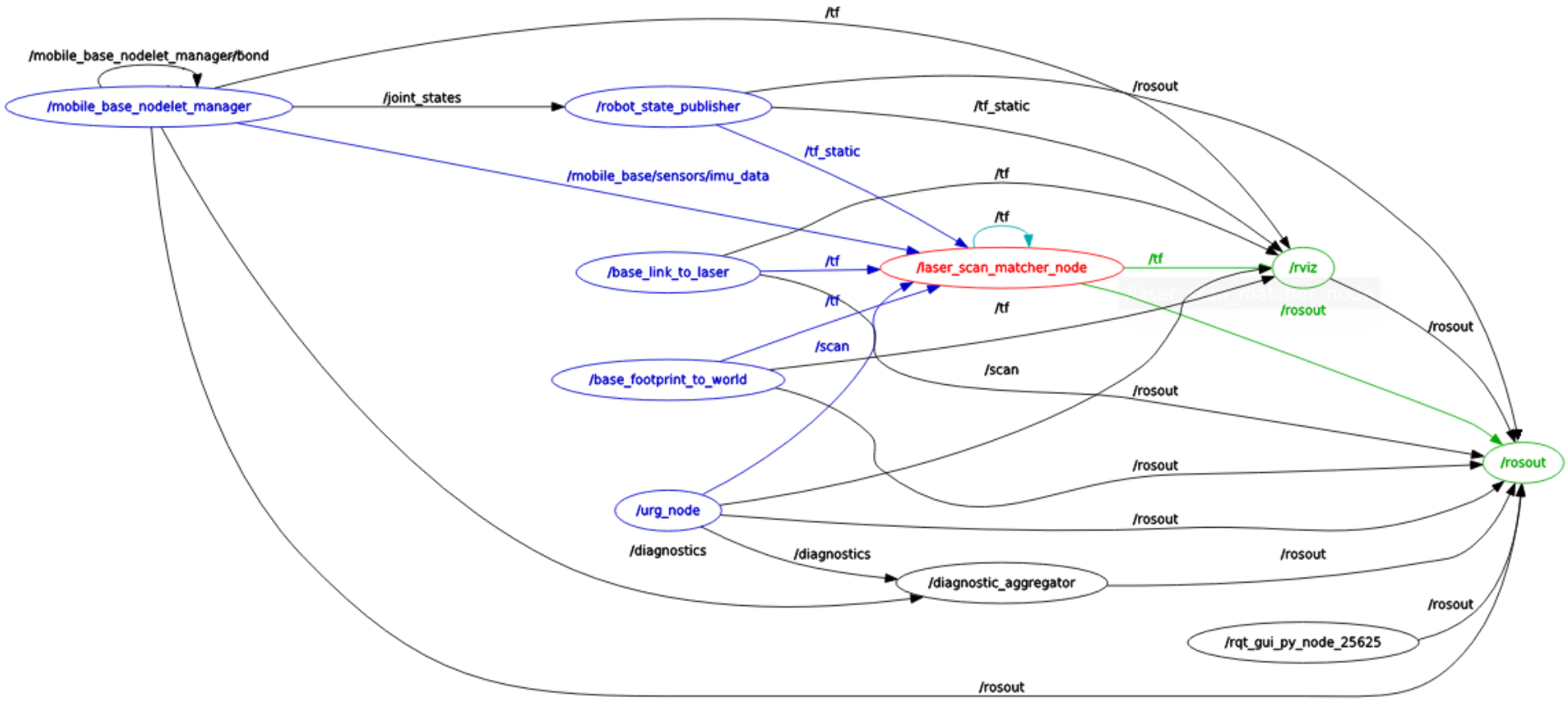
- package mobile_base (thuộc package Kobuki): thu nhận trạng thái robot Kobuki, dữ liệu IMU cũng như điều khiển robot di chuyển
- package laser_scan_matcher (thuộc package Scan_tools): triển khai của thuật toán Andrea Censi's Canonical Scan Matcher (CSM) - triển khai C thuần túy của thuật toán PL-ICP được tối ưu hóa cho kết hợp quét tìm phạm vi
- package rviz: mô phỏng di chuyển của robot, dữ liệu laser cũng như kết quả của thuật toán

Tôi xây dựng file thực thi client.launch chạy trên Nvidia Jetson TX1 Developer Kit để publish và subcrise các message từ Kobuki và laser. Kết nối thông qua mạng WLAN với ROS_MASTER cài trên laptop chạy file thực thi thuật toán laser_scan_matcher.launch. Thuật toán ngoài dữ liệu laser đã sử dụng kết hợp cả dữ liệu imu của kobuki để nâng cao kết quả.



