

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG

HUỲNH XUÂN NGHĨA
LÝ NGỌC TRÂN CHÂU

LUẬN VĂN TỐT NGHIỆP
ỨNG DỤNG THUẬT TOÁN SLAM VÀ THỊ GIÁC MÁY CHO
ROBOT TỰ HÀNH TRONG NHÀ
KỸ SƯ NGÀNH KỸ THUẬT ĐIỀU KHIỂN & TỰ ĐỘNG HÓA

TP. HỒ CHÍ MINH, 2021

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG

HUỲNH XUÂN NGHĨA – 1710202

LÝ NGỌC TRÂN CHÂU – 1710027

LUẬN VĂN TỐT NGHIỆP

ỨNG DỤNG THUẬT TOÁN SLAM VÀ THỊ GIÁC MÁY CHO
ROBOT TỰ HÀNH TRONG NHÀ

APPLICATION OF SLAM ALGORITHM AND COMPUTER
VISION FOR AUTONOMOUS INDOOR ROBOT

KỸ SƯ NGÀNH KỸ THUẬT ĐIỀU KHIỂN & TỰ ĐỘNG HÓA

GIÁNG VIÊN HƯỚNG DẪN

TS. NGUYỄN VĨNH HẢO

TP. HỒ CHÍ MINH, 2021

BỘ MÔN: ĐIỀU KHIỂN TỰ ĐỘNG

TP. HCM, ngày....tháng.....năm.....

**NHẬN XÉT LUẬN VĂN TỐT NGHIỆP
CỦA CÁN BỘ HƯỚNG DẪN**

Tên luận văn:

**ỨNG DỤNG THUẬT TOÁN SLAM VÀ THỊ GIÁC MÁY CHO ROBOT TỰ HÀNH
TRONG NHÀ**

Nhóm Sinh viên thực hiện:

Huỳnh Xuân Nghĩa

1710202 TS. Nguyễn Vĩnh Hảo

Lý Ngọc Trân Châu

1710027 TS. Nguyễn Vĩnh Hảo

Cán bộ hướng dẫn:

Đánh giá Luận văn

1. Về cuốn báo cáo:

Số trang

Số chương

Số bảng số liệu

Số hình vẽ

Số tài liệu tham khảo

Sản phẩm

Một số nhận xét về hình thức cuốn báo cáo:

2. Về nội dung luận văn:

3. Về tính ứng dụng:

4. Về thái độ làm việc của sinh viên:

Đánh giá chung: Luận văn đạt/không đạt yêu cầu của một luận văn tốt nghiệp kỹ sư, xếp loại
Giỏi/ Khá/ Trung bình

Điểm từng sinh viên:

Huỳnh Xuân Nghĩa:...../10

Lý Ngọc Trân Châu:...../10

Cán bộ hướng dẫn
(Ký tên và ghi rõ họ tên)

BỘ MÔN: ĐIỀU KHIỂN TỰ ĐỘNG

TP. HCM, ngày....tháng.....năm.....

**NHẬN XÉT LUẬN VĂN TỐT NGHIỆP
CỦA CÁN BỘ PHẢN BIỆN**

Tên luận văn:

**ỨNG DỤNG THUẬT TOÁN SLAM VÀ THỊ GIÁC MÁY CHO ROBOT TỰ HÀNH
TRONG NHÀ**

Nhóm Sinh viên thực hiện:

Huỳnh Xuân Nghĩa

1710202 TS. Phạm Việt Cường

Lý Ngọc Trân Châu

1710027 TS. Phạm Việt Cường

Cán bộ phản biện:

5. Về cuốn báo cáo:

Số trang _____ Số chương _____

Số bảng số liệu _____ Số hình vẽ _____

Số tài liệu tham khảo _____ Sản phẩm _____

Một số nhận xét về hình thức cuốn báo cáo:

6. Về nội dung luận văn:

7. Về tính ứng dụng:

8. Về thái độ làm việc của sinh viên:

Đánh giá chung: Luận văn đạt/không đạt yêu cầu của một luận văn tốt nghiệp kỹ sư, xếp loại
Giỏi/ Khá/ Trung bình

Điểm từng sinh viên:

Huỳnh Xuân Nghĩa:...../10

Lý Ngọc Trân Châu:...../10

Người nhận xét
(Ký tên và ghi rõ họ tên)

BỘ MÔN: ĐIỀU KHIỂN TỰ ĐỘNG

TP. HCM, ngày 31 tháng 07 năm 2021

ĐỀ CƯƠNG CHI TIẾT

TÊN LUẬN VĂN:
Cán bộ hướng dẫn: TS. NGUYỄN VĨNH HẢO
Thời gian thực hiện: Từ ngày 22/02 đến ngày 30/06
Sinh viên thực hiện: Huỳnh Xuân Nghĩa – 1710202 Lý Ngọc Trân Châu – 1710027
Nội dung đề tài: <ul style="list-style-type: none">- Nghiên cứu xây dựng ứng dụng trên nền tảng ROS cho robot- Thiết kế mô hình xe tự hành cho môi trường trong nhà- Nghiên cứu các thuật toán SLAM, định vị, điều hướng cho robot tự hành- Ứng dụng cảm biến Lidar và Camera cho robot xây dựng bản đồ và tự hành- Ứng dụng thị giác máy để robot bám theo người
Kế hoạch thực hiện: Các công việc của nhóm được hai thành viên phân chia như sau:

Giai đoạn	Huỳnh Xuân Nghĩa	Lý Ngọc Trân Châu
Giai đoạn 1	<ul style="list-style-type: none"> - Vẽ mô hình robot trên SolidWorks và đặt cắt laser. - Tạo mô hình mô phỏng trên ROS. 	<ul style="list-style-type: none"> - Viết code đọc cảm biến IMU, Encoder và điều khiển động cơ trên vi điều khiển.
Giai đoạn 2	<ul style="list-style-type: none"> - Tính toán vấn đề công suất và lắp ráp robot. 	<ul style="list-style-type: none"> - Lắp ráp robot.
Giai đoạn 3	<ul style="list-style-type: none"> - Viết package trên ROS điều khiển robot qua rosserial. 	<ul style="list-style-type: none"> - Viết code vi điều khiển giao tiếp với máy tính nhúng.
Giai đoạn 4	<ul style="list-style-type: none"> - Viết package giao tiếp RPLidar và Camera. 	<ul style="list-style-type: none"> - Calib magnetometer trên Matlab. - Calib gyroscope, accelerometer. - Viết code cho bộ lọc Madgwick, và tính toán Odometry của robot trên vi điều khiển.
Giai đoạn 5	<ul style="list-style-type: none"> - Sử dụng package rtabmap_ros vẽ bản đồ với thuật toán RTAB-Map. Đánh giá hạn chế và khắc phục. 	<ul style="list-style-type: none"> - Đánh giá, lựa chọn thuật toán nhận diện người và các thuật toán phục vụ cho tác vụ bám theo người
Giai đoạn 6	<ul style="list-style-type: none"> - Thực hiện định vị robot trong bản đồ dùng bộ lọc AMCL. Đánh giá hạn chế và tìm cách khắc phục. 	<ul style="list-style-type: none"> - Xây dựng thuật toán bám theo người.
Giai đoạn 7	<ul style="list-style-type: none"> - Thực hiện điều hướng robot trong bản đồ dựa trên navigation stack, đánh giá hạn chế và tìm cách khắc phục. 	
Giai đoạn 8	<ul style="list-style-type: none"> - Dùng thuật toán scan_matcher trong package laser_scan_matcher để xác định odometry của robot trong bản đồ, thay thế odometry tính từ IMU và Encoder. 	<ul style="list-style-type: none"> - Thu thập và đánh giá kết quả bám theo người, chỉnh sửa thuật toán. - Đóng gói tác vụ bám theo người thành package trên ROS và kết hợp với tác vụ định vị.
Giai đoạn 9	<ul style="list-style-type: none"> - Thu thập và đánh giá kết quả vẽ bản đồ, định vị và điều hướng robot, giải thích hạn chế. 	<ul style="list-style-type: none"> - Hỗ trợ thu thập kết quả vẽ bản đồ, định vị và điều hướng.
Giai đoạn 10	<ul style="list-style-type: none"> - Tiến hành viết báo cáo. 	<ul style="list-style-type: none"> - Tiến hành viết báo cáo.

Xác nhận của Cán bộ hướng dẫn (Ký tên và ghi rõ họ tên)	TP. HCM, ngày 31 tháng 07 năm 2021 Sinh viên (Ký tên và ghi rõ họ tên) Huỳnh Xuân Nghĩa Lý Ngọc Trần Châu
---	---

DANH SÁCH HỘI ĐỒNG BẢO VỆ LUẬN VĂN

Hội đồng chấm luận văn tốt nghiệp, thành lập theo Quyết định số ngày của Hiệu trưởng Trường Đại học Bách khoa TP.HCM.

1. TS. Phạm Việt Cường..... – Chủ tịch.
2. ThS. Trần Hoàng Khôi Nguyên..... – Thư ký.
3. ThS. Nguyễn Thanh Tâm..... – Ủy viên.
4. Ông Trường Triều Thuận..... – Ủy viên.

LỜI CẢM ƠN

Lời đầu tiên xin gửi lời tri ân đến gia đình, là điểm tựa tinh thần vững chắc trong suốt thời gian chúng con học tập tại trường, cũng là nguồn động lực to lớn giúp chúng con có thêm nghị lực vượt qua khó khăn, vươn lên trong học tập, có thêm niềm tin và vững bước trên con đường theo đuổi tri thức.

Tiếp theo, chúng em xin gửi lời cảm ơn chân thành đến thầy Nguyễn Vĩnh Hảo, người đã tận tình hướng dẫn chúng em thực hiện luận văn. Thầy không chỉ là điểm tựa tinh thần, giải đáp các vấn đề chuyên ngành và đưa ra những lời khuyên hết sức quý giá, mà còn tạo điều kiện vật chất, môi trường làm việc nghiên cứu thuận lợi nhất giúp nhóm hoàn thành luận văn với những mục tiêu đã đề ra.

Chúng em cũng chân thành cảm ơn các thầy cô Khoa Điện-Điện Tử, đặc biệt là những thầy cô trong bộ môn Điều Khiển và Tự Động Hóa. Các thầy cô luôn hết sức giảng dạy cho sinh viên một cách chu đáo, truyền đạt kiến thức nền tảng đến chuyên sâu cũng như những kinh nghiệm thực tế của bản thân, góp phần giúp chúng em nắm vững và áp dụng kiến thức vào luận văn, tạo động lực học tập nghiên cứu lâu dài.

Sau cùng, nhóm gửi lời cảm ơn đến các bạn cùng chuyên ngành, đặc biệt là các anh chị, các bạn ở phòng thí nghiệm 207B3 đã nhiệt tình giúp đỡ và đóng góp ý kiến để nhóm hoàn thành luận văn một cách hoàn thiện.

TP. Hồ Chí Minh, ngày 31 tháng 07 năm 2021

Sinh viên

MỤC LỤC

LỜI CẢM ƠN.....	8
MỤC LỤC	9
DANH MỤC HÌNH ẢNH.....	11
DANH MỤC BẢNG	14
DANH MỤC TỪ VIẾT TẮT	16
TÓM TẮT LUẬN VĂN.....	1
ABSTRACT	2
CHƯƠNG 1. GIỚI THIỆU	3
1.1 Tổng quan đề tài.....	3
1.2 Mục tiêu đề tài.....	4
1.3 Phương pháp thực hiện	5
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT	6
2.1. Nền tảng ROS	6
2.2. Thuật toán RTAB-Map	11
2.3. Thuật toán định vị AMCL (Adaptive Monte-Carlo Localization)	14
2.4. Thuật toán Scan Matching	18
2.5. Thuật toán điều hướng robot.....	19
2.6. Mô hình MobileNet SSD	23
2.7. Kmean-Clustering	30
2.8. Hiệu chỉnh Magnetometer.....	31
2.9. Kết hợp cảm biến bằng bộ lọc Madgwick	33
2.10. Điều khiển tốc độ động cơ.....	36
CHƯƠNG 3. NỘI DUNG THỰC HIỆN	39
3.1. Thiết kế và thi công mô hình	39
3.2. Xây dựng phần mềm trên mạch điều khiển	49
3.3. Đọc dữ liệu cảm biến trên nền tảng ROS	57
3.4. Tác vụ điều khiển robot trên ROS	63
3.5. Tính toán odometry từ dữ liệu Lidar.....	65
3.6. Tác vụ vẽ bản đồ	65

3.7. Tác vụ định vị robot trong bản đồ.....	68
3.8. Tác vụ điều hướng robot.....	72
3.9. Thuật toán bám theo người (Human Following)	82
CHƯƠNG 4. KẾT QUẢ THỰC HIỆN.....	91
4.1. Kết quả xây dựng mô hình robot	91
4.2. Kết quả điều khiển tốc độ động cơ có tải.....	92
4.3. Kết quả tính toán odometry.....	94
4.4. Kết quả vẽ bản đồ với RTAB-Map.....	95
4.5. Kết quả định vị robot trong bản đồ	100
4.6. Kết quả điều hướng robot	101
4.7. Kết quả thuật toán bám theo người	102
CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	110
5.1. Kết luận	110
5.2. Hướng phát triển	112
PHỤ LỤC	114
PHỤ LỤC I	114
PHỤ LỤC II	119
TÀI LIỆU THAM KHẢO	121

DANH MỤC HÌNH ẢNH

Hình 1.1. Tổng quan hệ thống đề tài luận văn sử dụng	4
Hình 2.1 Cấu trúc File System trên ROS	6
Hình 2.2 Truyền nhận message trong ROS	7
Hình 2.3 Service trong ROS	7
Hình 2.4 Computational Graph Level	7
Hình 2.5 Vai trò của ROS Master	8
Hình 2.6 Trao đổi dữ liệu giữa các Node qua Topic trên ROS	8
Hình 2.7 Cấu trúc tầng ROS Community	9
Hình 2.8 Cấu trúc TF của một Robot	9
Hình 2.9 Giao tiếp giữa rosserial client và server	10
Hình 2.10 Giới hạn giao tiếp khác nhau cho các dòng vi điều khiển	11
Hình 2.11 Định dạng Packet trong Rosserial	11
Hình 2.12 Sơ đồ của node rtabmap – node chính trong package rtabmap_ros	12
Hình 2.13 Sơ đồ khôi tạo Occupancy Grid Map trong RTAB-Map	14
Hình 2.14. Hàm bel(x) thể hiện nhận thức của robot về vị trí của mình	15
Hình 2.15. Không gian tìm kiếm vận tốc của thuật toán DWA	22
Hình 2.16. Góc lệch mục tiêu θ	23
Hình 2.17 Kết quả của một lớp feature map sau khi qua một bộ lọc dự đoán	24
Hình 2.18 Kiến trúc mạng SSD	25
Hình 2.19 Kết quả huấn luyện mô hình SSD trên tập dữ liệu VOC2007 so với các mô hình khác	27
Hình 2.20 Ứng dụng sử dụng MobileNet SSD cho tác vụ Object Detection	27
Hình 2.21 Sự kết hợp của depthwise convolution và pointwise convolution	28
Hình 2.22 Kiến trúc mạng MobileNet	29
Hình 2.23 So sánh MobileNet với các mô hình khác khi là base network trong một mạng Object Detection huấn luyện trên tập COCO	30
Hình 2.24 Độ từ thiên và độ từ khuynh của từ trường trái đất	32
Hình 2.25 Mặt cầu từ trường	32
Hình 2.26 Mô hình Differential Drive	36
Hình 2.27 Sơ đồ khôi Lowpass Filter	37
Hình 2.28 Sơ đồ khôi bộ điều khiển PID	38
Hình 2.29 Ảnh hưởng của các hệ số Kp, Ki, Kd lên bộ điều khiển PID	38
Hình 3.1 RPLIDAR A1	39
Hình 3.2 Các Packet truyền khi LiDAR quét 360°	39
Hình 3.3 Camera Realsense Intel D435	40
Hình 3.4 Hình ảnh có độ sâu thu được từ Realsense	40
Hình 3.5. Cảm biến MPU9250	41
Hình 3.6 NVIDIA Jetson Nano	41

Hình 3.7 Kit phát triển STM32F407 Discovery board.....	42
Hình 3.8 Driver cầu H HI216 Nhật Bản.....	42
Hình 3.9 Động cơ DCM50-775 12V DC	43
Hình 3.10 Sơ đồ khối mạch điều khiển	43
Hình 3.11 Sơ đồ khối máy tính nhúng.....	44
Hình 3.12 Bản vẽ schematic của mạch shield cho board Discovery.....	45
Hình 3.13 Bản in PCB của mạch shield cho board Discovery.....	45
Hình 3.14. Mô hình robot thiết kế dùng SolidWorks	46
Hình 3.15. Trang cấu hình thông số để tạo file urdf từ mô hình thiết kế	47
Hình 3.16. Mô hình quan sát trong phần mềm Rviz	48
Hình 3.17 Quỹ tích các mẫu giá trị từ trường trước khi hiệu chỉnh	49
Hình 3.18 Quỹ tích các mẫu giá trị từ trường trên mặt phẳng Oxy trước khi hiệu chỉnh	50
Hình 3.19 Quỹ tích các mẫu giá trị từ trường sau khi hiệu chỉnh	50
Hình 3.20 Quỹ tích các mẫu giá trị từ trường trên mặt phẳng Oxy sau khi hiệu chỉnh	51
Hình 3.21 Quỹ tích các mẫu giá trị từ trường kiểm thử trên mặt phẳng Oxy sau khi hiệu chỉnh	51
Hình 3.22 Hướng của robot so với n-frame	52
Hình 3.23 Ảnh hướng đối với accelerometer khi không nằm tại tâm xoay robot.....	53
Hình 3.24 Sơ đồ khối kết hợp cảm biến	53
Hình 3.25 Robot trong O-frame	54
Hình 3.26 Sơ đồ khối điều khiển vận tốc hai bánh xe.....	56
Hình 3.27. Lưu đồ giải thuật để đọc cảm biến Lidar.....	58
Hình 3.28. Lưu đồ giải thuật đọc cảm biến Camera trên nền tảng ROS	60
Hình 3.29. Khi chạy package điều khiển vận tốc trên ROS	64
Hình 3.30. Sự liên kết giữa các node thông qua topic trong tác vụ vẽ bản đồ	67
Hình 3.31. File hình ảnh và file thông số của bản đồ 2D được vẽ	68
Hình 3.32. Quá trình robot thực hiện tự định vị trên bản đồ với sự quan sát trên Rviz	71
Hình 3.33. Mô hình Navigation stack trong ROS	72
Hình 3.34. Kết quả khi tải lên bản đồ đã vẽ	73
Hình 3.35. Kết quả điều hướng robot khi không có vật cản và quan sát trên Rviz.....	79
Hình 3.36. Kết quả điều hướng robot khi có vật cản tĩnh và quan sát trên Rviz	80
Hình 3.37. Kết quả điều hướng robot khi có vật cản động và quan sát trên Rviz.....	81
Hình 3.38 Vấn đề occlusion trong Human Tracking.....	82
Hình 3.39 Cách tính toán IOU giữa hai bounding box.....	85
Hình 3.40 Bounding box chứa người cần trích xuất đặc trưng	86
Hình 3.41 Bounding box sau khi đã được crop để dùng trích xuất đặc trưng.....	86
Hình 3.42 Sơ đồ khối trích xuất đặc trưng của ảnh trong bounding box	87
Hình 3.43 Sơ đồ giải thuật tóm tắt của thuật toán bám theo người	88
Hình 4.1. Mô hình robot thực tế	91

Hình 4.2 Đồ thị đáp ứng điều khiển tốc độ hai động cơ có tải bằng PID (đơn vị trực x: s – y: m/s)	92
Hình 4.3. Đồ thị đáp ứng điều khiển tốc độ hai động cơ có tải bằng PID với $r(t)$ ở nhiều mức khác nhau (đơn vị trực x: s – y: m/s)	93
Hình 4.4 Đồ thị đáp ứng điều khiển tốc độ hai động cơ có tải bằng PID với $r(t)$ luân phiên đảo dấu (đơn vị trực x: s – y: m/s)	93
Hình 4.5. Môi trường vẽ bản đồ và thực hiện các tác vụ là phòng thí nghiệm 207B3 ĐH Bách Khoa TP.HCM	96
Hình 4.6. Môi trường vẽ bản đồ và thực hiện các tác vụ là một căn phòng	97
Hình 4.7. Kết quả vẽ bản đồ 2D tại phòng thí nghiệm 207B3 ĐH Bách Khoa TP.HCM	98
Hình 4.8. Kết quả vẽ bản đồ 2D tại căn phòng trọ và kích thước thật của nó	98
Hình 4.9. Kết quả vẽ bản đồ 3D tại căn phòng trọ	99
Hình 4.10 Ảnh được lấy trích đặc trưng tham chiếu	103
Hình 4.11 Ba màu center của đặc trưng tham chiếu	103
Hình 4.12 Ảnh người được nhận diện là target person	104
Hình 4.13 Ba màu center đặc trưng của người được nhận diện là target person	104
Hình 4.14 Ảnh người không được nhận diện là target person	104
Hình 4.15 Ba màu center đặc trưng của người không được nhận diện là target person	105
Hình 4.16 Ảnh người được nhận diện sai thành target person	105
Hình 4.17 Ba màu center đặc trưng của người được nhận diện sai thành target person	105
Hình 4.18 Đồ thị thể hiện khả năng bám theo target person mỗi khung ảnh trong 1600 khung	108
Hình 4.19 Đồ thị thể hiện khả năng bám theo target person mỗi khung ảnh trong 400 khung	108
Hình 0.1 Mô tả sai số thiết bị	115
Hình 0.2 Mô tả nhiễu từ trường	116