Hình 3.4.5 Mô hình kiến trúc phần mềm

Cụ thể khi chạy chương trình, ta có hình ảnh ros graph về sự tương tác của các nodes, topics với nhau



Hình 3.4.6 ROS graph về sự tương tác của các nodes, topics với nhau

3.4.2 Các thí nghiệm kiểm chứng thuật toán

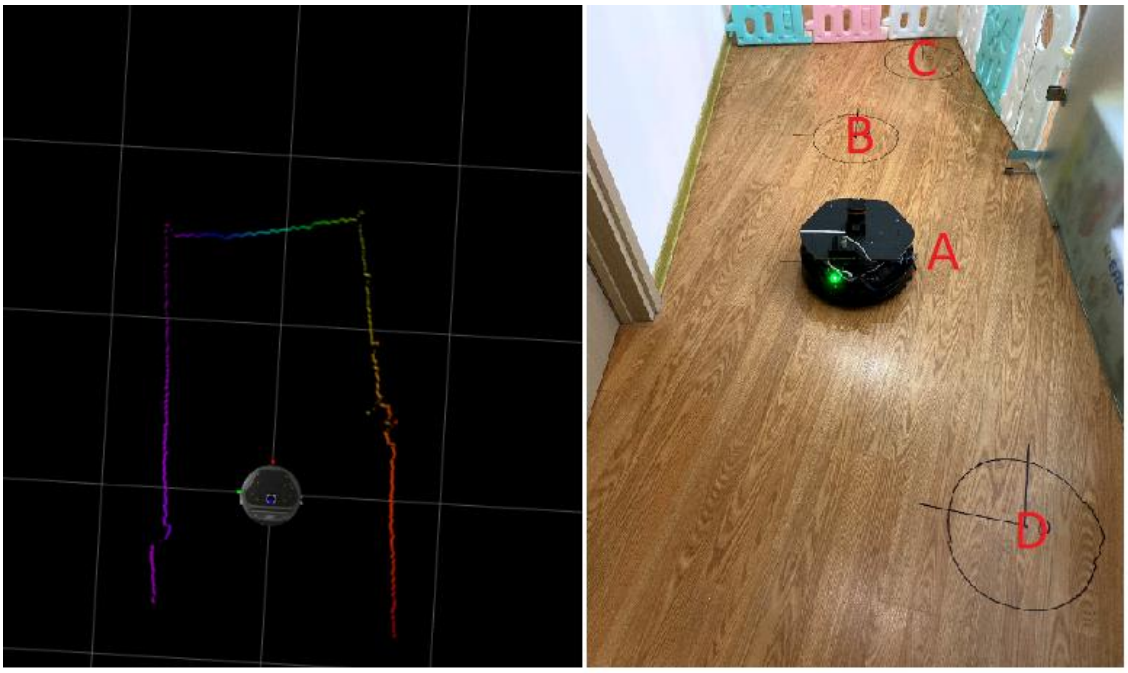
Các thí nghiệm được thực hiện với phần cứng đã được mô tả, trong điều kiện môi trường tiết diện thẳng, tiết diện gồ ghề, hành lang dài, môi trường trong nhà có nhiều phòng nhỏ thông nhau

Cụ thể chúng tôi thực hiện các thí nghiệm như sau:

- Thí nghiệm 1: Xác định vị trí robot trong điều kiện môi trường trong phòng tiết diện thẳng
- Thí nghiệm 2: Xác định vị trí robot trong điều kiện môi trường trong phòng tiết diện gồ ghề
- Thí nghiệm 3: Xác định vị trí robot trong địa hình hành lang dài
- Thí nghiệm 4: Xác định vị trí robot trong môi trường có nhiều phòng nhỏ thông nhau

Thí nghiệm 1: Xác định vị trí robot trong điều kiện môi trường trong phòng tiết diện thẳng