DANH MỤC BẢNG

Bảng 2.1. Thuật toán amcl dùng để định vị robot	18
Bảng 2.2. Thuật toán tìm đường đi ngắn nhất Dijkstra	20
Bảng 3.1. Bảng thông tin kết nối các thành phần trong mô hình robot.....	48
Bảng 3.2 Bảng thông số chọn cho bộ lọc Madgwick	53
Bảng 3.3 Bảng thông tin tín hiệu bộ điều khiển PID	55
Bảng 3.4 Bảng giá trị thông số điều khiển tốc độ động cơ	56
Bảng 3.5 Bảng giá trị chu kỳ thực hiện tác vụ trong chương trình vi điều khiển	57
Bảng 3.6. Bảng giá trị thông số cấu hình cho giao tiếp giữa ROS và Lidar	58
Bảng 3.7. Bảng thông tin về dữ liệu của Lidar trong ROS	59
Bảng 3.8. Một số thông số cấu hình camera thông qua ROS	60
Bảng 3.9. Một số thông số cấu hình camera thông qua ROS (tiếp theo)	61
Bảng 3.10. Các topic giao tiếp giữa mạch điều khiển và ROS thông qua rosserial	62
Bảng 3.11. Bảng thông số cấu hình điều khiển vận tốc trên ROS	64
Bảng 3.12. Bảng thông tin ROS của package điều khiển vận tốc	64
Bảng 3.13. Một số thông số cấu hình RTAB-Map.....	65
Bảng 3.14. Một số thông số cấu hình RTAB-Map (tiếp theo)	66
Bảng 3.15. Ngõ vào và ngõ ra của package rtabmap_ros	66
Bảng 3.16. Ngõ vào và ngõ ra của package amcl.....	68
Bảng 3.17. Các thông số cấu hình bộ lọc AMCL.....	69
Bảng 3.18. Một số thông số cấu hình tác vụ định vị trong điều hướng robot	74
Bảng 3.19. Thông số cấu hình cơ bản của move_base.....	74
Bảng 3.20. Một số thông số cấu hình chung của cosmap	75
Bảng 3.21. Thông số cấu hình Costmap toàn cục	75
Bảng 3.22. Một số thông số cấu hình Local Costmap	76
Bảng 3.23. Một số thông số cấu hình Local Planner.....	77
Bảng 3.24 Bảng thông số threshold (ngưỡng) cho thuật toán bám theo người.....	90
Bảng 4.1. Thông số của mô hình robot	91
Bảng 4.2 Bảng kết quả điều khiển tốc độ động cơ có tải bằng PID	92
Bảng 4.3 Kết quả đo odometry thông qua IMU và Encoder	94
Bảng 4.4. Kết quả đo odometry thông qua dữ liệu từ Lidar.....	95
Bảng 4.5. Kết quả kích thước bản đồ 2D vẽ được tại căn phòng trọ	100
Bảng 4.6. Kết quả định vị của robot	101
Bảng 4.7. Kết quả điều hướng robot.....	102
Bảng 4.8 Kết quả nhận diện target person ở 60 lần dự đoán (target person:1/ không phải target person: 0)	107
Bảng 4.9 Đánh giá kết quả thuật toán bám theo người	108
Bảng 0.1 Thông số kỹ thuật RPLidar A1	119
Bảng 0.2 Thông số kỹ thuật Camera Realsense D435	119

Bảng 0.3 Thông số kỹ thuật Jetson Nano	119
Bảng 0.4 Bảng thông số kỹ thuật MPU9250	119
Bảng 0.5 Thông số kỹ thuật vi điều khiển STM32F407VG	119
Bảng 0.6 Thông số kỹ thuật cầu H HI216	119
Bảng 0.7 Thông số kỹ thuật động cơ DCM50-775	120

DANH MỤC TỪ VIỆT TẮT

ROS	Robot Operating System
SLAM	Simultaneously Localization and Mapping
DWA	Dynamic Window Approach
CNN	Convolutional Neural Network
SSD	Single Shot Detector
FPS	Frame per second
mAP	Mean Average Precision
IOU	Intersection over Union
ReLUs	Rectified Linear Units
RMS	Root Mean Square
GPU	Graphics Processing Unit

TÓM TẮT LUẬN VĂN

Đề tài này thực hiện thiết kế thi công phần cứng và thiết kế xây dựng phần mềm cho robot tự hành môi trường trong nhà. Robot thực hiện các tác vụ như: vẽ bản đồ, định vị, điều hướng, tránh vật cản và bám theo người. Bộ não của robot là một máy tính nhúng Jetson Nano, xử lý thuật toán RTAB-Map và thuật toán bám người trên hệ điều hành ROS. Các cảm biến được sử dụng cho robot là Lidar 2D, Depth Camera, IMU và Encoder. Dựa vào các dữ liệu cảm biến này mà bản đồ 2D hoặc 3D có thể được xây dựng bởi thuật toán RTAB-Map, từ đó thực hiện định vị và điều hướng cho robot. Thuật toán AMCL được sử dụng để định vị robot trong bản đồ, cùng với đó là thuật toán Dijkstra và Dynamic Window Approach giúp robot quy hoạch quỹ đạo, tìm đường đi tối ưu và tránh vật cản. Thuật toán bám người được vận hành riêng biệt với các tác vụ khác, sử dụng Depth Camera và được xây dựng dựa trên các mô hình CNN và các thuật toán máy học. Máy tính nhúng giao tiếp với bảng mạch nhúng có vi điều khiển STM32F407VG để nhận dữ liệu Odometry từ Encoder và IMU, cũng như gửi tín hiệu điều khiển robot. Dữ liệu bản đồ vẽ trên máy tính nhúng có thể được hiển thị trên PC bất kỳ có cài đặt hệ điều hành ROS và phần mềm RVIZ thông qua chuẩn UDP không dây. UDP cũng giúp gửi các lệnh điều khiển trên PC xuống Jetson để hỗ trợ quá trình vận hành robot.

Odometry sử dụng cho việc vẽ bản đồ được lấy từ một trong hai nguồn cảm biến: Lidar; hoặc IMU và Encoder. Kết quả thực hiện khi sử dụng hai nguồn trên được so sánh và đánh giá về khả năng cũng như giới hạn.

Sai số của việc vẽ bản đồ 2D vào bé hơn 5%, sai số định vị và điều hướng lần lượt vào khoảng 6cm và 4cm đối với tọa độ (x, y), vào khoảng 10° và 8° đối với góc yaw. Nhóm không tìm được cách đánh giá tốt hơn cho tác vụ bám theo người như được nêu ở mục 4.7 theo đó, tác vụ có F1-score đạt 77.922%. Vì giới hạn về khả năng xử lý của máy tính nhúng, tần số bản đồ được cập nhật vào khoảng 4 Hz, tác vụ bám người có thể được xử lý với tốc độ 7 FPS. Để đảm bảo vận hành tốt nhất từ đánh giá thực tế, robot được giới hạn tốc độ di chuyển trong khoảng [-0.3 m/s; 0.3 m/s], step size là 0.075m/s đối với vận tốc tịnh tiến và trong khoảng [-0.5 rad/s; 0.5 rad/s], step size là 0.075 rad/s đối với vận tốc góc.

ABSTRACT

This thesis project designs and builds a mobile robot for indoor environment. Main tasks of this robot include: mapping, localization, navigation, obstacles avoidance and human following. It has a brain which is Jetson Nano – an embedded PC to process such algorithms as RTAB-Map and the human following one on ROS. Lidar 2D, Depth Camera, IMU and Encoder are sensors used inside the robot. Thanks to data from these sensors, an indoor map (2D or 3D) can be built using RTAB-Map and then used to do localization and navigation. AMCL algorithm is used to localize the robot in the map. Dijkstra algorithm and Dynamic Window Approach are used to navigate the robot with optimized trajectory as well as help it avoid obstacles. The human following task is operated separately with the others. It uses a Depth Camera and is designed based on CNN models and Machine learning algorithms. In the purpose of receiving Odometry data from IMU and Encoder, as well as sending controlling commands to the robot, Jetson Nano interacts with an embedded board with the heart of microcontroller STM32F407VG. The map built on Jetson Nano can be visualized on any PC which runs ROS and has RVIZ installed through a wireless UDP standard. UDP standard also helps to send commands from PC to Jetson to support robot's operation.

Robot's odometry for mapping is received from either of the two sensors data: Lidar; or IMU in combination with Encoder. The two results are then benchmarked to point out each one's pros and cons.

The error of the 2D mapping is smaller than 5%, the errors of localization and navigation task are respectively around 6cm and 4cm for coordinates (x, y), around 10° and 8° in terms of yaw angle. The performance of the human following task is hard to analyzed but as detailed in mục 4.7, it has the F1-score of 77.922%. We have not been able to find a better way to evaluate it yet. Depending on the processing and calculating limit of Jetson Nano, the frequency of updating the map is about 4Hz, and the human following task can be processed with a speed of 7FPS. To ensure the best operation of the robot according to experiences, it can have speeds within range of [-0.3 m/s; 0.3 m/s], step size of 0.075 m/s in terms of linear velocity, and range of [-0.5 rad/s; 0.5 rad/s], step size of 0.075 rad/s when it comes to angle velocity.

CHƯƠNG 1. GIỚI THIỆU

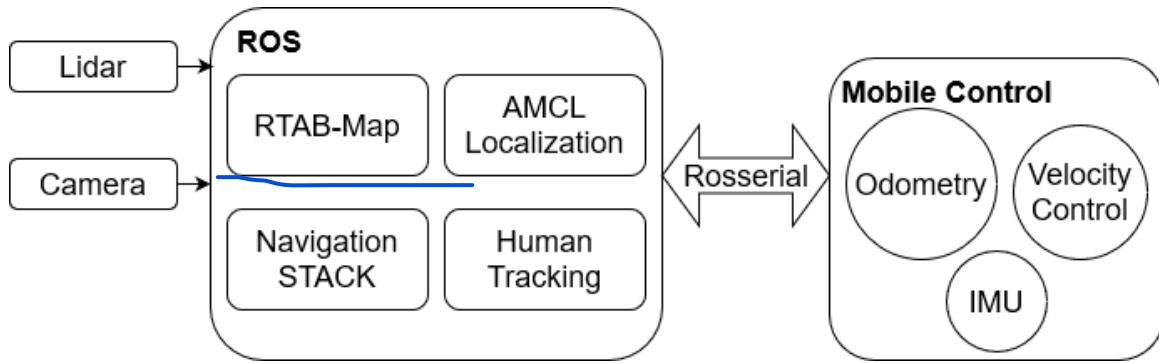
1.1 Tổng quan đề tài nghiên cứu

Hiện nay, công nghệ liên quan đến robot di động (Mobile Robot) như phương tiện tự hành (autonomous vehicles) hoặc robot phục vụ trong nhà đang được đưa rất nhiều vào trong nghiên cứu. Một thành phần thiết yếu trong ứng dụng robot là hệ thống điều hướng (navigation), nó giúp robot nhận dạng và tái tạo môi trường để việc di chuyển trở nên hiệu quả hơn. Một robot tự hành cần đáp ứng được các yêu cầu tất yếu: thứ nhất, cần phải xác định được vị trí của của bản thân trong hệ tọa độ tham chiếu; thứ hai, robot phải lập cho mình một kế hoạch tìm đường và tranh vật cản tự động; cuối cùng là robot phải nhận biết được môi trường xung quanh thông qua các cảm biến (nhận thức). SLAM (Simultaneous Localization and Mapping) là một hệ thống định vị và lập bản đồ trực quan thời gian thực giúp robot hoạt động ưu việt hơn.

Nhiều hệ thống SLAM khác nhau đang được nghiên cứu phát triển, hầu hết chúng sử dụng cảm biến quang, nổi bật trong số đó là Visual SLAM (camera-based) sử dụng thông tin ảnh thu được từ camera và Lidar SLAM (lidar-based) sử dụng thông tin laser từ cảm biến lidar. Với mục tiêu hoạt động ở môi trường trong nhà, đề tài nghiên cứu được nhóm lựa chọn sử dụng cảm biến Lidar và Depth Camera phục vụ nhiệm vụ SLAM thông qua thuật toán Real-Time Appearance-Based Mapping (RTAB-Map), một thuật toán mã nguồn mở giúp xây dựng bản đồ 2D và 3D với sự kết hợp của hai loại cảm biến trên. Đồng hành với việc xây dựng bản đồ, nhóm tiếp tục nghiên cứu xây dựng tác vụ định vị và điều hướng sử dụng các thuật toán phổ biến của một robot tự hành và tích hợp vào hệ thống như thuật toán AMCL, thuật toán Dijkstra sẽ được trình bày sau. Robot tự hành bên cạnh việc phải đáp ứng các yêu cầu đặt ra ở trên, một số tính năng cũng được tích hợp vào như khả năng nhận dạng đối tượng và bám theo người.

Để xây dựng một hệ thống hoàn thiện thì đây là một đề tài khá rộng và cần có những chuyên gia có kiến thức, kinh nghiệm để có thể phát triển một hệ thống ổn định, hoạt động trôi chảy trên nhiều điều kiện môi trường khác nhau. Vì tính ứng dụng rộng lớn và tính phức tạp của vấn đề, luận văn chỉ ở mức tìm hiểu thuật toán và ứng dụng các thuật toán cơ bản để có thể xây dựng được mô hình hoạt động hoàn thiện.

1.2 Mục tiêu đề tài



Hình 1.1. Tổng quan hệ thống đề tài luận văn sử dụng

Luận văn thực hiện với mục tiêu kết hợp Lidar-based SLAM và Visual-based SLAM thông qua thuật toán RTAB-Map, được hỗ trợ phát triển bởi ROS. Hình 1.1 thể hiện tổng quan hệ thống cần xây dựng của đề tài. ROS sẽ đọc cảm biến và xử lý dữ liệu thông tin về môi trường và người dùng. Các khối trong ROS có chức năng nhận dữ liệu từ ROS để xử lý cho ra điều ra mong muốn. Cùng với đó, chuẩn giao tiếp rosserial liên kết giữa phần xử lý trung tâm và phần điều khiển để nhận dữ liệu về odometry cũng như gửi tín hiệu điều khiển vận tốc xuống để robot di chuyển. Từ đó, nhóm đặt ra các mục tiêu cần hoàn thành gồm:

- Hiểu và nắm bắt thuật toán RTAB-Map để vẽ bản đồ, thuật toán AMCL định vị robot, thuật toán Dijkstra tìm đường tối ưu và thuật toán Dynamic Window Algorithm tránh vật cản tự động.
- Xây dựng mô hình robot nhận tín hiệu điều khiển vận tốc và thực thi đúng với yêu cầu đặt ra.
- Xây dựng bản đồ 2D và 3D của môi trường trong nhà chưa biết trước thông qua sự điều khiển của người dùng hoặc khả năng tự tìm đường đi.
- Định vị robot trong bản đồ đã vẽ.
- Thực hiện quy hoạch quỹ đạo tìm đường đi tối ưu cho robot để vị trí mong muốn trên bảng đồ và tránh vật cản tự động trong quá trình di chuyển.
- Ứng dụng thị giác máy tính phục vụ tác vụ nhận diện đối tượng và bám theo đối tượng.

1.3 Phương pháp thực hiện

Dựa vào môi trường hoạt động của robot là ở trong nhà nên nhóm tự thiết kế và lắp ráp khung robot với động cơ, nguồn, vi điều khiển, cảm biến, ... sao cho khung robot chịu được tải trọng của robot và có hình dáng giúp robot dễ dàng di chuyển trong nhà. Quá trình thiết kế khung xe cùng mạch điều khiển và cảm biến được thực hiện qua thông các phần mềm hỗ trợ, trình bày ở mục 3.1. Với các nhiệm vụ đặt ra, nhóm tiến hành tìm hiểu lý thuyết của thuật toán RTAB-Map cùng các thuật toán cơ bản của robot tư hành gồm bộ lọc AMCL, thuật toán Dijkstra, thuật toán Dynamic Window Algorithm, song song cùng quá trình tìm hiểu mô hình Mobilenet, trình bày ở CHƯƠNG 2 và bắt đầu thực hiện. Việc thiết kế phần mềm cho mạch điều khiển bao gồm đọc và xử lý dữ liệu cảm biến IMU cùng Encoder để tính toán odometry của robot, đồng thời tích hợp giao thức rosserial để giao tiếp với nền tảng ROS, trình bày ở mục 3.2. Robot sử dụng máy tính nhúng JetsonNano là bộ xử lý trung tâm, hoạt động với hệ điều hành Ubuntu có tích hợp ROS. Dựa trên đó, nhóm tiến hành thiết kế phần mềm xử lý chính trên ROS gồm việc đọc cảm biến, ứng dụng RTAB-Map, định vị và điều hướng robot cùng các tác vụ cơ bản như điều khiển từ xa, trình bày từ mục 3.3 đến mục 3.6. Tác vụ bám theo người được thực hiện song song với quá trình trên, sau đó được trình hợp vào ROS, trình bày ở mục 3.9. Cuối cùng, nhóm thu thập kết quả tổng kết ở CHƯƠNG 4, từ đó đánh giá quá trình thực hiện và tìm hiểu khắc phục các nhược điểm.

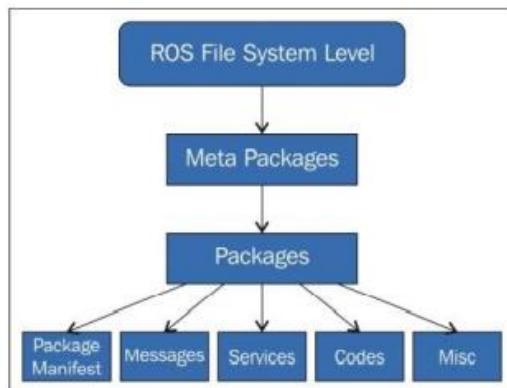
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Nền tảng ROS

ROS hiện nay được biết đến là hệ điều hành cực kì phổ biến trong lĩnh vực robot. Mặc dù chưa hẳn là một hệ điều hành hoàn chỉnh mà chính xác hơn là platform, nhưng ROS có các yếu tố cấu thành một hệ điều hành: trừu tượng hóa phần cứng, chuyển thông tin giữa các tiến trình, quản lý dữ liệu theo các gói, ... Mô hình ROS được khái quát như sau.

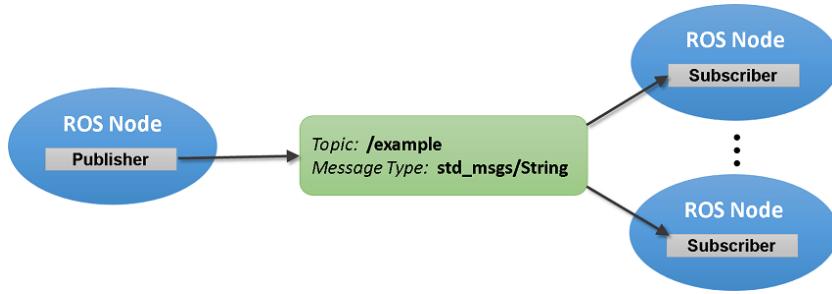
2.1.1. Tầng ROS File System

Filesystem chủ yếu là các tài nguyên trên phần cứng, cấu trúc thư mục và các tập tin tối thiểu để ROS hoạt động.



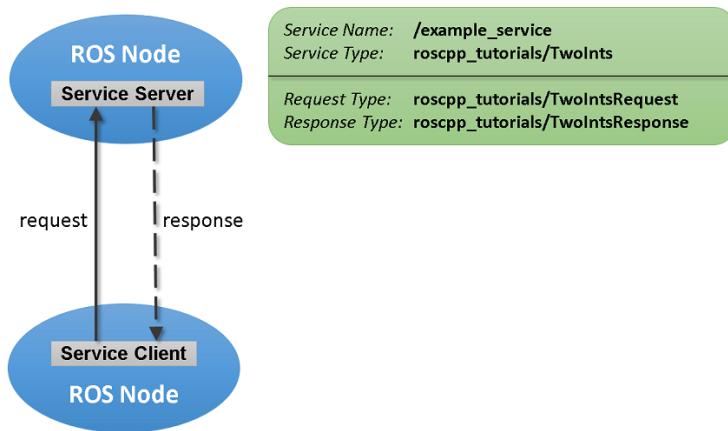
Hình 2.1 Cấu trúc File System trên ROS

- Packages: ROS Package là thành phần cơ bản nhất trên ROS. Nó bao gồm các node, thư viện, và các tập tin cấu hình, ... được sắp xếp với nhau thành một thành phần duy nhất. Mục đích của gói mã nguồn là tạo ra chương trình tổng quát nhất để dễ dàng sử dụng lại cũng như phát triển.
- Message: chứa cấu trúc thông tin theo một dạng cố định để giao tiếp giữa các node. Việc truyền/nhận messages dựa theo cơ chế publish/subscribe. Node Publisher sẽ gửi yêu cầu lấy thông tin từ node Subscriber, sau khi kết nối giữa các node thành công, quá trình gửi và nhận sẽ diễn ra liên tục.



Hình 2.2 Truyền nhận message trong ROS

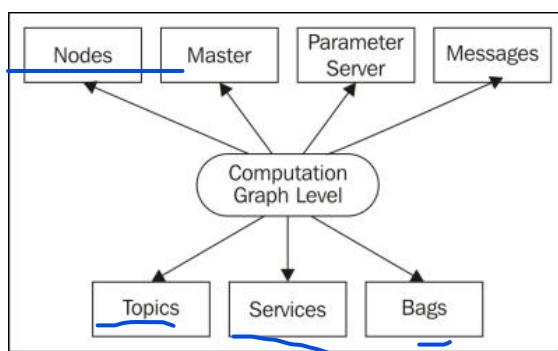
- Services: tương tự như messages, services được dùng để trao đổi thông tin giữa các node. Tuy nhiên services hoạt động theo cơ chế request/respond.



Hình 2.3 Service trong ROS

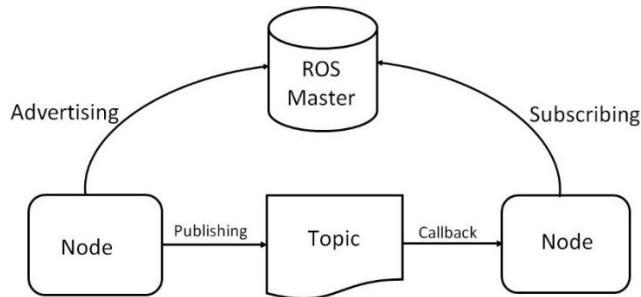
2.1.2. Tầng ROS Computation Graph

Computation Graph là nơi mà các quy trình trong ROS được kết nối với nhau. Tất cả các node trong hệ thống đều có thể truy cập vào đây để tương tác, thực hiện trao đổi dữ liệu nằm trong mạng. Tầng này bao gồm các khái niệm cơ bản sau:



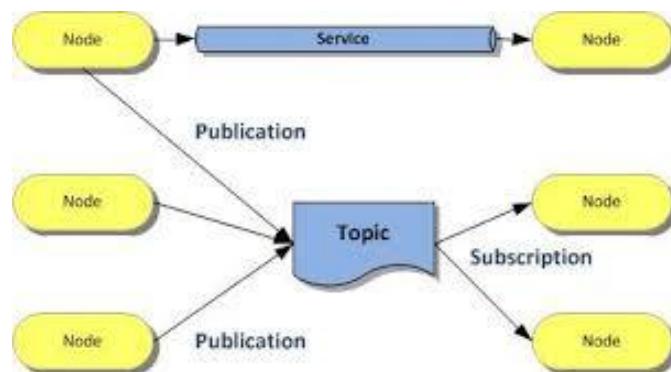
Hình 2.4 Computational Graph Level

- Node: đơn vị thực thi cơ bản nhất. Mỗi node thực hiện một nhiệm vụ cố định là xử lý và truyền/nhận dữ liệu, nhờ vậy các tác vụ lớn, cần nhiều tính toán dễ dàng thực hiện bằng việc liên kết các node lại với nhau.
- Master: đóng vai trò kết nối các node với nhau. Do đó, master luôn được khởi động đầu tiên bằng câu lệnh roscore sau đó thì bạn có thể gọi bất kỳ các node nào trong hệ thống.



Hình 2.5 Vai trò của ROS Master

- Message: Đây là cấu trúc dữ liệu được các node sử dụng để trao đổi với nhau tương tự như các kiểu dữ liệu double, int trong các ngôn ngữ lập trình. Các node tương tác với nhau bằng cách send và receive ROS message.
- Topics: là phương pháp giao tiếp trao đổi dữ liệu giữa hai node, nó bao gồm nhiều cấp bậc thông tin mà chúng có thể giao tiếp thông qua ROS message. Hai phương thức trong topic bao gồm publish và subscribe.



Hình 2.6 Trao đổi dữ liệu giữa các Node qua Topic trên ROS

2.1.3. Tầng ROS Community

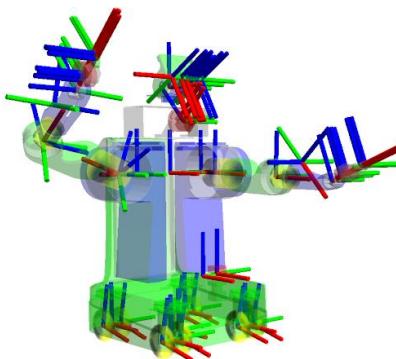
ROS Community là nguồn tài nguyên của ROS được cộng đồng người dùng trao đổi với nhau về phần mềm và kiến thức.



Hình 2.7 Cấu trúc tầng ROS Community

2.1.4. Package TF trong ROS

TF là một package giúp người dùng quản lý được nhiều khung tọa độ qua thời gian. TF giữ được mối quan hệ giữa các khung tọa độ trong một cấu trúc dạng cây (tree structure) được đếm vào theo thời gian, cho phép người dùng chuyển đổi (transform) các điểm, vector, ... giữa 2 khung tọa độ bất kỳ thời điểm nào. Robot được trang bị nhiều cảm biến, việc tìm ra mối quan hệ vị trí giữa cảm biến so với robot là rất quan trọng cho việc phân tích ý nghĩa của dữ liệu sau này. Thư viện TF được ROS cung cấp giúp người dùng dễ dàng định nghĩa mô hình vật lý cho robot.



Hình 2.8 Cấu trúc TF của một Robot

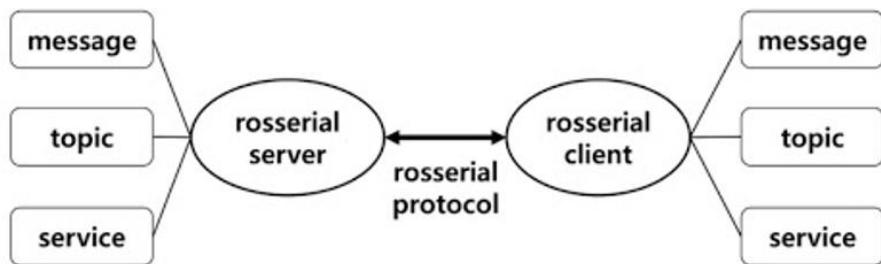
2.1.5. Các phần mềm mô phỏng trong ROS

RVIZ và Gazebo là hai phần mềm nổi tiếng nhất trong ROS. Đây là một giao diện đồ họa của ROS, cung cấp khả năng mô phỏng chính xác và hiệu quả robot trong môi trường phức tạp cả trong nhà lẫn ngoài trời, cho phép hiển thị dữ liệu 3 chiều của các loại cảm biến hoặc các robot sử dụng phương thức mô tả mô hình vật lý Unified Robot Description Format (URDF) và SDF.

2.1.6. Kết nối Rosserial

Hệ thống nhúng trong đè tài giao tiếp với ROS thông qua giao thức rosserial. Đây là phiên bản giao tiếp nối tiếp point-point (P2P) của ROS giúp người dùng kết hợp được ROS và các loại vi điều khiển thông thường được sử dụng (Arduino, STM32, Tiva, ...). Rosserial bao gồm giao thức và thư viện dùng trên vi điều khiển, và nodes để dùng cho phía PC. Ngoài các định nghĩa về giao thức, có ba loại package được đề cập trong Rosserial:

- Thư viện Client: cho phép người dùng dễ dàng kích hoạt các node trên ROS và chạy trên nhiều hệ thống khác nhau. Các package hiện tại gồm rosserial_arduino, rosserial_tivac, rosserial_stm32, ...
- ROS-side Interface: thiết bị chạy mã nguồn rosserial cần yêu cầu một node chạy trên máy chủ (host machine) để bắt kết nối từ giao thức nối tiếp đến mạng lưới ROS.
- Ngoài ra, còn có các package về mở rộng và ví dụ cho việc sử dụng rosserial.



Hình 2.9 Giao tiếp giữa rosserial client và server

Số lượng Publisher và Subscriber được giới hạn trong 25, và kích thước bộ đệm được giới hạn bởi 512 bytes mặc định cho rosserial_client. Tuy nhiên, các số liệu trên có thể bị giảm đi tùy theo giới hạn SRAM của các vi điều khiển.

AVR Model	Input/Output buffer sizes	Publishers/Subscribers
ATMEGA168	150/150 bytes	6/6
ATMEGA328P	280/280 bytes	25/25
All others	512/512 bytes	25/25

Hình 2.10 Giới hạn giao tiếp khác nhau cho các dòng vi điều khiển

Rosserial dùng cách ghép chuỗi và tách chuỗi giống nhau cho các message ROS chuẩn, đơn giản là thêm header và footer, cho phép nhiều topic cùng chia sẻ một kết nối nối tiếp.

```

1st Byte - Sync Flag (Value: 0xff)
2nd Byte - Sync Flag / Protocol version
3rd Byte - Message Length (N) - Low Byte
4th Byte - Message Length (N) - High Byte
5th Byte - Checksum over message length
6th Byte - Topic ID - Low Byte
7th Byte - Topic ID - High Byte
x Bytes - Serialized Message Data
Byte x+1 - Checksum over Topic ID and Message Data

```

Hình 2.11 Định dạng Packet trong Rosserial

2.2. Thuật toán RTAB-Map

2.2.1. Giới thiệu SLAM

Cũng như con người chúng ta, robot cũng cần sự trợ giúp từ bản đồ cho việc di chuyển có tri thức. Các nghiên cứu tiên phong về bài toán SLAM (Simultaneous Localization and Mapping) được thực hiện vào cuối những năm 1980s, đầu những năm 1990s. Bài toán SLAM yêu cầu robot tự xây dựng được bản đồ khi chúng di chuyển trong môi trường, giúp chúng xác định được vị trí của mình trong môi trường thông qua một quy trình nhiều bước bao gồm căn chỉnh dữ liệu từ các cảm biến đầu vào (sensor data alignment), ước lượng sự di chuyển (motion estimation), tính toán odometry, ...

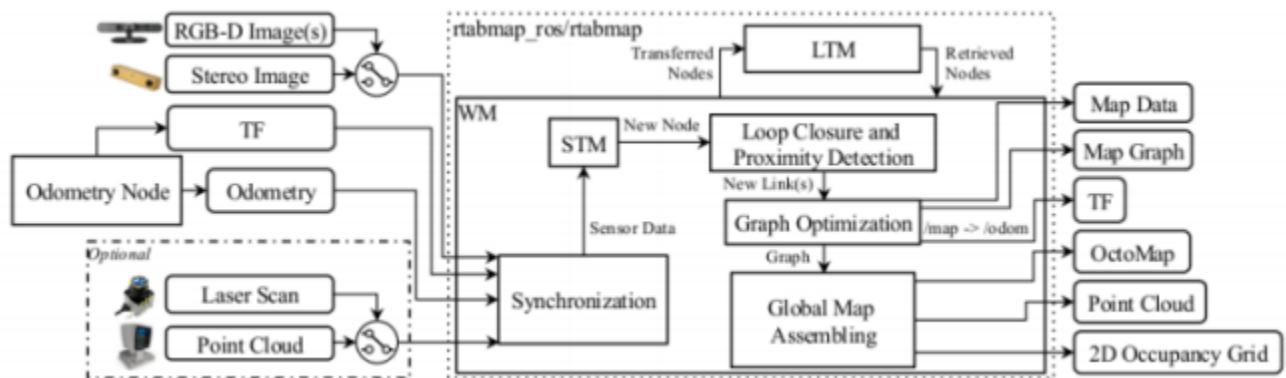
Có rất nhiều thuật toán SLAM mã nguồn mở có mặt trên ROS với hai loại chính: **Visual-based SLAM** với dữ liệu đầu vào là camera và **Lidar-based SLAM** với dữ liệu đầu vào là **Lidar 2D hoặc 3D**.

2.2.2. Mapping với RTAB-Map

Tổng quan

Real-Time Appearance-Based Mapping (RTAB-Map) [1] là thuật toán mã nguồn mở về SLAM. Nhờ có cơ chế kiểm tra vòng kín (loop closure detection) và quản lý bộ nhớ (memory management) mà việc giải bài toán SLAM trên môi trường rộng lớn trong thời gian dài vẫn đảm bảo được độ chính xác và tính thời gian thực. RTAB-Map được tạo ra để tích hợp cả hai hướng, **LiDAR-based** và **Visual-based**, vào hệ thống, cung cấp những công cụ thuận lợi cho việc so sánh khi thực hiện trên các loại robot khác nhau, sử dụng các loại cảm biến khác nhau, ... Được nghiên cứu và phát triển liên tục, hiện nay RTAB-Map đã đáp ứng được khá nhiều yêu cầu thực tế: **Xử lý online, giảm sai số trôi Odometry, tăng khả năng định vị, thực hiện SLAM nhiều lần liên tiếp, giải quyết vấn đề điểm khởi đầu**

RTAB-Map thực hiện SLAM theo hướng **graph-based** và được tích hợp thành một gói mã nguồn trên ROS với tên "**rtabmap_ros**".



Hình 2.12 Sơ đồ của node **rtabmap** – **node chính** trong package **rtabmap_ros**

Odometry Node

Odometry có thể được tính từ dữ liệu của những cảm biến đơn giản như encoder, IMU, ... cho đến phức tạp như LiDAR, camera. RTAB-Map không quan tâm cảm biến nào được sử dụng mà chỉ yêu cầu có ít nhất một loại cảm biến có thể cung cấp thông tin cho việc tính odometry. TF chứa thông tin về phép biến đổi tọa độ từ robot sang camera nhằm biết vị trí của camera trên robot.

Khối đồng bộ

Bởi vì các cảm biến không gửi dữ liệu cùng tần số, cũng như thời gian thường không trùng khớp với nhau, do đó việc đồng bộ thông tin là rất quan trọng. ROS cung cấp hai loại đồng bộ là chính xác và xấp xỉ. Đồng bộ chính xác yêu cầu đầu vào của các topic phải có cùng thời gian (timestamp), chẳng hạn hình ảnh bên trái và phải của stereo camera. Đồng bộ xấp xỉ không yêu cầu thời gian chính xác tuyệt đối mà chỉ cần độ trễ không vượt quá mức cho phép.

Khối STM

Short-Term Memory (STM) là nơi dữ liệu thu được từ cảm biến **được đóng gói** và đưa vào các điểm trong bản đồ. STM cập nhật trọng số cho các điểm dựa vào những đặc trưng giống nhau trích xuất từ hình ảnh. Sau đó STM quyết định xem điểm nào sẽ được đưa vào World Map (WM), điểm nào nên được xóa đi. Thông tin về vị trí hiện tại, ảnh màu, ảnh độ sâu, laser cũng sẽ được lưu trữ. Kích thước STM phụ thuộc vào tốc độ robot di chuyển và tốc độ thêm điểm vào bản đồ.

Khối kiểm tra vòng kín (Loop Closure Detection)

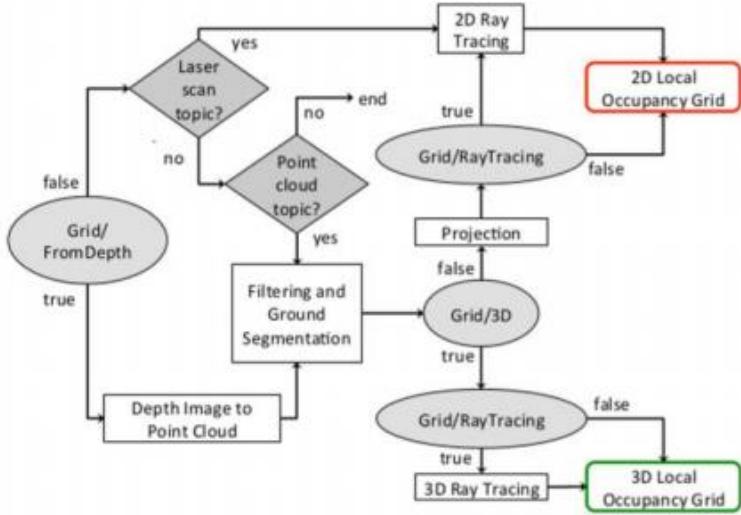
Nếu hình ảnh giữa hai điểm liên tiếp giống nhau, xem xét dựa theo tỉ lệ trùng khớp giữa những đặc trưng của mỗi ảnh, vượt qua mức ngưỡng, trọng số của điểm trước đó sẽ tăng lên 1 đơn vị. Nếu xác định robot không di chuyển dựa vào odometry, điểm vừa tạo ra sẽ được xóa đi.

Khối phát hiện vật cản (Proximity Detection)

Không giống với phát hiện vòng kín, quá trình phụ thuộc rất nhiều vào các điểm trong WM, phát hiện vật cản phụ thuộc đa số vào các điểm ở gần robot. Những điểm đó phải ở gần robot và số liên kết giữa chúng phải nhỏ hơn mức ngưỡng cho trước.

Khối lắp ghép bản đồ

Khi node mới được tạo ra trong STM, một local occupancy grid sẽ được tính toán từ hình ảnh có độ sâu, laser scan hoặc point cloud. Một local occupancy grid là tương quan với khung robot và nó chứa các ô trống, ô có ground và ô có vật cản với một kích thước cụ thể.



Hình 2.13 Sơ đồ khái tạo Occupancy Grid Map trong RTAB-Map

Phụ thuộc vào thông số Grid/FromDepth, Grid/3D và các topic đầu vào, local occupancy grid được tạo ra có thể khác nhau và ở dạng 2D hoặc 3D. Bản đồ 2D tốn ít bộ nhớ hơn bản đồ 3D và không thể được dùng để tạo ra global occupancy grid 3D, nhưng bản đồ 3D lại có thể tạo ra global occupancy grid 2D. Sự lựa chọn còn tùy thuộc vào yêu cầu của hệ thống và khả năng xử lý. Cuối cùng, đầu ra cuối cùng là một global map sẽ được kết hợp từ các local occupancy grid.

2.3. Thuật toán định vị AMCL (Adaptive Monte-Carlo Localization)

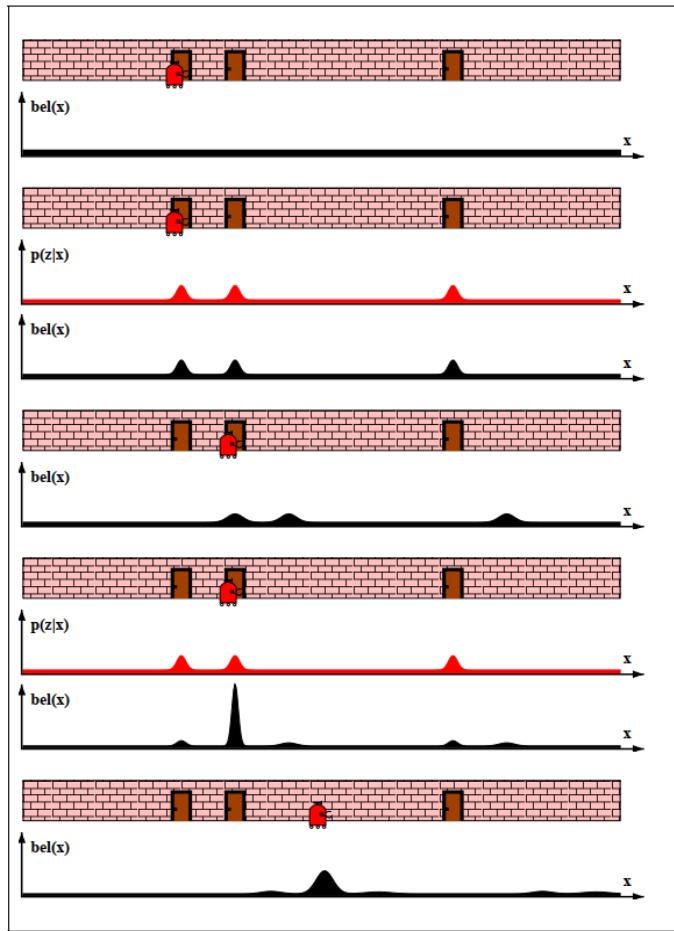
2.3.1. Localization với AMCL

Mô hình chuyển động [2] của robot diễn tả xác suất biến đổi trạng thái theo tín hiệu điều khiển $p(x_t | x_{t-1}, u_t)$, là bước dự đoán trạng thái trong các bộ lọc. Giả định robot di chuyển trên mặt phẳng, vector trạng thái của robot sẽ được xác định bởi vị trí của robot và hướng của nó tại vị trí đó như sau:

$$X = (x, y, \theta)^T$$

Tại thời điểm t , không thể dùng một vector để diễn tả trạng thái của robot do ảnh hưởng của nhiều yếu tố. Thay vào đó, trạng thái của robot sẽ được diễn tả bằng một phân phối xác suất là một tập các trạng thái mà có khả năng vị trí robot hiện tại ở đó.