- Mục đích : Kiểm chứng thử nghiệm thực tế của thuật toán PL-ICP trong môi trường tường nhẵn, tiết diện thẳng
- Thí nghiệm được thực hiện theo quy trình sau:
    - Đặt robot vào môi trường tường nhẵn, không có đồ đạc, tiết diện thẳng
    - Robot xuất phát tại vị trí gốc A ở giữa, điều khiển robot xoay 360 độ để quét xung quanh
    - Điều khiển từ xa để robot di chuyển một đoạn đến vị trí mới B trong hình vẽ
    - Thuật toán trả về khoảng cách giữa vị trí hiện tại và vị trí xuất phát
    - Đo trên thực tế giữa vị trí hiện tại và vị trí xuất phát
    - Lặp lại 10 lần quy trình trên
    - Thực hiện lại toàn bộ quy trình trên tương tự với vị trí mới C, D trong hình vẽ



Hình 3.4.7 Xác định vị trí robot trong môi trường tiết diện thẳng: ảnh quét laser (bên trái) và thực tế (bên phải)

- Kết quả:
    - Sai số trung bình là 18,42 mm; phương sai 33,74; độ lệch chuẩn 5,8mm
    - Sai số góc trung bình là $0{,}578^0$; phương sai 0,088; độ lệch chuẩn $0{,}239^0$

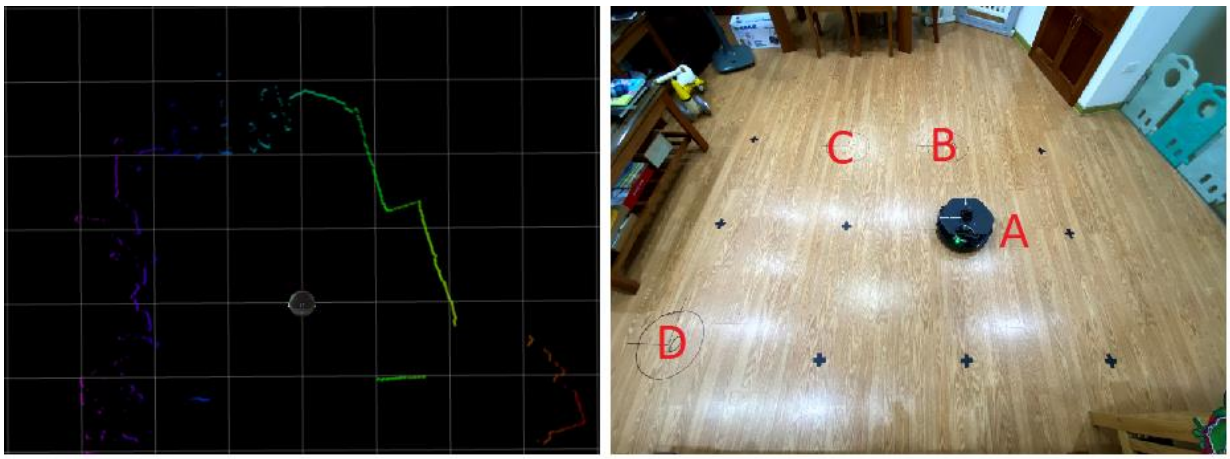
Bảng 3.4.1 Kết quả xác định vị trí robot trong môi trường tiết diện thẳng

	Điểm B (700 mm; 0 mm; 0 độ)		**Điểm C (1350 mm; -450 mm; -50 độ)**		**Điểm D (-770 mm; -480 mm; 45 độ)**	
	Vị trí (mm)	Góc (độ)	Vị trí (mm)	Góc (độ)	Vị trí (mm)	Góc (độ)
Sai số trung bình	17,54	0,0585	18,56	0,7412	19,17	0,9335
Phương sai	37,23	0,0001	34,06	0,1843	29,92	0,0784
Độ lệch chuẩn	6,10	0,0091	5,84	0,4293	5,47	0,2799

Thí nghiệm 2: Xác định vị trí robot trong điều kiện môi trường trong phòng tiết diện gồ ghề

- Mục đích : Kiểm chứng thử nghiệm thực tế của thuật toán PL-ICP trong môi trường tiết diện gồ ghề
- Thí nghiệm được thực hiện theo quy trình sau:
    - Đặt robot vào môi trường trong phòng nhiều đồ đạc, tiết diện thẳng
    - Robot xuất phát tại vị trí gốc A, điều khiển robot xoay 360 độ để quét xung quanh
    - Điều khiển từ xa để robot di chuyển một đoạn đến vị trí mới B trong hình vẽ