Mô hình quan sát [3] của robot diễn tả quá trình xử lý dữ liệu từ cảm biến Lidar, là một phân phối xác suất có điều kiện $p(z_t | x_t, m)$ với z_t là dữ liệu cảm biến, x_t là trạng thái của particle và m là bản đồ môi trường đã vẽ. Xác suất này được hiểu là trọng số của particle ứng với phép đo của cảm biến tại thời điểm t .



Hình 2.14. Hàm $bel(x)$ thể hiện nhận thức của robot về vị trí của mình

2.3.2. Particle filter

Bộ lọc không tham số particle filter (PF) [4] sử dụng một số hữu hạn các particle (mẫu trạng thái) để đại diện cho một phân phối xác suất $bel(x)$ thể hiện nhận thức của chính robot về trạng thái của mình trong không gian.

Khi bộ lọc PF dùng M particle thì tập hợp particle $X = \{x_t^1, x_t^2, \dots, x_t^M\}$ với mỗi particle $x_t^i (1 \leq i \leq M)$ là một vị trí trong môi trường thực tế mà robot có thể đang ở đó. Bộ lọc PF

dùng ngõ vào là tập trạng thái ở thời điểm $t-1$, tín hiệu điều khiển u_{t-1} và tín hiệu đo z_t , ngõ ra là tập trạng thái ở thời điểm t . Các bước thực hiện của bộ lọc PF gồm:

Bước 1: Tạo tập particle rỗng:

$$\dot{X}_t = X_t = \emptyset$$

Bước 2: Dự đoán vị trí robot dựa trên vị trí trước đó và tín hiệu điều khiển. Với mỗi particle từ tập trạng thái ở thời điểm $t-1$, thực hiện lấy mẫu (1), tính trọng số cho từng mẫu (2), cập nhật tập hợp particle tạm thời (3):

$$x_t^{[i]} = p(x_t \mid x_{t-1}^{[i]}, u_{t-1}) \quad (1)$$

$$w_t^{[i]} = p(z_t \mid x_t^{[i]}) \quad (2)$$

$$\dot{X}_t = \dot{X}_t + x_t^{[i]}, w_t^{[i]} > \quad (3)$$

Bước 3: Thực hiện cập nhật tập hợp particle X_t bằng việc lấy mẫu $x_t^{[i]}$ từ tập particle tạm thời \dot{X}_t theo phân phối xác suất của trọng số các particle (4). Như vậy, tập particle chính thức chứa các mẫu quan trọng, loại bỏ các particle có trọng số thấp. Thông thường, các trọng số ở thời điểm t sẽ được cập nhật dựa trên trọng số ở thời điểm $t-1$, với giá trị ban đầu bằng 1 (5):

$$X_t = X_t + x_t^{[i]}, w_t^{[i]} > \quad (4)$$

$$w_t^{[i]} = p(z_t \mid x_t^{[i]}) \times w_{t-1}^{[i]} \quad (5)$$

2.3.3. Bộ lọc thích nghi AMCL dùng định vị robot

Bộ lọc Monte-Carlo là một dạng bộ lọc PF dùng để xác định vị trí của robot trong không gian bản đồ đã vẽ. Các bước thực hiện của bộ lọc Monte-Carlo tương tự bộ lọc PF với mô hình chuyển động và mô hình quan sát như trình bày ở mục 2.3.1 và tập particle X_t phân phối theo hàm $bel(x)$. Quá trình thực hiện được lặp lại cho những lần tiếp theo với số mẫu hội tụ và giảm.

Nhược điểm của bộ lọc Monte-Carlo là không thể định vị khi robot bị dời đi một khoảng lớn hơn vùng lân cận hoặc thất bại trong bài toán định vị toàn cục.

Bộ lọc AMCL là cải tiến của bộ lọc Monte-Carlo, giải quyết nhược điểm nêu trên bằng việc thêm vào một số particle ngẫu nhiên vào tập particle đang quan sát khi cần thiết, tức là khi có sai số lớn giữa tập particle và dữ liệu đo đạc thì số lượng particle tăng lên cùng với giá trị trọng số nhỏ lại để tăng khả năng ước lượng đúng vị trí robot, còn khi sai số nhỏ thì số lượng particle giảm đi và giá trị trọng số tăng lên để tăng hiệu quả tính toán của thuật toán. Đây là tính thích nghi của bộ lọc AMCL. Các particle ngẫu nhiên thêm vào trước lúc lấy mẫu và được tính toán dựa trên trọng số ước lượng dài hạn ω_{slow} và trọng số ước lượng ngắn hạn ω_{fast} với ω_{avg} là trung bình trọng số tất cả particles, điều kiện $0 \leq \omega_{slow} \leq \omega_{fast}$:

$$\begin{cases} \omega_{slow} = \omega_{slow} + \alpha_{slow}(\omega_{avg} - \omega_{slow}) \\ \omega_{fast} = \omega_{fast} + \alpha_{fast}(\omega_{avg} - \omega_{fast}) \end{cases}$$

Thuật toán Kullback-Leibler Divergence (KLD) [5] được sử dụng để tái lấy mẫu trong thuật toán AMCL và tính số particle yêu cầu. Thuật toán này có thể tính ra số particle yêu cầu dựa trên phân số của trọng số các particle. Giới hạn trên N_{top} của số particle yêu cầu được tính như sau:

$$N_{top} = \frac{k-1}{2\alpha} \left\{ 1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}} \beta \right\}^3$$

Trong đó, α và β lần lượt là sai số tối đa và lượng tử của phân bố chuẩn hóa giữa phân bố thực và phân số ước lượng.

Tóm lại, thuật toán định vị AMCL được tổng kết thông qua Bảng 2.1 bên dưới:

Bảng 2.1. Thuật toán amcl dùng để định vị robot

```

1. Inputs:  $S_{t-1} = \{\langle x_{t-1}^{(i)}, w_{t-1}^{(i)} \rangle \mid i = 1, \dots, n\}$  representing belief  $Bel(x_{t-1})$ ,  

   control measurement  $u_{t-1}$ , observation  $z_t$ ,  

   bounds  $\varepsilon$  and  $\delta$ , bin size  $\Delta$ , minimum number of samples  $n_{\chi_{min}}$ 
2.  $S_t := \emptyset, n = 0, n_{\chi} = 0, k = 0, \alpha = 0$  // Initialize
3. do // Generate samples ...
   // Resampling: Draw state from previous belief
4. Sample an index  $j$  from the discrete distribution given by the weights in  $S_{t-1}$ 
   // Sampling: Predict next state
5. Sample  $x_t^{(n)}$  from  $p(x_t \mid x_{t-1}, u_{t-1})$  using  $x_{t-1}^{(j)}$  and  $u_{t-1}$ 
6.  $w_t^{(n)} := p(z_t \mid x_t^{(n)})$ ; // Compute importance weight
7.  $\alpha := \alpha + w_t^{(n)}$  // Update normalization factor
8.  $S_t := S_t \cup \{\langle x_t^{(n)}, w_t^{(n)} \rangle\}$  // Insert sample into sample set
9. if  $(x_t^{(n)})$  falls into empty bin  $b$  then
10.    $k := k + 1$  // Update number of bins with support
11.    $b :=$  non-empty // Mark bin
12.   if  $n \geq n_{\chi_{min}}$  then
13.      $n_{\chi} := \frac{k-1}{2\varepsilon} \left\{ 1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)} z_{1-\delta}} \right\}^3$  // Update number of desired samples
14.    $n := n + 1$  // Update number of generated samples
15. while  $(n < n_{\chi}$  and  $n < n_{\chi_{min}})$  // ... until KL-bound is reached
16. for  $i := 1, \dots, n$  do // Normalize importance weights
17.    $w_t^{(i)} := w_t^{(i)} / \alpha$ 
18. return  $S_t$ 

```

2.4. Thuật toán Scan Matching

Đây là thuật toán hỗ trợ robot có trang bị cảm biến lidar, giúp tối ưu hóa sự liên kết giữa các chùm điểm cuối (end-points) trên bản đồ và các chùm điểm cuối vừa mới thu được từ cảm biến lidar sử dụng phương pháp lặp Gaussian-Newton [6].

Với trạng thái ước tính của robot là $\xi = (p_x, p_y, \psi)$, dữ liệu end-points thu được từ cảm biến là $S_i(\xi)$ được dùng so sánh với giá trị end-points bị chiếm trên bản đồ $M(S_i(\xi))$ (0 là trống, 1 là bị chiếm). Sự biến đổi của các end-points thu được từ cảm biến được thể hiện thông qua $S_i(\xi)$:

$$S_i(\xi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} s_{x,i} \\ s_{y,i} \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \end{bmatrix}$$

Công thức được dùng tối ưu sự liên kết của thuật toán như sau:

$$\xi = \underset{\xi}{\operatorname{argmin}} [1 - M(S_i(\xi))]^2$$

Giải thuật lặp Gauss-Newton được sử dụng để giải quyết các bài toán bình phương nhỏ nhất phi tuyến tính, hàm mục tiêu r_i được định nghĩa như sau:

$$r_i = 1 - M(S_i(\xi))$$

Thông qua việc liên tục tính vector Gradient G và ma trận Hessian H của hàm mục tiêu, phương pháp Gauss-Newton sẽ lặp 2 phép tính dưới đây để tính ra ξ và tối thiểu hàm mục tiêu:

$$\begin{cases} \xi_t = \xi_{t-1} - H^{-1}G \\ \Delta\xi = \xi_{t-1} - \xi_t = H^{-1}G \end{cases}$$

2.5. Thuật toán điều hướng robot

2.5.1. Thuật toán tìm đường đi ngắn nhất Dijkstra

Thuật toán Dijkstra [7] là một thuật toán giải quyết bài toán tìm đường đi ngắn nhất nguồn đơn trong một đồ thị có hướng. Thuật toán thực hiện tìm đường đi từ một đỉnh đến tất cả các đỉnh còn lại của đồ thị có trọng số không âm. Thuật toán Dijkstra bình thường sẽ có độ phức tạp là $O(n^2 + m^2)$, trong đó m là số cạnh, n là số đỉnh của đồ thị đang xét. Thuật toán thường được sử dụng trong định tuyến với một chương trình con trong các thuật toán đồ thị hay trong công nghệ hệ thống định vị toàn cầu (GPS).

Thuật toán Dijkstra được trình bày vắn tắt thông qua Bảng 2.2 dưới đây.

Bảng 2.2. Thuật toán tìm đường đi ngắn nhất Dijkstra

```

1. Init() // Khởi tạo các giá trị
2.     dist[v] =  $\infty$  ; p[v] = undefined ; d[s] = 0;
3. Update(u,v) // Cập nhật khoảng cách đến các nodes
4.     if d[v] > d[u] + c(u,v) // c(u,v) là khoảng cách từ u đến v
5.     then d[v] = d[u] + c(u,v) ; p[v] = u ;
6. Main()
7.     Init();
8.     Q = the set of all nodes in Graph
9.     Repeat :
10.        u: u  $\in$  Q | d(u) min;
11.        Q = Q  $\cup$  {u}
12.        for all v  $\in$  neighbor(u) and v  $\notin$  Q
13.            Update(u,v);
14.        Until Q = V

```

Ý tưởng của thuật toán như sau: Đặt d là biến khoảng cách, ta có mảng d[u] là khoảng cách ngắn nhất từ đỉnh s tới đỉnh u trên đồ thị. Ban đầu d[s] = 0, các giá trị khác bằng dương vô cực. Ta sẽ lấy đỉnh u có d[u] nhỏ nhất vào thời điểm hiện tại, và sử dụng khoảng cách của nó để cập nhật khoảng cách ngắn nhất của các đỉnh xung quanh.

2.5.2. Thuật toán Dynamic Window

Thuật toán Dynamic Window Algorithm (DWA) [8] là một thuật toán tránh vật cản động được sử dụng phổ biến cho mobile robot, mục đích dùng để tìm ra tín hiệu điều khiển hợp lý gửi xuống robot để đi đến đích một cách an toàn, không va chạm trong một khoảng không gian nhất định dựa trên một global planner đã hoạch định đường đi toàn cục từ trước. Thuật toán này gồm 2 bước chính là cắt giảm không gian tìm kiếm và lựa chọn vận tốc trong không gian tìm kiếm để tối ưu hàm mục tiêu.

2.5.2.1. Cắt giảm không gian tìm kiếm

Đối với mobile robot, mô hình chuyển động trong thời gian thực yêu cầu việc tính toán rất lớn nên mô hình chuyển động tương đương [9] đã được áp dụng. Đối với mô hình này, chuyển động của robot phụ thuộc vào tập hợp vận tốc tuyến tính v_i và vận tốc xoay ω_i , và quỹ

đạo của robot sẽ là đường thẳng hoặc đường tròn (đường cong) phụ thuộc vào vận tốc xoay ω_i tại mỗi thời điểm.

Tùy thuộc vào vật cản động trong môi trường hoạt động của robot, không gian tìm kiếm của các vận tốc khả thi được cắt giảm thông qua ba bước:

- a) Tập hợp quỹ đạo tròn: Thuật toán DWA chỉ quan tâm đến quỹ đạo tròn (đường cong) duy nhất được xác định bởi biến vận tốc (v, ω) gồm vận tốc tịnh tiến và vận tốc xoay. Các biến vận tốc này được tập hợp lại tạo ra không gian tìm kiếm vận tốc V_s .
- b) Tìm vận tốc cho phép: để đảm bảo quỹ đạo chuyển động là an toàn thông qua các điều kiện. Biến vận tốc (v, ω) được xem là cho phép chỉ khi robot có khả năng dừng lại không va chạm trước khi nó tiến đến vật cản gần nhất trong đường cong quỹ đạo tương ứng. Vận tốc cho phép được định nghĩa như sau:

$$V_a = \{(v, \omega) | v \leq \sqrt{2 \cdot dist(v, \omega) \cdot \dot{v}_b} \wedge \omega \leq \sqrt{2 \cdot dist(v, \omega) \cdot \dot{\omega}_b}\}$$

Trong đó: V_a là tập hợp các biến vận tốc (v, ω) cho phép đảm bảo không va chạm

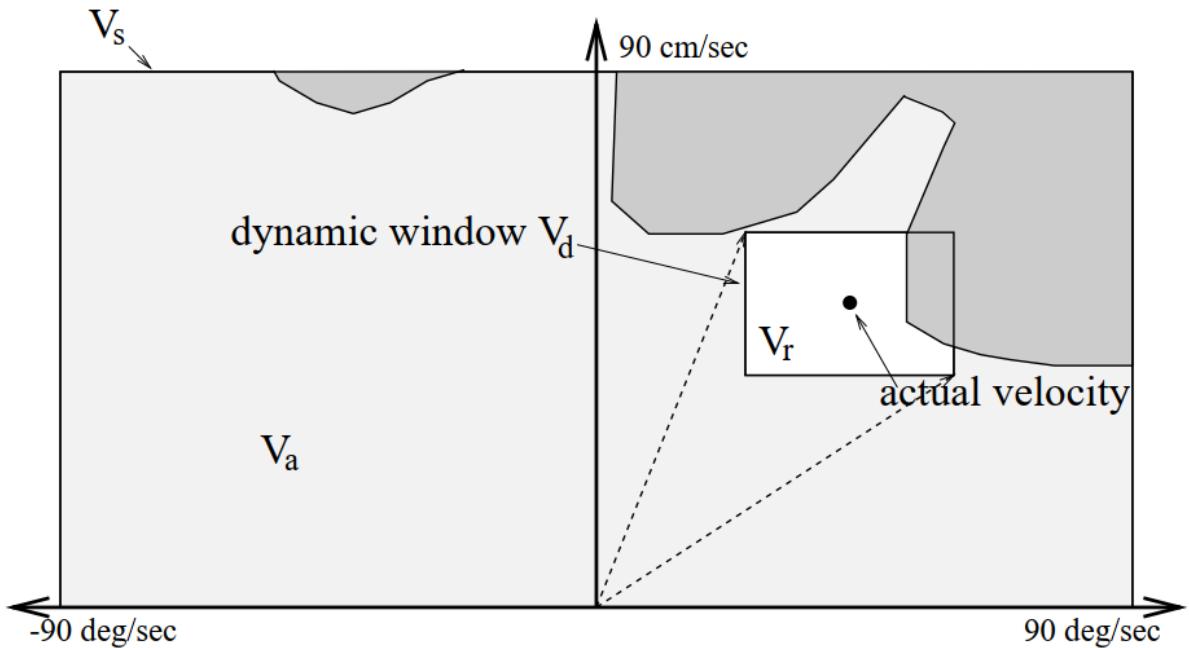
$dist(v, \omega)$ là khoảng cách nhỏ nhất robot dừng trước vật cản để không va chạm $\dot{v}_b, \dot{\omega}_b$ là gia tốc thẳng và gia tốc xoay tối đa nếu robot di chuyển sẽ gây va chạm

- c) Dynamic window: Bước này dùng để cắt giảm vận tốc cho phép thành những vận tốc có thể đạt được trong chu kỳ kế tiếp với gia tốc tối đa cho trước của robot. Việc cắt giảm này được thực hiện thông qua biến vận tốc thực (v_a, ω_a) và gia tốc thực thi tối đa $(\dot{v}, \dot{\omega})$ của robot với biểu thức định nghĩa như sau:

$$V_d = \{(v, \omega) | v \in [v_a - \dot{v} \cdot \Delta t, v_a + \dot{v} \cdot \Delta t] \wedge \omega \in [\omega_a - \dot{\omega} \cdot \Delta t, \omega_a + \dot{\omega} \cdot \Delta t]\}$$

Khi kết thúc ba bước trên thì tìm được không gian tìm kiếm của thuật toán:

$$V_r = V_s \wedge V_a \wedge V_d$$



Hình 2.15. Không gian tìm kiếm vận tốc của thuật toán DWA

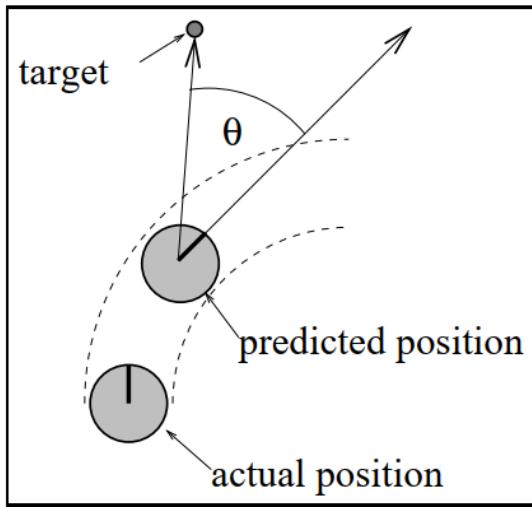
2.5.2.2. Lựa chọn vận tốc để tối ưu hàm mục tiêu

Hàm mục tiêu trong thuật toán DWA được định nghĩa như dưới đây:

$$G(v, \omega) = \sigma(\alpha \cdot \text{angle}(v, \omega) + \beta \cdot \text{dist}(v, \omega) + \gamma \cdot \text{vel}(v, \omega))$$

Ba hàm phụ thuộc biến vận tốc (v, ω) ở vế phải trong hàm mục tiêu là ba bước chính trong nhiệm vụ tối ưu hàm mục tiêu được đặt ra:

- Hướng mục tiêu (Target heading): hàm angle là phép đo sự điều chỉnh giữa hướng robot và hướng mục tiêu trong quá trình tiến đến điểm mục tiêu, thể hiện trực tiếp thông qua góc lệch mục tiêu θ . Khi robot di chuyển thẳng đến điểm mục tiêu thì giá trị này là tối đa. Bước này giúp robot điều chỉnh hướng đến mục tiêu một cách trơn tru khi mà gặp phải vật cản.



Hình 2.16. Góc lệch mục tiêu θ

- b) Clearance: Hàm $dist$ là khoảng cách đến vật cản gần nhất trên quỹ đạo. Giá trị này đạt giá trị lớn khi không có vật cản trên quỹ đạo đường cong di chuyển của robot. Ngược lại, giá trị này càng nhỏ khi khoảng cách này càng nhỏ và khả năng càng cao robot sẽ di chuyển theo quỹ đạo đường cong bám theo bề mặt vật cản.
- c) Velocity: Hàm vel là vận tốc thẳng của robot và hỗ trợ di chuyển nhanh hơn.

Các hệ số α, β, γ được lựa chọn phù hợp với yêu cầu hàm mục tiêu, đặt tính robot và môi trường hoạt động. Hàm σ làm tròn tổng trọng số ba thành phần khi tối ưu hàm mục tiêu, tạo ra khoảng trống nhiều hơn đến vật cản. Khi hàm mục tiêu có giá trị lớn nhất thì quỹ đạo tối ưu sẽ được chọn với biến vận tốc (v^*, ω^*) tương ứng và là kết quả của thuật toán.

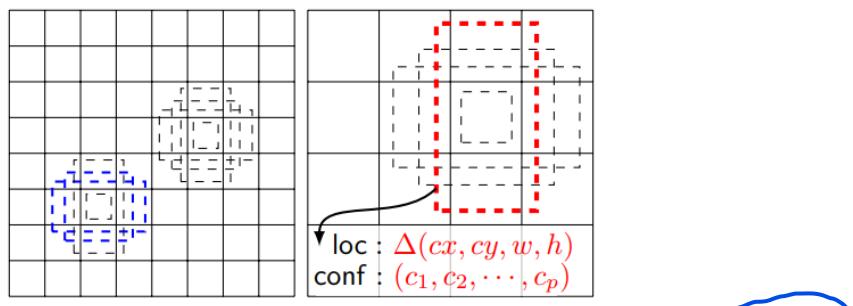
2.6. Mô hình MobileNet SSD

2.6.1. Single Shot Detector (SSD)

Được đề xuất vào năm 2016, SSD [10] là mô hình phát hiện nhiều vật thể trên ảnh chỉ với một lần quét (single stage), so với các phương pháp áp dụng RPN (Regional Proposal Network) như R-CNN cần hai lần quét cho việc tìm Region Proposals và Object Detection. Ngoài ra, mô hình còn loại bỏ các bước như tái lấy mẫu đặc trưng. Nó kết hợp tất cả quá trình tính toán trong một mạng duy nhất, nên dễ huấn luyện và tích hợp vào các hệ thống cần đến một Object Detector có khối lượng tính toán nhẹ và hỗ trợ dự đoán thời gian thực.

Kiến trúc của Single Shot Detector có thể được mô tả như dưới đây.

Multibox Detector

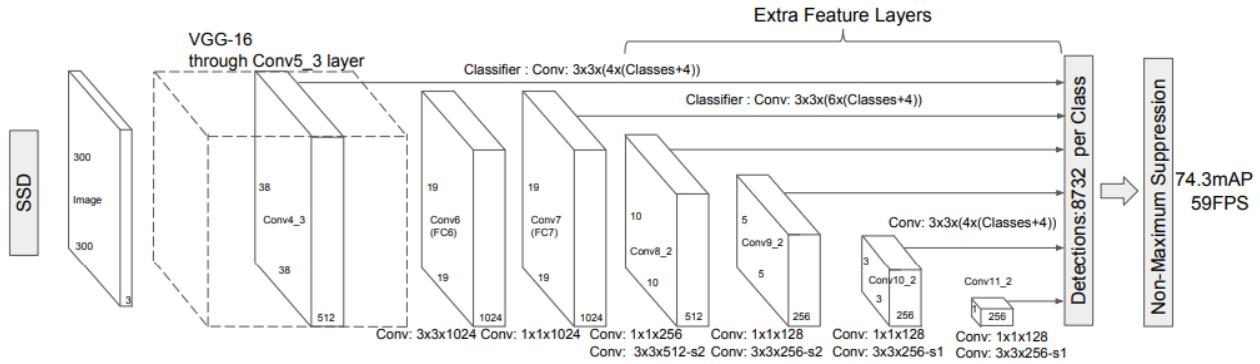


Hình 2.17 Kết quả của một lớp feature map sau khi qua một bộ lọc dự đoán

Ở phần trên cùng của kiến trúc mạng SSD là một tập các bộ lọc tích chập dùng cho việc dự đoán. Sau khi có được các feature maps với kích thước khác nhau từ các lớp trích xuất đặc trưng, ví dụ: mxn (như Hình 2.17 là 8×8 và 4×4) với số kênh là p , một kernel $3 \times 3 \times p$ được áp dụng lên đó. Các feature maps này chứa mxn vị trí cần được dự đoán. Với mỗi vị trí, ta thu được k bounding box với kích thước và tỉ lệ hình dạng khác nhau. Ý tưởng là, có lẽ một hình dạng bounding box nào đó sẽ phù hợp với một class cụ thể hơn (hình chữ nhật đứng cho người, ngang cho xe ô tô). Với mỗi bounding box sẽ tính được c class score và 4 offsets tương ứng với thông số của bounding box gốc tại vị trí mà bounding box đó thuộc về trên feature map. Vì thế, ta có được $(c + 4)kmn$ đầu ra.

Kiến trúc mạng

Để cho kết quả dự đoán chính xác hơn, các lớp khác nhau của các feature map cũng được cho qua các lớp tích chập 3×3 .



Hình 2.18 Kiến trúc mạng SSD

Ví dụ, tại lớp tích chập Conv4_3 có kích thước $38 \times 38 \times 512$, bộ lọc 3×3 được áp dụng. Có 4 bounding box tại mỗi vị trí trên feature map, và mỗi bounding sẽ có ($\text{số class} + 4$) đầu ra. Vì vậy, tại lớp tích chập này, giả sử có 20 class, số đầu ra sẽ là $38 \times 38 \times 4 \times (20+4) = 144,400$ và số bounding box là $38 \times 38 \times 4 = 5776$. Tương tự với các lớp tích chập khác, tổng cộng, ta có được 8732 bounding box, so với YOLO có 98 box.

Hàm mất mát (Loss function)

Hàm mất mát là sự kết hợp của hai hàm nhỏ: L_{conf} và L_{loc} . Trong đó, N là số box phù hợp. L_{loc} là hàm mất mát vị trí (localization loss) giữa thông số vị trí của box dự đoán (l) và ground truth box (g). Các thông số này bao gồm điểm trung tâm (cx, cy) và kích thước của box (w, h). L_{conf} là hàm mất mát về độ tự tin dự đoán (confidence loss) dưới dạng hàm softmax của c class score.

$$L(x, c, l, g) = \frac{1}{N} (L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

Tỉ lệ và tỉ số hình dáng của các box gốc trong feature map (Scale and aspect ratio of default boxes)

Trong mô hình SSD, mỗi feature map sẽ học để chịu trách nhiệm cho một tỉ lệ cụ thể của vật thể. Giả sử có m feature map để dự đoán, ta tính được s_k cho map thứ k . s_{min} là 0.2 và s_{max} là 0.9. Nghĩa là, tỉ lệ tại lớp thấp nhất là 0.2 và cao nhất là 0.9. Tất cả các lớp ở giữa được cách đều.

$$s_k = s_{\min} + \frac{s_{\max} - s_{\min}}{m-1}(k-1), \quad k \in [1, m]$$

Với mỗi tỉ lệ s_k , ta có 5 tỉ số hình dáng $\{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$. Ta có thể tính được kích thước cho mỗi default box như sau:

$$\omega_k^a = s_k \sqrt{a_r}, h_k^a = s_k / \sqrt{a_r}$$

Với tỉ số là 1, ta lại thêm một default box có tỉ lệ như bên dưới, tổng cộng, ta có nhiều nhất 6 bounding box cho mỗi vị trí trên feature map.

$$s'_k = \sqrt{s_k s_{k+1}}$$

Trong quá trình dự đoán, SSD sử dụng Non – Maximum Suppression để loại bỏ các dự đoán trùng lặp chỉ đến cùng một đối tượng. SSD sắp xếp các dự đoán theo độ tin cậy. Bắt đầu từ dự đoán có độ tin cậy lớn nhất, SSD đánh giá xem có bất kỳ bounding box dự đoán nào trước đó có $\text{IoU} > 0,45$ với bounding box dự đoán hiện tại cho cùng một lớp hay không. Nếu tìm thấy, bounding box dự đoán hiện tại sẽ bị bỏ qua. SSD giữ nhiều nhất 200 bounding box dự đoán hàng đầu cho mỗi ảnh.

Kết quả huấn luyện trên tập dữ liệu benchmark VOC2007

Với ảnh đầu vào 300x300 (SSD300), SSD đạt kết quả 74.3% mAP trên tập test VOC2007 với tốc độ 59 FPS khi dùng GPU Nvidia Titan X, và đạt 76.9% mAP cho ảnh đầu vào 512x512 (SSD512), vượt cả mô hình Fast R-CNN (7 FPS – 73.2% mAP) và YOLO (45 FPS – 63.4% mAP). So với các mô hình single stage khác, SSD có độ chính xác tốt hơn ngay cả với ảnh đầu vào có kích thước nhỏ hơn.

System	VOC2007 test mAP	FPS (Titan X)	Number of Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	~6000	~1000 x 600
YOLO (customized)	63.4	45	98	448 x 448
SSD300* (VGG16)	77.2	46	8732	300 x 300
SSD512* (VGG16)	79.8	19	24564	512 x 512

Hình 2.19 Kết quả huấn luyện mô hình SSD trên tập dữ liệu VOC2007 so với các mô hình khác

2.6.2. Mô hình MobileNet

SSD sử dụng các đặc trưng trích xuất được từ ảnh để thực hiện dự đoán, và các đặc trưng này được trích xuất bởi các thuật toán hoặc mô hình khác.

Được đề xuất vào năm 2017 đến từ Google, MobileNet [11] là một mô hình CNN được sử dụng cho việc phân loại vật thể trong ảnh (Object Classification). Với một số ưu điểm được nêu ra sau đây, MobileNet được dùng làm base network (các lớp neuron đầu thực hiện chức năng trích xuất đặc trưng trong một mô hình CNN cho ảnh), kết hợp với mô hình SSD để tạo thành một mô hình hoàn chỉnh thực hiện tác vụ Object Detection cho ảnh.



Hình 2.20 Ứng dụng sử dụng MobileNet SSD cho tác vụ Object Detection

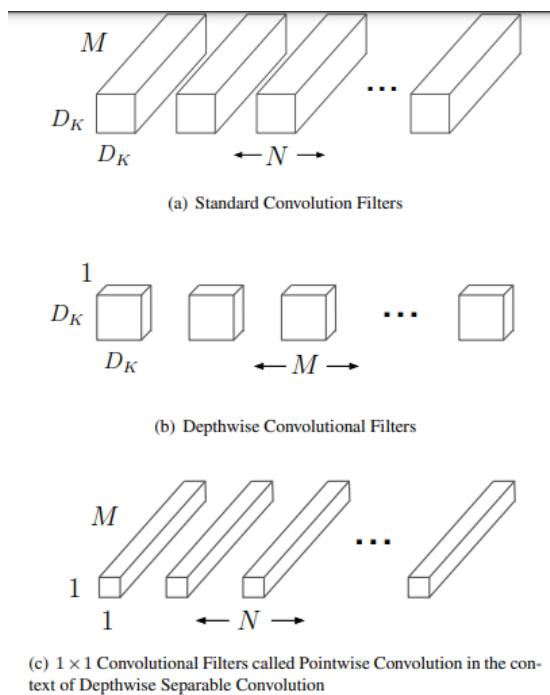
Trong MobileNet, Depthwise Separable Convolution được sử dụng để giảm kích thước và độ phức tạp của mô hình, thực sự hữu ích cho các ứng dụng di động và thị giác nhúng. Hiệu quả trên đạt được nhờ sử dụng ít đi các tham số, các phép toán cộng và nhân trong mô hình. Ngoài ra, 2 tham số còn được sử dụng để tinh chỉnh mô hình dễ dàng hơn là hệ số nhân chiều rộng α (width multiplier) và hệ số nhân độ phân giải ρ (resolution multiplier). Kiến trúc mạng

MobileNet được xây dựng nhằm mục đích giảm khối lượng tính toán và kích thước mô hình, nhờ vậy tăng đáng kể thời gian xử lý và có độ trễ thấp nhưng vẫn không giảm nhiều độ chính xác.

Mô hình MobileNet sẽ được giới thiệu như sau:

Depthwise Seperable Convolution

Depthwise Seperable Convolution là một depthwise convolution theo sau bởi một pointwise convolution.



Hình 2.21 Sự kết hợp của depthwise convolution và pointwise convolution

Các lớp tích chập cơ bản thường lọc và kết hợp đầu vào thành một đầu ra mới trong một bước duy nhất. Depthwise seperable convolution chia nó thành hai lớp. Phương pháp này giúp giảm khối lượng tính toán và kích thước mô hình. Hình 2.21 mô tả một lớp tích chập bình thường được tách thành hai lớp nêu trên.

Depthwise convolution là sử dụng các kernel có kích thước $DK \times DK$. Pointwise convolution là một số các kernel 1×1 cho kết quả của Depthwise convolution để đạt được số chiều mong muốn. Với M là số kênh đầu vào, N là số kênh đầu ra, DK là kích thước kernel,

DF là kích thước của feature map. So với tích chập thông thường, khối lượng tính toán của depth wise convolution giảm đi:

$$= \frac{\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F}}{\frac{1}{N} + \frac{1}{D_K^2}}$$

Khi $DK = 3$, khối lượng tính toán giảm đi 8 đến 9 lần với độ chính xác giảm không nhiều.

Kiến trúc toàn mạng

Kiến trúc của mạng MobileNet được mô tả trong Hình 2.22 bên dưới, với lưu ý là Batch Normalization (BN) và hàm ReLU activation được sử dụng sau mỗi lớp tích chập.

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5x Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Hình 2.22 Kiến trúc mạng MobileNet

Width Multiplier α và Resolution Multiplier ρ

Width Multiplier α được mô tả để quản lý số kênh, đặt M thành αM . Trong đó, α nằm trong khoảng $[0, 1]$ và thường được đặt giá trị 1, 0.75, 0.5 hoặc 0.25. Khối lượng tính toán và số tham số của mô hình được giảm vào khoảng bình phương của α .

Resolution Multiplier được mô tả là để quản lý được độ phân giải của ảnh đầu vào của mạng, với ρ , kích thước ảnh đầu vào sẽ bị ảnh hưởng.

Khối lượng tính toán trở thành:

$$D_K \cdot D_K \cdot \alpha M \cdot \rho D_F \cdot \rho D_F + \alpha M \cdot \alpha N \cdot \rho D_F \cdot \rho D_F$$

Kết quả

Khi được sử dụng làm base network cho các mô hình Object Detection khác và được huấn luyện trên tập dataset COCO, MobileNet cho kết quả tương đương với số lượng tham số và phép tính giảm gấp nhiều lần so với các mô hình trích xuất đặc trưng khác, kết quả được biểu diễn trong Hình 2.23.

Framework Resolution	Model	mAP	Billion Mult-Adds	Million Parameters
SSD 300	deeplab-VGG	21.1%	34.9	33.1
	Inception V2	22.0%	3.8	13.7
	MobileNet	19.3%	1.2	6.8
Faster-RCNN 300	VGG	22.9%	64.3	138.5
	Inception V2	15.4%	118.2	13.3
	MobileNet	16.4%	25.2	6.1
Faster-RCNN 600	VGG	25.7%	149.6	138.5
	Inception V2	21.9%	129.6	13.3
	MobileNet	19.8%	30.5	6.1

Hình 2.23 So sánh MobileNet với các mô hình khác khi là base network trong một mạng Object Detection huấn luyện trên tập COCO

Mô hình MobileNet SSD là mô hình sử dụng MobileNet làm base network và SSD làm Detector. Mô hình được huấn luyện trên tập dữ liệu COCO (Common Objects in Context) và sau đó được fine-tuned trên tập PASCAL VOC, đạt mAP 72.7%.

2.7. Kmean-Clustering

Kmeans -Clustering là một thuật toán học không giám sát. Với các dữ liệu không biết trước về nhãn (label), thuật toán thực hiện phân cụm chúng thành K cụm, mà các thành phần trong mỗi cụm có đặc tính giống nhau. Từ dữ liệu đầu vào và số cụm muốn phân, Kmeans sẽ chỉ ra center (điểm trung tâm) của mỗi cụm và phân các điểm dữ liệu đầu vào vào từng cụm tương ứng.

Với $\mathbf{X} = [\mathbf{x1}, \mathbf{x2}, \mathbf{x3}, \dots, \mathbf{x}_N]$ là ma trận các dữ liệu đầu vào, trong đó N là số điểm dữ liệu và $N > K$. Với mỗi điểm \mathbf{x}_i , đặt vector $\mathbf{y}_i = [y_{i1}, y_{i2}, \dots, y_{iK}]$ là vector nhãn của nó ở dạng one-hot coding. Ta cần tìm vector center $\mathbf{m} = [\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_K]$. Theo một bài viết về machine learning [12], các bước thuật toán được thực hiện như sau:

- Bước 1: Chọn K điểm dữ liệu bất kỳ làm center ban đầu và lưu vào \mathbf{m} .
- Bước 2: Phân mỗi điểm dữ liệu vào cụm có center gần với nó nhất (khoảng cách được tính bằng norm 2 của điểm dữ liệu và từng center).

$$j = \operatorname{argmin} \left\| \mathbf{x}_i - \mathbf{m}_j \right\|_2^2$$

$$y_{ij} = 1$$

- Bước 3: Nếu việc gán dữ liệu vào từng cluster ở bước 2 không thay đổi so với vòng lặp trước nó thì ta dừng thuật toán.
- Bước 4: Cập nhật center cho từng cluster bằng cách lấy trung bình cộng của tất cả các điểm dữ liệu đã được gán vào cluster đó sau bước 2.

$$\mathbf{m}_j = \frac{\sum_{i=1}^N y_{ij} \mathbf{x}_i}{\sum_{i=1}^N y_{ij}}$$

- Bước 5: Quay lại bước 2

2.8. Hiệu chỉnh Magnetometer

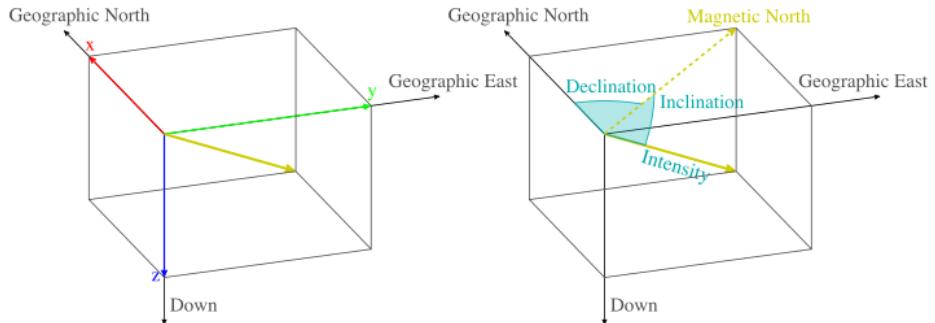
Các định nghĩa thuật ngữ, sự hình thành các ma trận và diễn giải phương trình trong mục này được trình bày rõ ở phụ lục I.

2.8.1. Từ trường trái đất

Vector từ trường trái đất trong hệ tọa độ NED (hay hệ tọa độ toàn cục – ký hiệu là n-frame) là \mathbf{h}_0 :

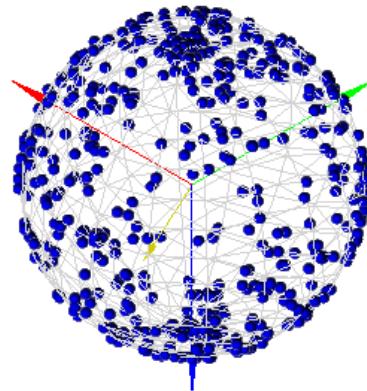
$$\mathbf{h}_0 = F \begin{bmatrix} \cos(I) \cos(D) \\ \cos(I) \sin(D) \\ \sin(I) \end{bmatrix}$$

Với I là độ từ thiêng, D là độ từ khuynh. Ở Việt Nam, độ từ thiêng là không đáng kể, vì thế có thể bỏ qua D .



Hình 2.24 Độ từ thiêng và độ từ khuynh của từ trường trái đất

Giả sử trong một môi trường không có nhiễu từ và ta có một magnetometer (cảm biến từ trường) 3 trục lý tưởng. Với \mathbf{h} là vector từ trường trong hệ tọa độ vật thể (hay hệ tọa độ của IMU) - ký hiệu là b-frame, khi xoay IMU 360° quanh tất cả các trục là đường thẳng đi qua tâm, ta có quỹ tích các mẫu \mathbf{h} sẽ nằm trên một mặt cầu. Tuy nhiên trong thực tế, magnetometer sẽ bị ảnh hưởng bởi hai nguồn nhiễu, đó là sai số thiết bị và nhiễu từ trường. Sai số thiết bị là duy nhất và là hằng số cho mỗi thiết bị. Hai thành phần chính của nhiễu từ trường là hard iron và soft iron.