- Thuật toán trả về khoảng cách giữa vị trí hiện tại và vị trí xuất phát
- Đo trên thực tế giữa vị trí hiện tại và vị trí xuất phát
- Lặp lại 10 lần quy trình trên
- Thực hiện lại toàn bộ quy trình trên tương tự với vị trí mới C, D trong hình vẽ



Hình 3.4.8 Xác định vị trí robot trong môi trường tiết diện gồ ghề: ảnh quét laser (bên trái) và thực tế (bên phải)

- Kết quả:
  - Sai số vị trí trung bình là 24,84 mm; phương sai 38,12; độ lệch chuẩn 6,103mm
  - Sai số góc trung bình là $0{,}106^0$; phương sai 0,052; độ lệch chuẩn $0{,}19^0$

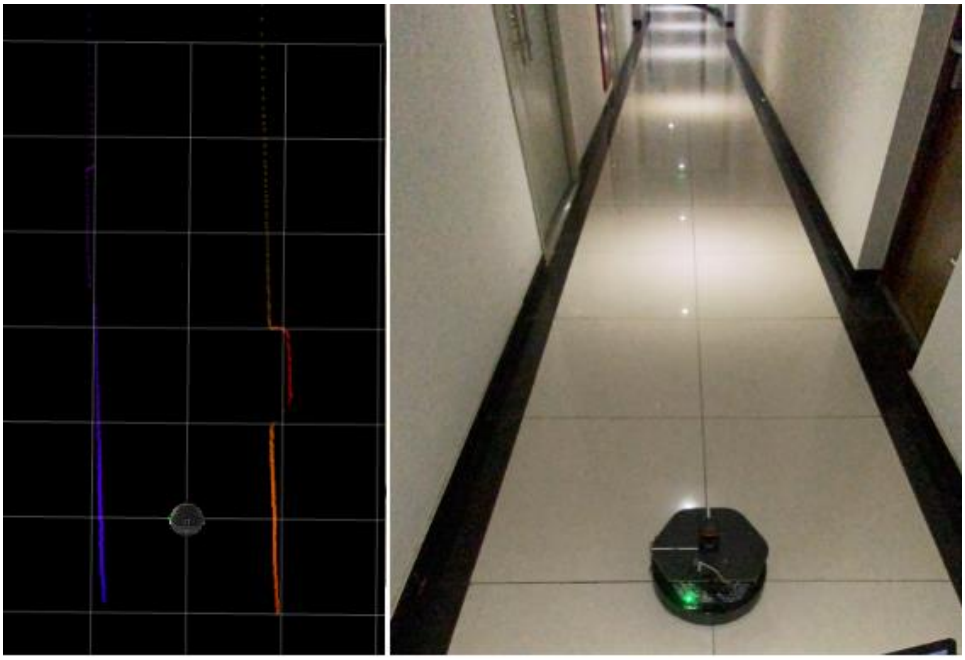
Bảng 3.4.2 Kết quả xác định vị trí robot trong môi trường tiết diện gồ ghề

	Điểm B (800 mm; 0 mm; 0 độ)		**Điểm C (750 mm; 850 mm; 90 độ)**		**Điểm D (-700 mm; 1650 mm; 150 độ)**	
	Vị trí (mm)	Góc (độ)	Vị trí (mm)	Góc (độ)	Vị trí (mm)	Góc (độ)
Sai số trung bình	22,26	0,0399	26,50	0,1186	25,76	0,1585
Phương sai	49,88	0,0002	23,35	0,0686	41,13	0,0866
Độ lệch chuẩn	7,06	0,0144	4,83	0,2619	6,41	0,2942

Thí nghiệm 3: Xác định vị trí robot trong địa hình hành lang dài

- Mục đích : Kiểm chứng sự chính xác của thuật toán trong môi trường hành lang dài (các dữ liệu quét gần như không thay đổi)
- Thí nghiệm được thực hiện theo quy trình sau:
  - Đặt robot vào môi trường hành lang dài khoảng 15m
  - Robot xuất phát tại vị trí gốc A ở khoảng giữa hành lang, điều khiển robot xoay 360 độ để quét xung quanh

- Điều khiển từ xa để robot di chuyển thẳng đến vị trí mới cách vị trí xuất phát lần lượt 600mm, 1200mm, 1800mm
- Thuật toán trả về khoảng cách giữa vị trí hiện tại và vị trí xuất phát
- Đo trên thực tế giữa vị trí hiện tại và vị trí xuất phát