Hình 2.25 Mặt cầu từ trường

2.8.2. Lý thuyết hiệu chỉnh magnetometer

Theo một phương pháp calib từ trường của Testlab [13], trong khi \mathbf{h} là giá trị đúng, giá trị từ trường đo được từ cảm biến khi có nhiễu có thể dễ dàng được viết như sau, trong đó, \mathbf{A}

là ma trận kết hợp các scale factor, sự lệch trục và hiệu ứng soft iron; \mathbf{b} là sự kết hợp của các vector bias bao gồm hard iron:

$$\mathbf{h}_m = \mathbf{A}\mathbf{h} + \mathbf{b}$$

Ta giả sử các giá trị dự đoán của \mathbf{A} và \mathbf{b} lần lượt là $\widehat{\mathbf{A}}$ và $\widehat{\mathbf{b}}$. Khi đó, bất cứ kết quả từ trường sau khi calib nào đều là \mathbf{h}_n , giả sử \mathbf{h}_n có giá trị giống với giá trị đúng \mathbf{h} , ta có phương trình liên hệ giữa giá trị đo và giá trị đúng như sau:

$$\mathbf{h} = \widehat{\mathbf{A}}^{-1}(\mathbf{h}_m - \widehat{\mathbf{b}})$$

Có thể chứng minh được rằng, sự thay đổi tuyến tính của \mathbf{h} ở phương trình trên khiến quỹ tích các mẫu của \mathbf{h}_m nằm trên một mặt ellipsoid. Vì vậy, ta cần sử dụng một thuật toán ellipsoid fitting để tìm được xấp xỉ các thông số của mặt ellipsoid này. Từ đó ta có thể tìm được $\widehat{\mathbf{A}}$ và $\widehat{\mathbf{b}}$ để calib lại cảm biến. Thuật toán ellipsoid fitting sử dụng trong đề tài là thuật toán bình phương cực tiểu được giới thiệu bởi J.G. Griffiths [14] và có thể dễ dàng được sử dụng với script MATLAB cung cấp bởi tác giả.

2.9. Kết hợp cảm biến bằng bộ lọc Madgwick

Quaternion là một số siêu phức (hyper-complex) có 1 thành phần vô hướng (scalar) và một vector 3 phần tử phức. Quaternion được dùng để biểu diễn hiệu quả sự xoay, cũng như hướng trong không gian ba chiều và có thể được sử dụng thay cho ma trận xoay DCM (Direction cosine matrix) với các góc Euler được dùng để tính toán sự tương quan về hướng của hai hệ tọa độ khác nhau. Một quaternion được biểu diễn như sau, và \mathbf{q}^* là liên hợp phức của \mathbf{q} :

$$\begin{aligned}\mathbf{q} &= q_1 + q_2i + q_3j + q_4k \\ \mathbf{q}^* &= q_1 - q_2i - q_3j - q_4k\end{aligned}$$

Khi cần tìm một vector trong hệ tọa độ A, khi biết nó trong hệ tọa độ B. Gọi \mathbf{q} là quaternion biểu diễn phép quay từ B sang A.

$$\mathbf{v}_A = \mathbf{q}^* \otimes \mathbf{v}_B \otimes \mathbf{q}$$

2.9.1. Orientation từ góc xoay (Gyroscope)

Gyroscope 3 trục đo vận tốc xoay trên trục x, y, z của b-frame, lần lượt là ω_x , ω_y , ω_z , và được biểu diễn dưới vector ω^b

$$\omega^b = [0 \ \omega_x \ \omega_y \ \omega_z]$$

Sự biến thiên góc xoay của n-frame so với b-frame được tính như sau, với t là thời điểm lấy mẫu hiện tại, và $t - 1$ là thời điểm trước đó một mẫu, \hat{q} biểu diễn giá trị được ước lượng, giá trị này sẽ được cập nhật bằng thuật toán fusion Madgwick sẽ được nêu ở phần sau:

$$\dot{q}_{\omega,t} = \frac{1}{2} \hat{q}_{t-1} \otimes \omega_t^b$$

Rồi rạc theo thời gian, ta tìm được quaternion của n-frame so với b-frame khi sử dụng vận tốc góc từ gyroscope:

$$q_{\omega,t} = \hat{q}_{t-1} + \dot{q}_{\omega,t} \Delta t = \hat{q}_{t-1} + \frac{1}{2} \Delta t \hat{q}_{t-1} \otimes \omega_t^b$$

2.9.2. Orientation từ accelerometer và magnetometer

Giả sử cảm biến từ trường đã qua hiệu chỉnh. Từ hai cảm biến accelerometer và magnetometer, một phương pháp dùng Madgwick filter [15] xem bài toán tìm hướng của b-frame so với n-frame là bài toán tối ưu, sao cho hướng của một trường tham chiếu được định nghĩa trước trong n-frame \hat{d} , sau khi được xoay với quaternion \hat{q} , sẽ thẳng hàng với hướng của trường đó đo được trong b-frame \hat{s} . Hàm mục tiêu sẽ là, với các phần tử đều là vector 1x4:

$$\min f(\hat{q}, \hat{d}, \hat{s})$$

$$f(\hat{q}, \hat{d}, \hat{s}) = \hat{q}^* \otimes \hat{d} \otimes \hat{q} - \hat{s}$$

Có rất nhiều thuật toán tối ưu tồn tại, nhưng gradient descent là đơn giản nhất cho việc tính toán và triển khai. Thuật toán này dựa trên phương pháp Newton tìm cực trị của hàm. Các giá trị q tại mỗi thời điểm sẽ được cập nhật như sau, với μ là hệ số cập nhật:

$$\mathbf{q}_t = \widehat{\mathbf{q}}_{t-1} - \mu_t \frac{\nabla f(\widehat{\mathbf{q}}_{t-1}, \widehat{\mathbf{d}}, \widehat{\mathbf{s}})}{\|\nabla f(\widehat{\mathbf{q}}_{t-1}, \widehat{\mathbf{d}}, \widehat{\mathbf{s}})\|}$$

Gradient của hàm mục tiêu được biểu diễn bởi chính nó và Jacobian của nó:

$$\nabla f(\widehat{\mathbf{q}}, \widehat{\mathbf{d}}, \widehat{\mathbf{s}}) = J^T(\widehat{\mathbf{q}}_{t-1}, \widehat{\mathbf{d}}) f(\widehat{\mathbf{q}}_{t-1}, \widehat{\mathbf{d}}, \widehat{\mathbf{s}})$$

Equation 2.1

Tuy nhiên nếu vector trường đó chỉ có thành phần trong một hoặc hai trục của n-frame, phương trình sẽ được đơn giản hóa. Hướng của trọng lực được tìm bởi accelerometer cũng là trục z của n-frame. Với $\widehat{\mathbf{a}}$ là vector gia tốc đã được chuẩn hóa và thêm phần tử 0 để trở thành vector 1x4, và $\widehat{\mathbf{g}} = [0 \ 0 \ 0 \ 1]$, hai vector này sẽ được thay cho $\widehat{\mathbf{d}}, \widehat{\mathbf{s}}$ trong Equation 2.1. Ngoài ra, từ trường trái đất chỉ có một thành phần trong trục x và một trong trục z của n-frame và được biểu diễn bởi vector $\widehat{\mathbf{b}} = [b_x \ 0 \ b_z]$. Với $\widehat{\mathbf{m}} = [m_x \ m_y \ m_z]$ là từ trường chuẩn hóa đo được từ magnetometer. Ta sẽ dùng vector $\widehat{\mathbf{b}}$ và $\widehat{\mathbf{m}}$ thay cho $\widehat{\mathbf{d}}$ và $\widehat{\mathbf{s}}$ trong Equation 2.1. Phương trình có được theo giá trị từ trường và hướng gia tốc trọng trường được kết hợp như sau:

$$\mathbf{f} = \begin{bmatrix} fg(\widehat{\mathbf{q}}, \widehat{\mathbf{a}}) \\ fb(\widehat{\mathbf{q}}, \widehat{\mathbf{b}}, \widehat{\mathbf{m}}) \end{bmatrix}$$

$$\mathbf{J} = \begin{bmatrix} Jg(\widehat{\mathbf{q}}) \\ Jb(\widehat{\mathbf{q}}, \widehat{\mathbf{b}}) \end{bmatrix}$$

Giá trị tối ưu của μ_t được tìm sao cho tốc độ hội tụ của \mathbf{q}_t được giới hạn bởi sự thay đổi hướng vật lý của robot để tránh sự vọt lô khi có step size lớn. Vì vậy được mô tả như phương trình.

$$\mu_t = \alpha \left\| \dot{\mathbf{q}}_{\omega, t} \right\| \Delta t, \alpha > 1$$

2.9.3. Bộ lọc Madgwick

Giá trị dự đoán quaternion của b-frame so với n-frame $\widehat{\mathbf{q}}_t$ được tính toán bởi sự kết hợp giữa \mathbf{q}_t và $\mathbf{q}_{\omega, t}$. Với γ_t là trọng số gắn cho mỗi giá trị đo. γ_t được tìm sao cho tốc độ hội

tụ của \mathbf{q}_t sẽ tỉ lệ nghịch với tốc độ hội tụ của $q_{\omega,t}$. Trong đó, $\frac{\mu_t}{\Delta t}$ là tốc độ hội tụ của \mathbf{q}_t và β là tốc độ phân kỳ của $q_{\omega,t}$.

$$\hat{\mathbf{q}}_t = \gamma_t \mathbf{q}_t + (1 - \gamma_t) \mathbf{q}_{\omega,t}, 0 \leq \gamma \leq 1$$

$$(1 - \gamma_t) \beta = \gamma_t \frac{\mu_t}{\Delta t}$$

$$\gamma_t = \frac{\beta}{\frac{\mu_t}{\Delta t} + \beta}$$

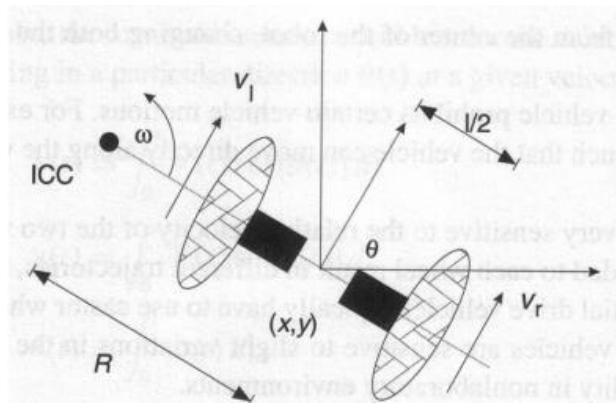
Với các phân tích và triển khai của bộ lọc Madgwick, giá trị của quaternion ở mỗi thời điểm t có thể được ước lượng như sau:

$$\hat{\mathbf{q}}_t = \hat{\mathbf{q}}_{t-1} + \beta \Delta t \left(-\frac{\nabla f}{\|\nabla f\|} \right) + \dot{\mathbf{q}}_{\omega,t} \Delta t$$

2.10. Điều khiển tốc độ động cơ

2.10.1. Differential Drive

Khi robot được thiết kế theo mô hình Differential Drive, mối liên hệ giữa vận tốc của từng bánh xe v_l , v_r với vận tốc xoay ω và vận tốc tịnh tiến v của robot có thể được miêu tả rõ ràng bằng công thức.



Hình 2.26 Mô hình Differential Drive

Với ICC là tâm xoay ảo khi robot di chuyển với vận tốc hai bánh như trên Hình 2.26. R là khoảng cách từ tâm trục hai bánh xe đến ICC, l là khoảng cách giữa hai bánh xe. Vận tốc hai bánh xe được tính từ v và ω như sau:

$$v_l = \frac{2v - \omega l}{2} \quad v_r = \frac{2v + \omega l}{2}$$

2.10.2.Lowpass Filter

Lowpass Filter (bộ lọc thông thấp), lọc đi tín hiệu có tần số cao (tín hiệu thay đổi nhiều và đột ngột), tạo độ trễ mong muốn cho tín hiệu. Với ω_{LPF} là tần số cắt của bộ lọc, hàm truyền Lowpass Filter có dạng:

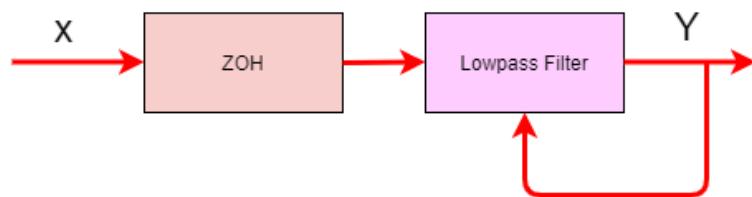
$$G(s) = \frac{1}{\omega_{LPF}(s + \frac{1}{\omega_{LPF}})}$$

Ở miền z, hàm truyền có dạng:

$$\frac{Y(z)}{X(z)} = (1 - z^{-1}) * L^{-1}\left\{\frac{G(s)}{s}\right\}$$

Sau khi biến đổi sang miền rời rạc, ta có công thức như sau, Với T là chu kỳ điều khiển:

$$y(k) = (1 - e^{-\omega_{LPF} * T})x(k-1) + e^{-\omega * T}y(k-1)$$



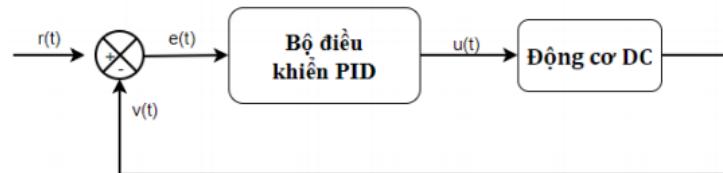
Hình 2.27 Sơ đồ khói Lowpass Filter

2.10.3.Bộ điều khiển PID

Bộ điều khiển PID là bộ điều khiển vòng kín (có hồi tiếp) được sử dụng phổ biến trong việc điều khiển Động cơ DC. Với T là chu kỳ điều khiển, $u(k)$ là tín hiệu điều khiển ở chu kỳ k , $r(k)$ là tín hiệu đặt, $v(k)$ là tín hiệu hồi tiếp, $e(k)$ là sai số giữa tín hiệu đặt và tín hiệu hồi

tiếp $e(t) = r(t) - v(t)$. Rồi rắc hàm truyền bộ PID, ta có phương trình cập nhật tín hiệu điều khiển như sau:

$$u(k) = u(k-1) + Kp * (e(k) - e(k-1)) + Ki * \frac{T}{2} * (e(k) + e(k-1)) + \frac{Kd}{T} * (e(k) - 2 * e(k-1) + e(k-2))$$



Hình 2.28 Sơ đồ khối bộ điều khiển PID

Trong đó, Kp, Ki, Kd là các hệ số của 3 khâu (khâu P, khâu tích phân I và khâu đạo hàm D) giúp tinh chỉnh bộ điều khiển, và có ảnh hưởng lên bộ điều khiển như bảng dưới đây.

Đáp ứng vòng kín	Thời gian lên	Vọt lố	Thời gian xác lập	Sai số xác lập
K_p	Giảm	Tăng	Thay đổi nhỏ	Giảm
K_I	Giảm	Tăng	Tăng	Loại bỏ
K_D	Thay đổi nhỏ	Giảm	Giảm	Thay đổi nhỏ

Hình 2.29 Ảnh hưởng của các hệ số Kp, Ki, Kd lên bộ điều khiển PID¹.

¹ Nguồn: Tài liệu thí nghiệm môn Cơ sở Điều khiển tự động, khoa Điện – Điện tử, trường ĐH Bách Khoa, TP HCM

CHƯƠNG 3. NỘI DUNG THỰC HIỆN

3.1. Thiết kế và thi công mô hình

3.1.1. Thành phần phần cứng

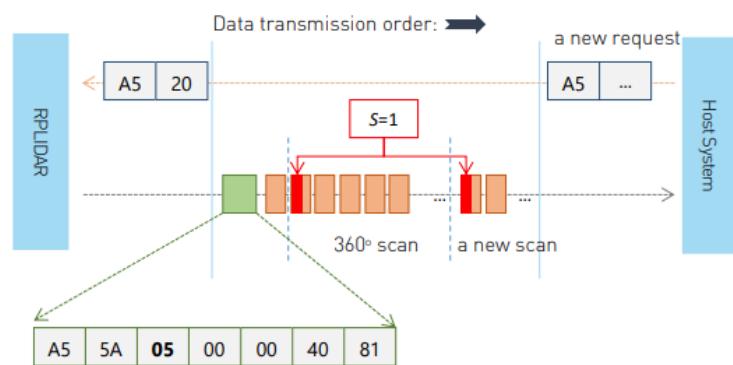
Các thông số kỹ thuật chi tiết của từng thiết bị phần cứng trong mục này được trình bày rõ ở phụ lục II.

3.1.1.1. RPLidar A1

Cảm biến Laser Radar RPLIDAR A1 12m của hãng SLAMTEC sử dụng giao tiếp UART, kết nối máy tính qua mạch chuyển USB-UART và phần mềm đi kèm. Dữ liệu truyền về sẽ là các point clouds. Các point cloud này được sử dụng để vẽ bản đồ 2D và tìm đường tối ưu.



Hình 3.1 RPLIDAR A1



Hình 3.2 Các Packet truyền khi LiDAR quét 360°

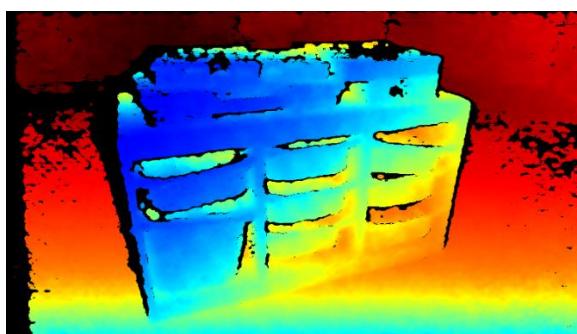
3.1.1.2. Camera Realsense D435

Dữ liệu từ Camera là đầu vào bắt buộc cho thuật toán RTAB-Map. Camera Realsense của hãng Intel là lựa chọn cho đề tài này vì ngoài việc cung cấp ảnh có độ sâu với độ phân giải ảnh cao và được hỗ trợ driver trên ROS, công nghệ Realsense còn hỗ trợ nhiều hệ điều hành và ngôn ngữ lập trình khác nhau. Intel Realsense còn được tích hợp với rất nhiều phần mềm thứ ba ngoài ROS như Unity, OpenCV, PCL và Matlab, giúp dễ dàng ứng dụng vào hệ thống sau này.



Hình 3.3 Camera Realsense Intel D435

Camera Realsense D435 đã được cǎn chỉnh (calibrate) bằng phần cứng từ nhà máy. Cảm biến trả về depth map (bản đồ có độ sâu) được tính sẵn bởi phần cứng camera với tốc độ lên tới 90FPS và chạy ở nguồn áp USB 5v, tiêu thụ từ 1 đến 1.5W.



Hình 3.4 Hình ảnh có độ sâu thu được từ Realsense

3.1.1.3. Cảm biến IMU

Cảm biến IMU được sử dụng trong đề tài là MPU9250, gồm một MPU6050 chứa cảm biến Gyroscope đo tốc độ góc 3 trục, và một cảm biến Accelerometer đo gia tốc 3 trục. Ngoài ra, MPU9250 còn tích hợp một cảm biến Magnetometer AK8963 đo từ trường 3 trục. Tín hiệu

được đọc thông qua giao thức I2C hoặc SPI. IC có đường bus master I2C phụ, hỗ trợ đọc giá trị cảm biến ngoại (cảm biến áp suất).



Hình 3.5. Cảm biến MPU9250

3.1.1.4. Máy tính nhúng Jetson Nano

Các giải thuật SLAM vẽ bản đồ và định vị cho robot được thực hiện trên ROS, vì vậy cần có một máy tính chạy hệ điều hành này gắn trên robot. Bộ kit phát triển NVIDIA Jetson Nano là một máy tính nhúng nhỏ nhưng rất mạnh mẽ, cho khả năng tính toán cao, phù hợp với các ứng dụng nhúng. Máy tính nhúng có tích hợp nhiều cổng đọc được dữ liệu từ camera, lidar, USB UART, ... và sử dụng nguồn 5V – 4A.