Hình 3.4.9 Xác định vị trí robot trong địa hình hành lang dài: ảnh quét laser (bên trái) và ảnh thực tế (bên phải)

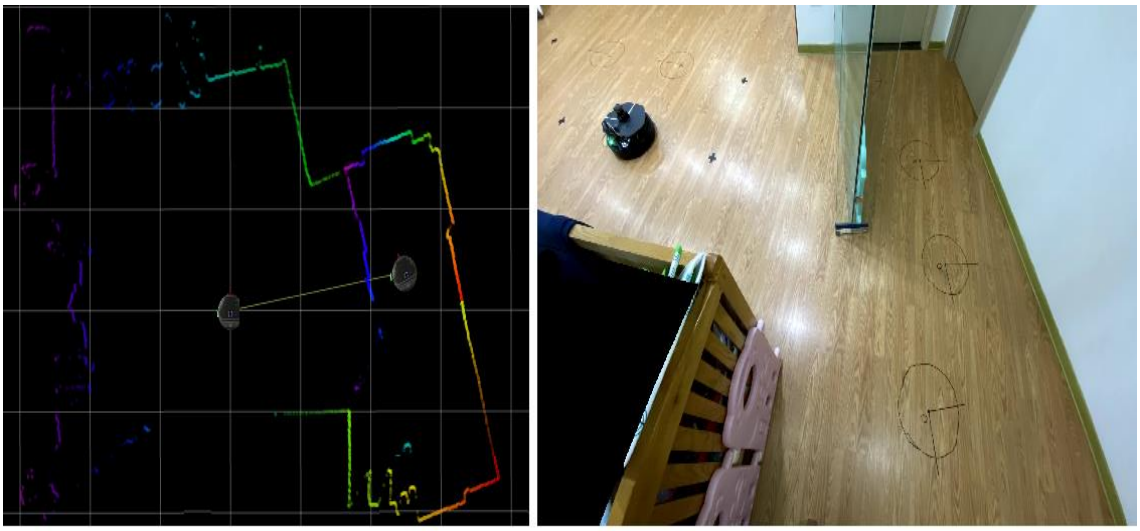
- Kết quả:
  - Sai số vị trí trung bình là 25,81 mm; phương sai 181,40; độ lệch chuẩn 13,47mm
  - Sai số góc trung bình là $0,041^0$; phương sai 0,001; độ lệch chuẩn $0,024^0$

Bảng 3.4.3 Kết quả xác định vị trí robot trong địa hình hành lang dài

	Theo thuật toán			Theo thực tế			Sai số	
	x (mm)	y (mm)	θ (độ)	x (mm)	y (mm)	θ (độ)	Vị trí (mm)	Góc (độ)
Lần 1	621	5	0,028	600	0	0	21,59	0,0280
Lần 2	618	7	0,042	600	0	0	19,31	0,0420
Lần 3	615	3	0,006	600	0	0	15,30	0,0060
Lần 4	591	-1	0,012	600	0	0	9,06	0,0120
Lần 5	1241	9	0,068	1200	0	0	41,98	0,0680
Lần 6	1225	17	0,046	1200	0	0	30,23	0,0460
Lần 7	1189	4	0,031	1200	0	0	11,70	0,0310
Lần 8	1852	11	0,073	1800	0	0	53,15	0,0730
Lần 9	1831	18	0,025	1800	0	0	35,85	0,0250
Lần 10	1819	6	0,081	1800	0	0	19,92	0,0810
Sai số trung bình							25,81	**0,041**
Phương sai							181,40	**0,001**
Độ lệch chuẩn							13,47	**0,024**

Thí nghiệm 4: Xác định vị trí robot trong môi trường có phòng thông nhau

- Mục đích : Kiểm chứng sự chính xác của thuật toán trong điều kiện môi trường có phòng thông nhau (các dữ liệu quét thay đổi rất nhiều)
- Thí nghiệm được thực hiện theo quy trình sau:
    - Đặt robot vào môi trường có 2 phòng thông nhau
    - Robot xuất phát tại vị trí gốc A ở phòng thứ nhất, điều khiển robot xoay 360 độ để quét xung quanh
    - Điều khiển từ xa để robot di chuyển đến phòng thứ 2 rồi sau đó lại quay về phòng thứ nhất, tại vị trí bắt đầu (trên thực tế là robot không thay đổi vị trí)
    - Thuật toán trả về khoảng cách giữa vị trí hiện tại và vị trí xuất phát
    - Lặp lại 10 lần quy trình trên



Hình 3.4.10 Xác định vị trí robot trong môi trường có phòng thông nhau:ảnh quét laser (bên trái) và ảnh thật (bên phải)

- Kết quả:
  - Sai số vị trí trung bình là 101,22 mm, phương sai 640,75; độ lệch chuẩn 25,31mm
  - Sai số góc trung bình là 1,8040, phương sai 0,967; độ lệch chuẩn 0,980

Bảng 3.4.4 Kết quả xác định vị trí robot trong môi trường có phòng thông nhau

	Theo thuật toán			Sai số	
	x (mm)	y (mm)	θ (độ)	Vị trí (mm)	Góc (độ)
Lần 1	112	5	-0.454	112.11	0.454
Lần 2	-134	-36	-1.998	138.75	1.998
Lần 3	-114	-58	-2.667	127.91	2.667
Lần 4	104	10	-1.203	104.48	1.203
Lần 5	71	6	-3.638	71.25	3.638
Lần 6	102	45	1.151	111.49	1.151
Lần 7	-121	-21	-0.643	122.81	0.643
Lần 8	92	-8	-1.332	92.35	1.332
Lần 9	57	19	2.942	60.08	2.942
Lần 10	-46	54	2.017	70.94	2.017
Sai số trung bình				**101.22**	**1.8045**
Phương sai				**640.75**	**0.9667**
Độ lệch chuẩn				**25.31**	**0.9832**

3.4.3 Đánh giá

Qua các thí nghiệm đã thực hiện, ta tính được trung bình sai số vị trí và sai số góc như sau:

Bảng 3.4.5 Kết quả trung bình sai số vị trí và sai số góc của thuật toán

	Sai số vị trí (mm)			Sai số góc (độ)		
	Sai số trung bình	Phương sai	Độ lệch chuẩn	Sai số trung bình	Phương sai	Độ lệch chuẩn
Thí nghiệm 1	18.42	33.7387	5.8027	0.58	0.0876	0.2394
Thí nghiệm 2	24.84	38.1232	6.1030	0.11	0.0518	0.1902
Thí nghiệm 3	25.81	181.4033	13.4686	0.04	0.0006	0.0244
Thí nghiệm 4	101.22	640.7538	25.3131	1.80	0.9667	0.9832
Trung bình	**42.573**	**223.505**	**12.672**	**0.632**	**0.277**	**0.359**

Như vậy qua các thí nghiệm kiểm chứng thực hiện trong nhà, với nhiều điều kiện môi trường khác nhau như tiết diện thẳng, tiết diện gồ ghề, hành lang dài, phòng thông nhau ta có sai số trung bình về vị trí là 42,6 mm độ lệch chuẩn 12,67mm và sai số góc là $0,6^0$ độ lệch chuẩn $0,36^0$

CHƯƠNG 4. KẾT LUẬN

Trong luận văn này, tôi đã tiến hành:

- Tìm hiểu các công nghệ xác định vị trí robot trong nhà bao gồm Vision, Infrared, Wireless Local Area Network (WLAN), RFID, Bluetooth và Laser range finder
- Tìm hiểu các phương pháp xác định vị trí gồm: phương pháp dẫn đường dự đoán (dead-reckoning), hệ thống dẫn đường cột mốc chủ động, hệ thống dẫn đường cột mốc thụ động và định vị sử dụng bản đồ cục bộ
- Tìm hiểu thuật toán Scan Matching với một số phương pháp tiếp cận như IDC, ICP, PL-ICP
- Tìm hiểu hệ điều hành robot ROS và các package liên quan đến cảm biến laser rangder finder, robot Kobuki, thuật toán laser scan matcher
- Thử nghiệm và đánh giá để chứng minh rằng thuật toán Scan Matching đã đề xuất để xác định vị trí cho robot trong nhà là hoàn toàn khả thi và đầy hứa hẹn.