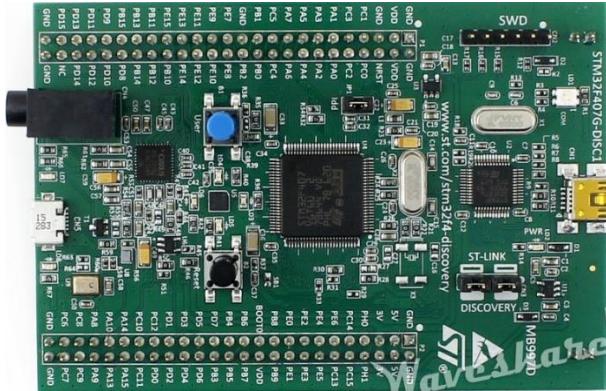


Hình 3.6 NVIDIA Jetson Nano

3.1.1.5. Vi điều khiển STM32F407VG

Hệ thống cần có một mạch nhúng thích hợp để điều khiển động cơ, đọc dữ liệu từ IMU, tính toán odometry và truyền, nhận dữ liệu với các topic trên ROS. Đề tài luận văn sử dụng kit phát triển STM32F407 Discovery board với lõi là chip STM32F407VG. Board có tích hợp

mạch nạp ST-Link V2 hỗ trợ quá trình nạp chương trình và debug, ngoài ra còn tích hợp rất nhiều ngoại vi, hỗ trợ tốt việc đọc dữ liệu cảm biến và truyền thông tin. Có thể được cấp nguồn thông qua các cáp USB với nguồn áp 5V.

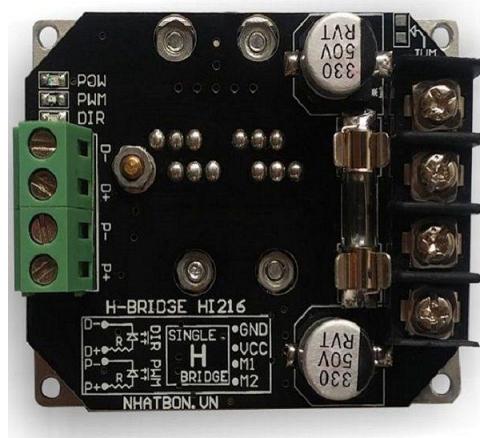


Hình 3.7 Kit phát triển STM32F407 Discovery board

3.1.1.6. Cầu H HI216 và động cơ DCM50-775 12V có Encoder

Board cầu H HI216 dùng IC kích FET chuyên dụng, cho FET luôn dẫn tốt nhất, tránh hiện tượng trùng dẫn, giúp công suất và hiệu năng của board luôn đạt ngưỡng dẫn tốt nhất và ít hư hỏng Fet. Driver sử dụng nguồn 12V – 4A, có thể được dùng để điều khiển cho một động cơ DC. Driver nhận đầu vào là hai chân điều khiển hướng, hai chân cấp xung PWM; đầu ra là hai chân nguồn cho động cơ và hai chân điều khiển động cơ.

Động cơ DC được sử dụng là DCM50-775 dùng nguồn cấp 12V – 5A.



Hình 3.8 Driver cầu H HI216 Nhật Bôn

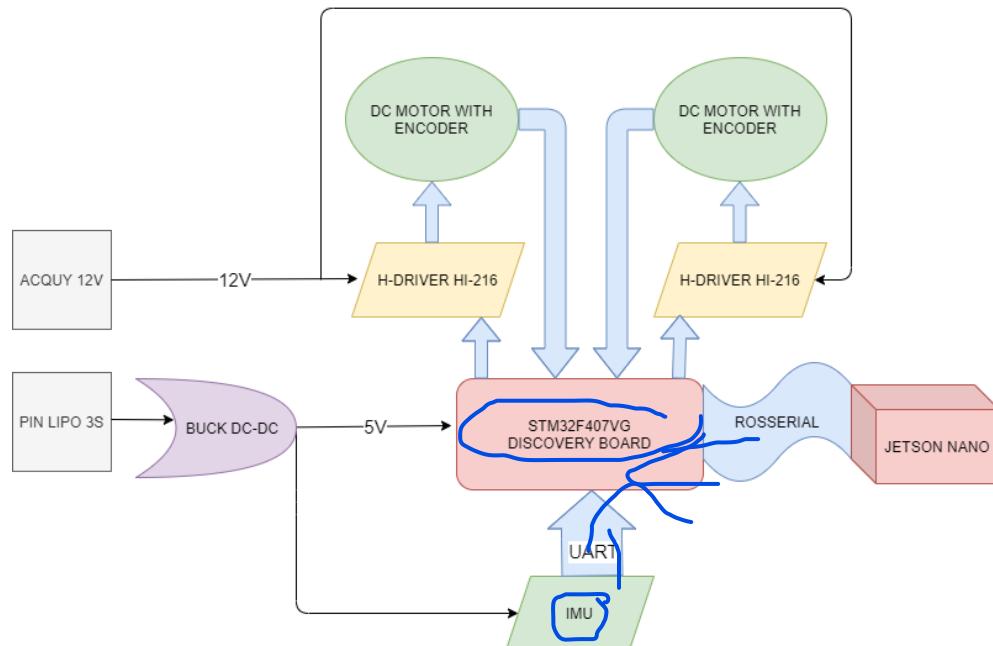


Hình 3.9 Động cơ DCM50-775 12V DC

3.1.2. Kết nối phần cứng

3.1.2.1. Khối vi điều khiển

Kit phát triển STM32F407 Discovery Board được cấp nguồn bởi pin Lipo 3S 12V hạ áp bằng mạch buck DC-DC XL4005 để có được nguồn áp 5V. Nguồn này cũng là nguồn áp vận hành IMU. Khi nhận được thông tin điều khiển từ Jetson Nano (ROS) thông qua giao thức Rosserial, vi điều khiển dùng bộ điều khiển PID để xuất xung PWM cho cầu H HI216 điều khiển động cơ. Driver cầu H được cấp nguồn bằng một acquy 12V riêng, nhằm tách riêng nguồn cho mạch công suất và nguồn cho mạch điều khiển.



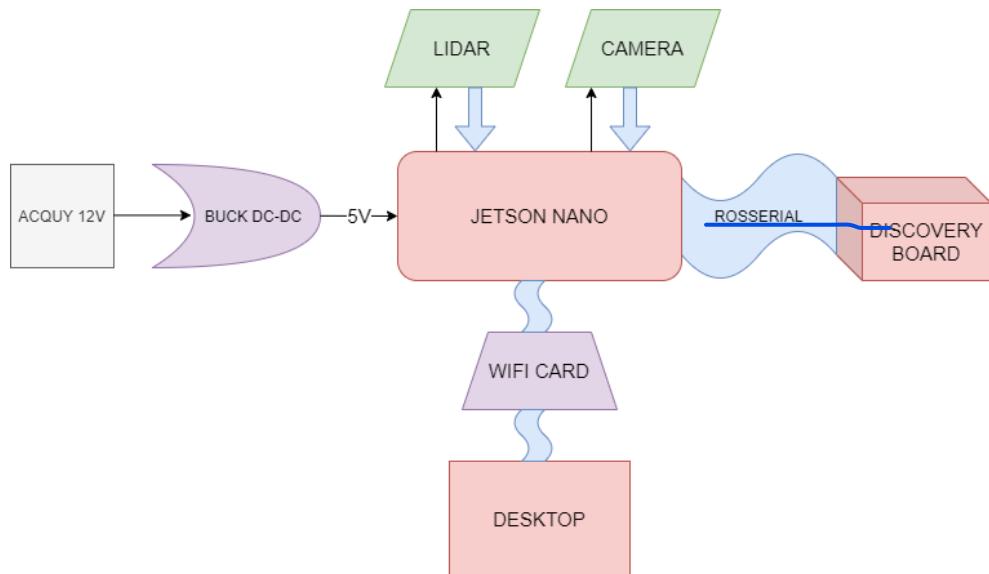
Hình 3.10 Sơ đồ khối mạch điều khiển
(Mũi tên nhỏ: đường nguồn; Mũi tên lớn: đường tín hiệu)

IMU sau khi được calib hoàn chỉnh và tính toán góc xoay sẽ được kết hợp với thông tin từ Encoder để tính được Odometry của robot và gửi về ROS trên Jetson Nano dưới dạng các message của topic /odom qua giao thức Rosserial.

3.1.2.2. Khối máy tính nhúng

Hệ điều hành Robot (ROS) được cài đặt trên máy tính nhúng Jetson Nano. Máy tính nhúng được cấp nguồn bởi một acquy 12V, qua mạch buck DC-DC 30V-4.5V 12A để có được nguồn áp 5V 4A.

Các thư viện đọc dữ liệu từ cảm biến Lidar và Camera được viết trong các package ROS trên Jetson Nano. Jetson chạy các package này để lấy thông tin và publish chúng dưới dạng các topic message lên node trên ROS, RTAB-Map sẽ subscribe vào các topic này để lấy thông tin phục vụ tác vụ của nó. hai cảm biến trên sẽ lấy nguồn từ chính máy tính nhúng thông qua các cổng USB.

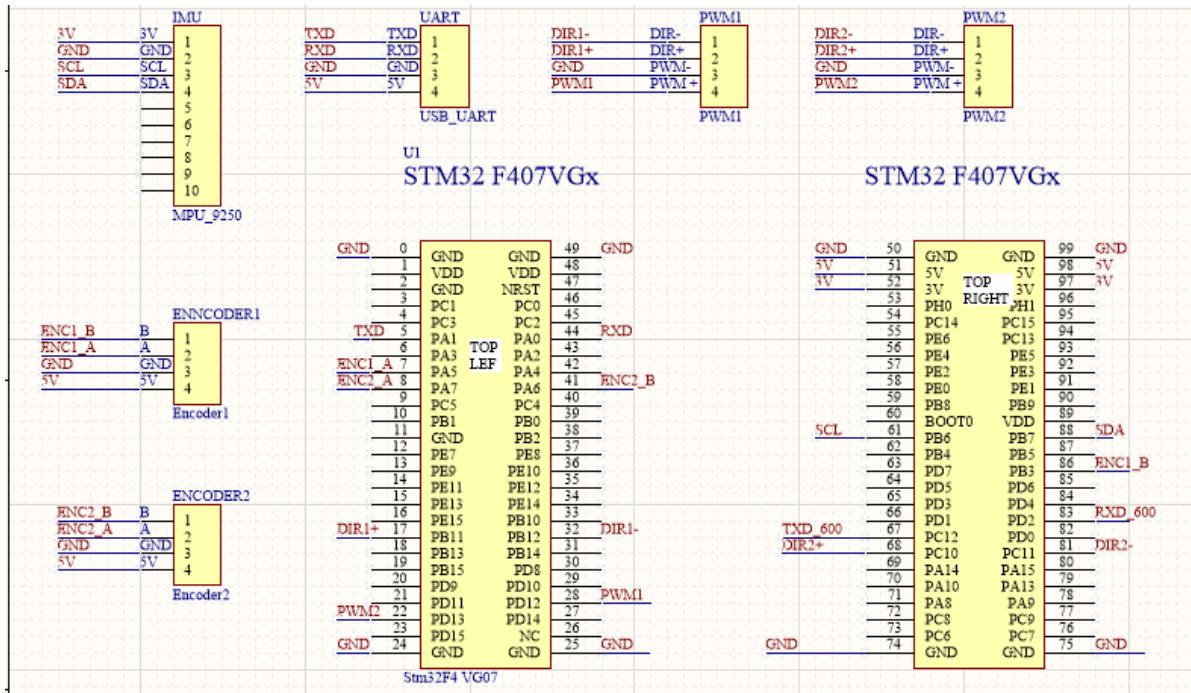


Hình 3.11 Sơ đồ khái niệm về khung máy tính nhúng.

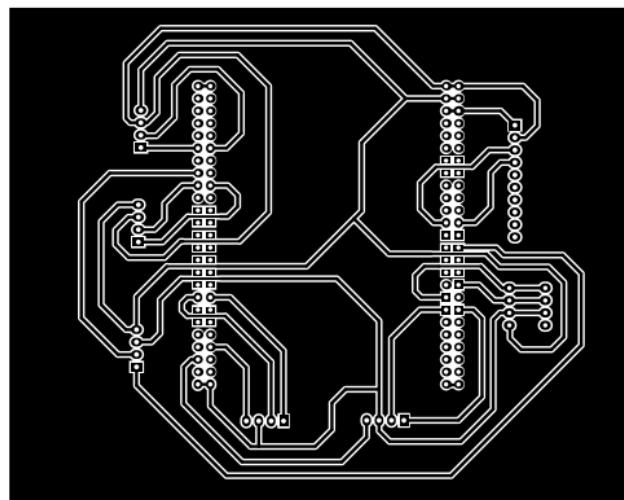
(Mũi tên nhỏ: đường nguồn; Mũi tên lớn: đường tín hiệu)

Để quan sát được kết quả của các tác vụ và thực hiện các lệnh cần thiết trên Jetson, một card wifi được gắn vào Jetson để có thể kết nối vào Wifi Hotspot của một Desktop khác. Nhờ vậy Desktop này có thể đóng vai trò như một trạm điều khiển từ xa đến Jetson Nano.

3.1.3. Thi công mạch điều khiển



Hình 3.12 Bản vẽ schematic của mạch shield cho board Discovery.



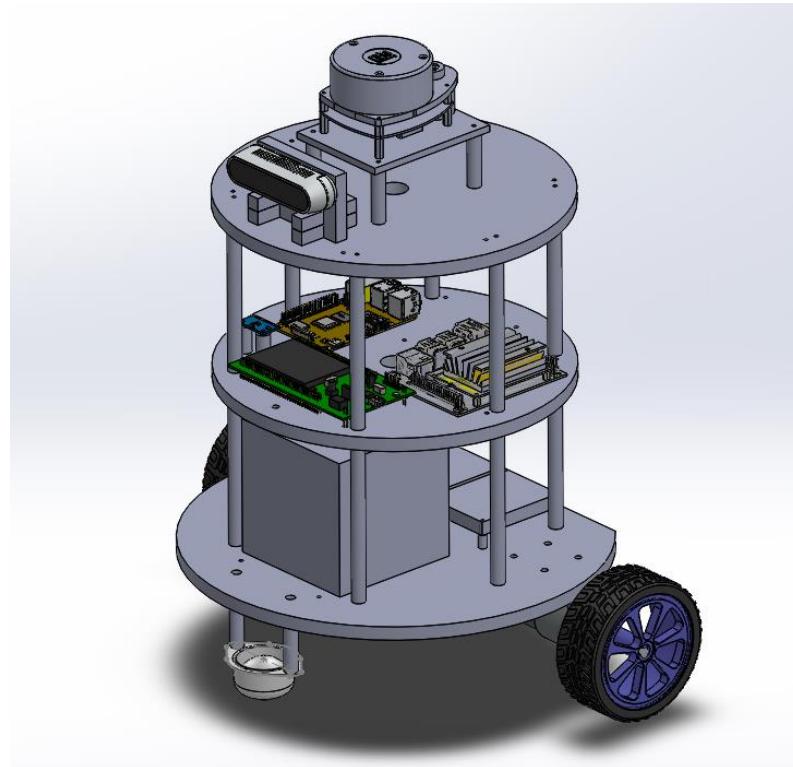
Hình 3.13 Bản in PCB của mạch shield cho board Discovery.

3.1.4. Thiết kế mô hình

Nhóm sử dụng phần mềm vẽ 3D Solidworks để vẽ mô hình robot, tự thiết kế các tầng robot đáp ứng các nhu cầu lắp ráp cần thiết cho các thành phần phần cứng được trình bày ở mục 3.1.1. Robot sẽ được vẽ với 3 tầng nhằm thực hiện 3 nhiệm vụ riêng biệt với nhau gồm:

- Tầng dưới: sẽ được sử dụng để lắp 2 động cơ có bánh xe, lắp 2 driver của động cơ và là nơi cố định nguồn điện cho robot (gồm 2 acquy và 1 pin lipo). Nhiệm vụ chính của tầng này là nhận tín hiệu điều khiển di chuyển robot và giữ nguồn điện.
- Tầng giữa: sẽ lắp vào vi điều khiển STM32F4, cảm biến IMU cùng với đó là các mạch Buck giảm áp sử dụng nguồn điện từ tầng dưới cung cấp cho các phần tử dùng điện năng và một cầu chì phòng khi chạm mạch. Nhiệm vụ chính của tầng này là thực hiện giải thuật điều khiển động cơ, thu thập dữ liệu từ cảm biến IMU và encoder, giao tiếp với thành phần máy tính nhúng thực hiện các thuật toán vẽ map.
- Tầng trên: là thành phần chính của robot, sẽ lắp đặt máy tính nhúng Jetson, cảm biến Lidar và Camera, là nơi thu thập dữ liệu môi trường và thực hiện tác vụ vẽ map cũng như các tác vụ khác của robot.

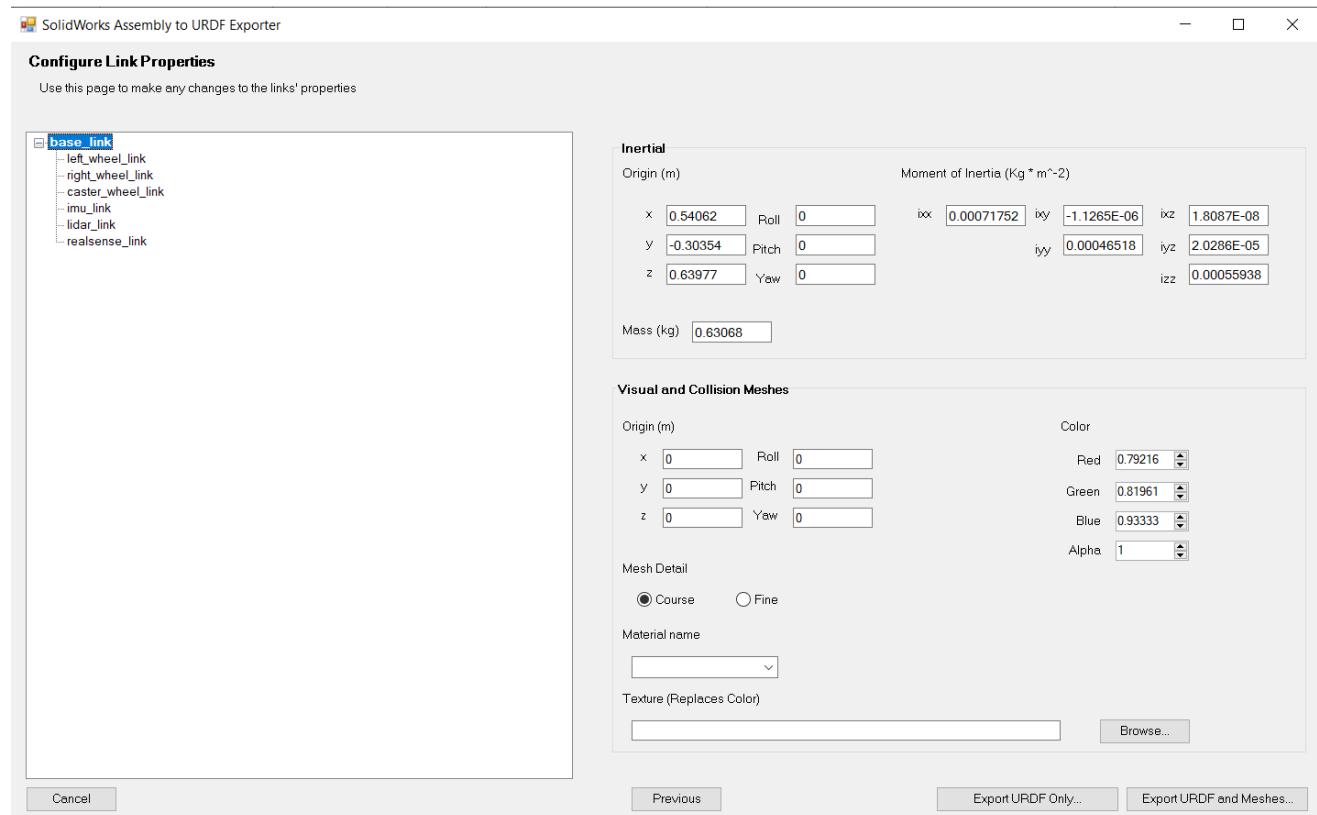
Sau quá trình tính toán kĩ lưỡng và vẽ nhiều mẫu thiết kế các nhau, nhóm đã quyết định chon thiết kế mô hình cuối cùng như hình dưới đây để có thể đáp ứng được các yêu cầu đặt ra, thực hiện các tác vụ mà robot cần làm.



Hình 3.14. Mô hình robot thiết kế dùng SolidWorks

3.1.5. Quan sát mô hình mô phỏng

Cùng với quá trình thiết kế mô hình robot 3D, để đảm bảo mô hình có thể hoạt động đúng khi quan sát trên phần mềm Rviz, một plugin được hỗ trợ bởi SolidWorks mang tên sw_urdf_exporter được sử dụng để chuyển đổi thành file URDF, hỗ trợ việc quan sát robot trong các tác vụ của ROS sau này.