Dựa trên các nghiên cứu, thực nghiệm và đánh giá, tôi đã rút ra được một số kết luận như sau:

- So sánh với một số công nghệ xác định vị trí trong nhà công nghệ Laser range finder có độ phức tạp thấp, chi phí rẻ, độ chính xác chấp nhận được; phù hợp trở thành lựa chọn hàng đầu trong việc xác định vị trí cho các thiết bị robot trong nhà
- Trong số các phương pháp xác định vị trí thì phương pháp định vị sử dụng bản đồ cục bộ có ưu điểm có thể áp dụng linh động không cần thiết lập trước các cột mốc, không bị ảnh hưởng bởi các sai số từ môi trường như gió, vật cản, bề mặt di chuyển không bằng phẳng hay sai lệch giữa thiết kế và thực tế của động cơ, sai lệch giữa các động cơ trong cùng một robot, …
- Thuật toán biến thể PL ICP (ICP with point-to-line metric) dựa trên tính toán từ Điểm tới đường thẳng so với thuật toán IDC và ICP thực hiện ít vòng lặp, thời gian xử lý trung bình thấp cũng như độ chính xác vượt trội so với các thuật toán còn lại.

- Tiến hành thử nghiệm thực tế thuật toán Scan Matching PL ICP trên hệ điều hành ROS sử dụng đế robot Kobuki để di chuyển, cảm biến laser Hokuyo UTM-30LX thu thập dữ liệu trong điều kiện môi trường tiết diện thẳng, tiết diện gồ ghề, hành lang dài, phòng có nhiều phòng thông nhau thu được kết quả với sai số trung bình về vị trí là 42,6 mm độ lệch chuẩn 12,67mm và sai số góc là $0,6^0$ độ lệch chuẩn $0,36^0$ trong thời gian thực. Tuy nhiên do kết cấu và lệnh điều khiển của robot Kobuki chỉ phù hợp để di chuyển ở nơi có địa hình bằng phẳng nên phạm vi thí nghiệm còn hạn chế

Tôi tin rằng với kết quả đã thử nghiệm thực tế này, hoàn toàn có thể xây dựng một hệ thống robot dựa trên cảm biến laser và thuật toán Scan Matching để thực hiện các chức năng như mang đồ vật, hút bụi, giám sát an ninh tòa nhà, robot di chuyển và hoạt động trong các môi trường con người không thể tiếp cận như khu vực cách ly, kho lạnh, hầm lò, …

TÀI LIỆU THAM KHẢO

[1] L.Chen; E. Wu; G. Chen, "Intelligent Fusion Of Wi-fi And Inertialsensor-Based Positioning Systems For Indoor Pedestrian Navigation," in *Ieeesensors j., vol PP, Issue 99*, June 2014.

[2] S.Fang, C. Wang, T. Huang, C. Yang, Y. Chen, "An Enhanced Zigbee Indoorpositioning System With An Ensemble Approach," in *IEEE Communications Letters*, Apr 2012.

[3] L.catarinucci, R. Colella, M. De Blasi, V. Mighali, l. Patrono, l. Tarricone, "High Performance Rfid Tags For Item-Level Tracing Systems," in *proc. Ofinternational Conference On Software, Telecommunications And Computer Networks(Softcom)*, Dubrovnik, Sept 2010.

[4] Y.Chen, D. Lymberopoulos, J. Liu, B. Priyantha, "FM-based IndoorLocalization," in *10th Int. Conf. Mobile System Applications, And Services(MobiSys)*, New York, USA, June 2012.

[5] R. Zhang, F. Höflinger, L. Reindl, "Inertial Sensor Based Indorrr Localization And Monitoring System For EmergencyResponders," in *IEEE Sensors J*, Feb 2013.

[6] Y. Liu, M. Dashti, M. A. A. Rahman, J. Zhang, "Indoor Localization Using Smartphone Inertial Sensors," in *11thWorkshop Positioning, Navigation and Comm. (WPNC)*, Dresden, March 2014.

[7] W. Sakperea, M. Adeyeye-Oshinb and N. B. Mlitwac, "A state-of-the-art Survey Of Indoorpositioning And Navigation Systems Andtechnologies," SACJ 29(3), December 2017.

[8] D. Hahnel, W. Burgard, D. Fox and S. Thrun, "An Efficient FastSLAM Algorithm for Generating Maps of Large-Scale Cyclic Environments From Raw Laser Range Measurements," in *IEEE/RSJ International Conference On Intelligent Robots and Systems (IROS)*, Las Vegas, USA, 2003.

[9] A. Censi, "An ICP Variant Using A Point-to-line Metric," in *the IEEE International Conference On Robotics And Automation (ICRA)*, Pasadena, CA, May 2008.

[10] "Wiki ROS," [Online]. Available: http://wiki.ros.org/.