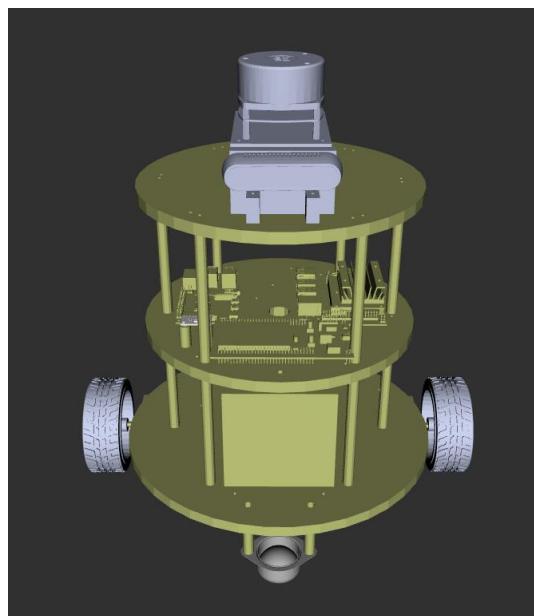
Hình 3.15. Trang cấu hình thông số để tạo file urdf từ mô hình thiết kế

Với các kích thước và chi tiết đã viết, plugin hỗ trợ tính ra khối lượng và các thông số động lực học của các thành phần robot. Sau đó, nhóm thực hiện cấu hình liên kết các thành phần với nhau thông qua các “link” (tên thành phần) và “joint” (khớp nối giữa hai thành phần) cũng như vị trí gốc của chúng. Yêu cầu cấu hình “link” và “joint” rất khắc khe vì việc này sẽ ảnh hưởng đến hệ trực dữ liệu thu được từ cảm biến và việc chuyển đổi hệ trực thông qua package tf của ROS được sử dụng sau này.

Bảng 3.1. Bảng thông tin kết nối các thành phần trong mô hình robot

Tên thành phần	Tên “link”	Kết nối với “link”	Tên “joint”
Khung chính	base_link		
Bánh xe trái	left_wheel_link	base_link	left_wheel_joint
Bánh xe phải	right_wheel_link	base_link	right_wheel_joint
Bánh dẫn động	caster_wheel_link	base_link	caster_wheel_joint
Cảm biến IMU	imu_link	base_link	imu_joint
Cảm biến Lidar	lidar_link	base_link	lidar_joint
Cảm biến Camera	realsense_link	base_link	realsense_joint

Sau quá trình chỉnh sửa cấu hình trên ROS, mô hình quan sát thu được như hình dưới đây, đáp ứng đúng các hệ trực yêu cầu của dữ liệu thu được từ cảm biến thông qua robot thực tế.



Hình 3.16. Mô hình quan sát trong phần mềm Rviz

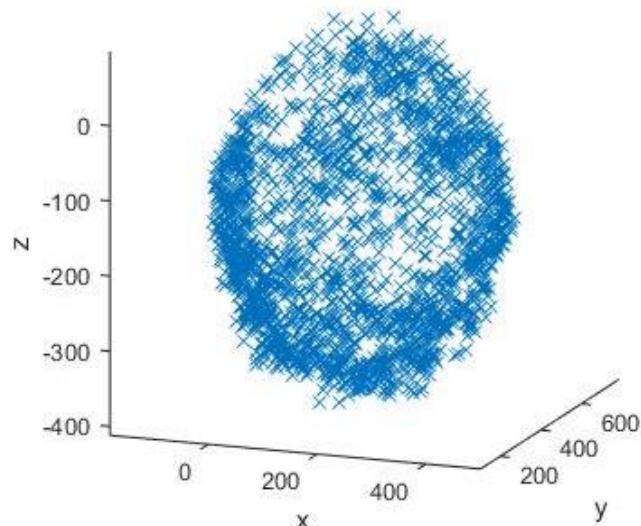
Một yêu cầu quan trọng trong các tác vụ của robot là việc tải lên thông số robot_description của mô hình quan sát. Việc này được thực hiện thông qua file thesis_robot_remote.launch, file sẽ được gọi tại mỗi tác vụ cần quan sát mô hình trên ROS, là nơi chứa cách thức gọi thông số của mô hình và sử dụng trong Rviz.

3.2. Xây dựng phần mềm trên mạch điều khiển

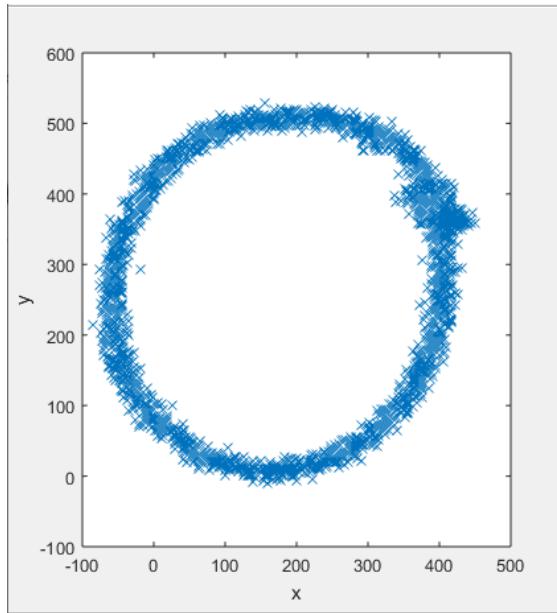
3.2.1. Tính toán Odometry

3.2.1.1. Hiệu chỉnh Magnetometer

Để lấy được các mẫu giá trị đo cho việc calib, xoay IMU 360 độ quanh tất cả trục là đường thẳng đi qua tâm trong lúc tất cả các thiết bị trên robot đều đang vận hành để có thể mô phỏng được gần nhất nhiều trong lúc robot vận hành. Do robot trong đè tài chỉ di chuyển trên mặt phẳng, nên chủ yếu tập trung lấy mẫu khi cho xoay quanh trục z của b-frame. Các mẫu giá trị từ trường trên ba trục b-frame được đưa vào phần mềm Matlab để thực hiện calib. Khi dùng Matlab vẽ các điểm (mx, my, mz) này trong không gian 3D, ta thu được mặt ellipsoid như Hình 3.17. Khi vẽ các điểm này trên mặt phẳng Oxy (mặt phẳng xoay của robot), ta thu được một đường ellipse.

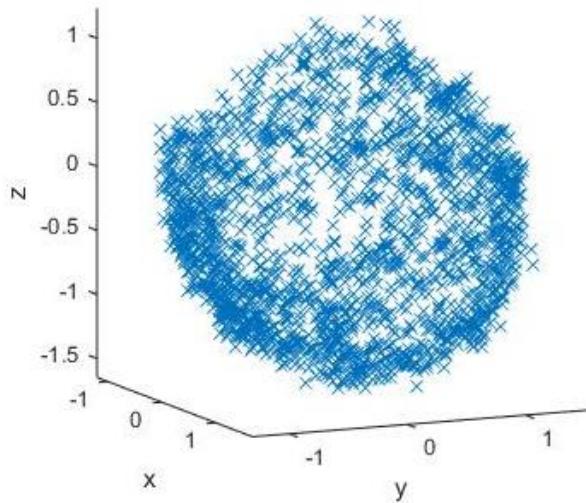


Hình 3.17 Quỹ tích các mẫu giá trị từ trường trước khi hiệu chỉnh

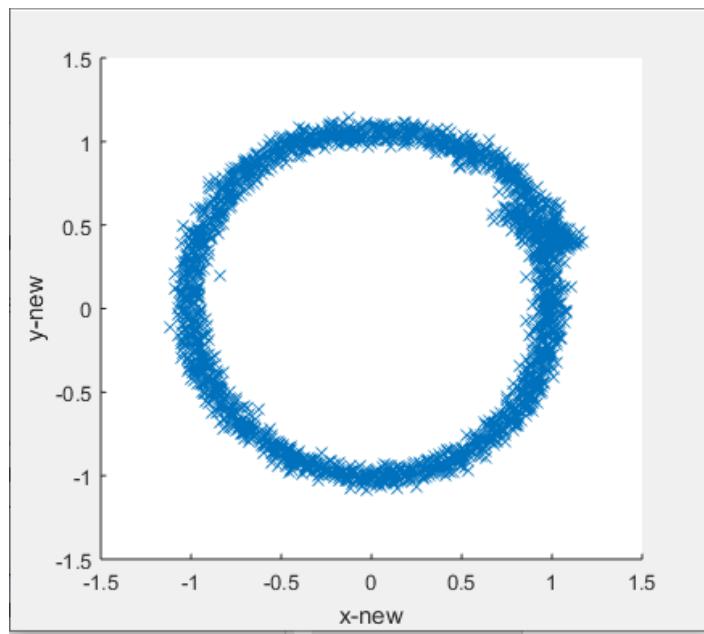


Hình 3.18 Quỹ tích các mẫu giá trị từ trường trên mặt phẳng Oxy trước khi hiệu chỉnh

Sau khi tìm được thông số mặt ellipsoid bằng phương pháp ellipsoid fit với code Matlab, ta có thể thu được ma trận \mathbf{A}, \mathbf{b} và dùng nó cho việc hiệu chỉnh magnetometer trong chương trình firmware đọc giá trị cảm biến. Lấy mẫu các giá trị từ trường sau hiệu chỉnh theo như cách trên và vẽ ra trong Matlab, ta thu được một mặt cầu có tâm gần gốc tọa độ và bán kính là 1. Xét trên mặt phẳng Oxy, các điểm này nằm trên một đường tròn có tâm ở gốc tọa độ và bán kính là 1.

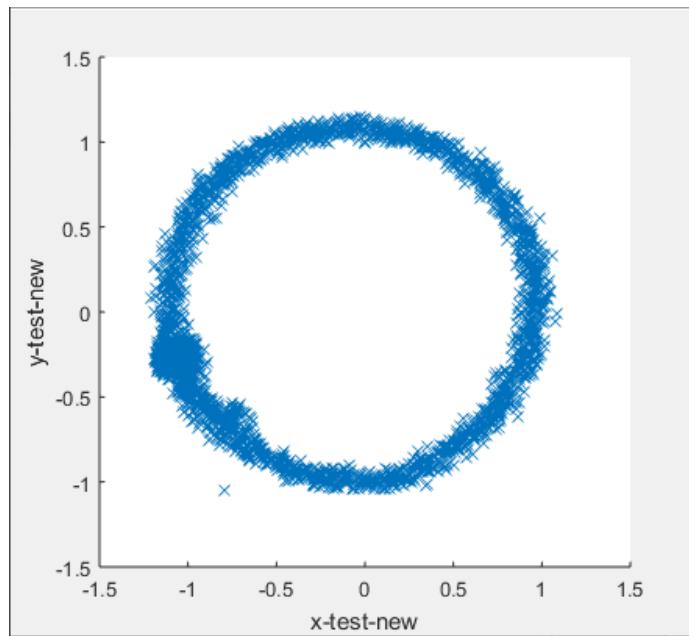


Hình 3.19 Quỹ tích các mẫu giá trị từ trường sau khi hiệu chỉnh



Hình 3.20 Quỹ tích các mẫu giá trị từ trường trên mặt phẳng Oxy sau khi hiệu chỉnh

Cuối cùng, thực hiện lấy mẫu thêm một tập giá trị mới, khác với tập được mang đi hiệu chỉnh vừa rồi. Sau khi qua phương trình calib với hai ma trận \mathbf{A}, \mathbf{b} có được vừa rồi, vẽ các mẫu này ra trong mặt phẳng Oxy, ta thu được đường tròn bán kính là 1, tâm tại gốc tọa độ như Hình 3.21. Lúc này, ta hoàn tất quá trình hiệu chỉnh.



Hình 3.21 Quỹ tích các mẫu giá trị từ trường kiểm thử trên mặt phẳng Oxy sau khi hiệu chỉnh

3.2.1.2. Kết hợp cảm biến với bộ lọc Madgwick

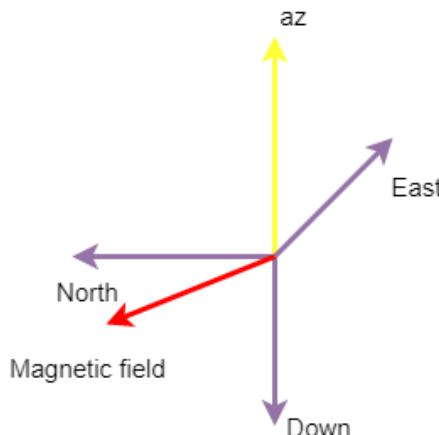
Thuật toán RTAB-Map cần đến thông tin về hướng và vị trí của robot ở dạng Odometry để thực hiện các tác vụ vẽ bản đồ và tự hành. Các thông tin này có thể dễ dàng được tính toán nếu ta có được giá trị đọc của encoder và giá trị góc xoay từ IMU chính xác.

Để tính toán góc xoay của IMU, cũng là góc xoay của robot, theo lý thuyết, ta có thể dùng giá trị vận tốc xoay ${}^b\omega$ kết hợp với thời gian lấy mẫu Δt . Hướng của robot so với gốc tọa độ sẽ được tìm bằng cách cộng dồn các giá trị $\Delta\alpha$ sau mỗi mẫu.

$$\Delta\alpha = {}^b\omega \Delta t$$

Tuy nhiên, việc cộng dồn giá trị sẽ tích lũy sai số của gyroscope sau mỗi lần lấy mẫu, và từ từ hình thành độ trôi ngày càng lớn (drift).

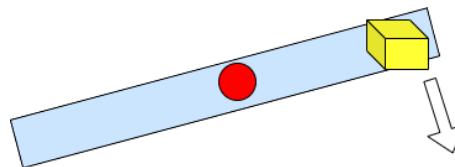
Magnetometer đo được từ trường trong b-frame, từ đó ta xác định được hướng tương đối của vector từ trường \vec{m} so với robot. Vector từ trường này không chỉ theo hướng Bắc địa lý, mà là hướng Bắc từ, và nằm trong mặt phẳng chứa vector North và Down (lần lượt là trục x và z của n-frame) do độ từ thiên và độ từ khuynh.



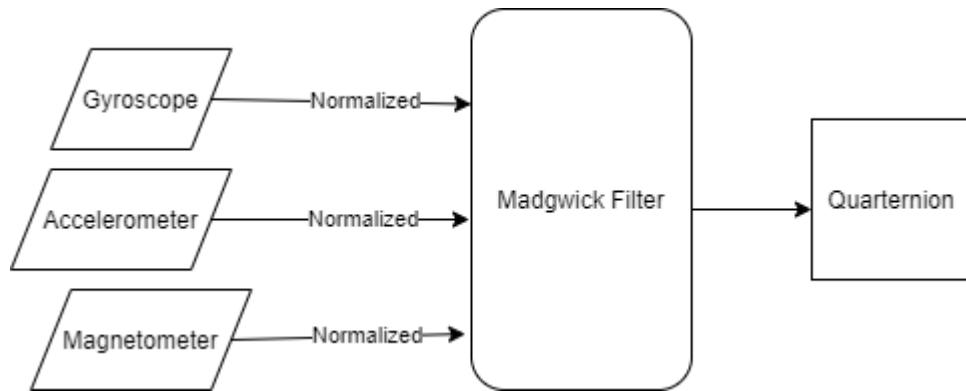
Hình 3.22 Hướng của robot so với n-frame

Các giá trị giá tốc đo được từ accelerometer giúp ta xác định được hướng của trọng lực (trục z của n-frame), kết hợp với \vec{m} , vì vector từ trường \vec{m} chỉ nằm trong mặt phẳng xOz của n-frame, ta xác định được vector từ trường trong n frame.

Dù đã qua hiệu chỉnh, magnetometer vẫn rất nhạy cảm với nhiễu. Ngoài ra, khi accelerometer không được đặt ở tâm xoay của robot, nó sẽ đo được giá tốc trên trục x, y khi robot chỉ xoay quanh trục z như Hình 3.23. Khi magnetometer đo được một sự thay đổi trong từ trường, gyroscope sẽ được dùng cho việc xác định xem có phải nó đến từ sự xoay của robot không, hay là do nhiễu hoặc yếu tố khác. Với các yếu tố nêu trên, sự kết hợp của Encoder, Accelerometer, Gyroscope và Magnetometer sẽ giúp tính toán được chính xác Odometry của robot.



Hình 3.23 Ảnh hưởng đối với accelerometer khi không nằm tại tâm xoay robot



Hình 3.24 Sơ đồ khối kết hợp cảm biến

Các giá trị đọc của 3 cảm biến Accelerometer, Gyroscope và Magnetometer sau khi được hiệu chỉnh và chuẩn hóa về khoảng [-1 1] sẽ được đưa vào bộ lọc Madgwick để dự đoán quaternion biến đổi b-frame sang n-frame. Bộ lọc này được viết dưới ngôn ngữ lập trình C trong chương trình vi điều khiển. Các thông số bộ lọc cần thiết được chọn là β – hệ số tốc độ phân kỳ của $q_{\omega,t}$ (phản tử quaternion tính từ gyroscope) và Δt – chu kỳ lấy mẫu được chọn như Bảng 3.2 dưới đây.

Bảng 3.2 Bảng thông số chọn cho bộ lọc Madgwick

β	0.5
Δt	60 ms

3.2.1.3. Tính Odometry

Odometry là các thông tin về vị trí, hướng và vận tốc của robot mà thuật toán RTAB-Map cần cho các tác vụ vẽ bản đồ, định vị và điều hướng. Các thông tin này dùng tham chiếu là vị trí và hướng ban đầu của robot, và cũng là hệ tọa độ bản đồ, được ký hiệu là O-frame.

Với quaternion của b-frame so với n-frame tìm được nhờ bộ lọc Madgwick, ta có thể tính được các góc Euler Roll, Pitch, Yaw giữa b-frame và n-frame lần lượt như sau:

$$\phi = \arctan2(2(q_1q_2 + q_3q_4), q_1^2 - q_2^2 - q_3^2 + q_4^2)$$

$$\theta = \arcsin(-2(q_1q_3 - q_2q_4))$$

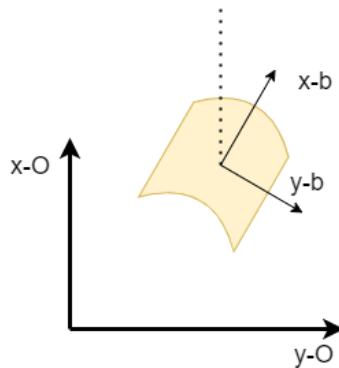
$$\psi = \arctan2(2(q_2q_3 + q_1q_4), q_1^2 + q_2^2 - q_3^2 - q_4^2)$$

Ở mỗi lần thực hiện tính toán Odometry (chu kỳ $T_{odom}=100$ ms), hiệu giá trị góc Yaw giữa 2 lần tính toán là sự thay đổi góc của robot. Vận tốc xoay của robot được tính như sau:

$$\bar{\omega} = \Delta Yaw * T_{odom}$$

Vì robot trong đê tài chỉ di chuyển theo trục x, nên vận tốc tịnh tiến \bar{v} cũng là vận tốc chuyển động theo trục x. \bar{v} có thể được tính từ vận tốc đo ở hai bánh \bar{v}_l, \bar{v}_r nhờ công thức Differential Drive. Khoảng dịch chuyển Δs của robot trong mỗi khoảng thời gian T_{odom} được tính dựa vào \bar{v} :

$$\Delta s = \bar{v} * T_{odom}$$



Hình 3.25 Robot trong O-frame

Như Hình 3.25, ta dễ dàng tính được vị trí của robot trong O-frame được tính dựa trên α và Δs bằng cách cộng dồn các giá trị như sau, với vận tốc theo trục y-b là 0, và α là góc giữa trục x của b-frame và trục x của O-frame:

$$x = x + \Delta s * \cos(\alpha) \quad y = y + \Delta s * \sin(\alpha)$$

Với góc α được tính bằng cách cộng dồn các giá trị Δyaw lại với nhau.

$$\alpha = \alpha + \Delta yaw$$

Vì robot có các góc pitch, roll không thay đổi, vậy nên quaternion từ b-frame sang O-frame có thể được tính chỉ từ góc Yaw α .

3.2.2. Điều khiển tốc độ động cơ

Việc điều khiển tay để robot di chuyển và vẽ bản đồ, hay cho việc Navigation, cũng như Human Following đều cần sự tương tác giữa máy tính nhúng và vi điều khiển. Máy tính nhúng sẽ gửi giá trị vận tốc tịnh tiến v và vận tốc xoay ω qua cổng nối tiếp bằng giao thức rosserial. Do robot được thiết kế theo mô hình Differential Drive, ta sẽ tính toán được giá trị vận tốc riêng cho từng bánh v_r, v_l để đạt được v và ω .

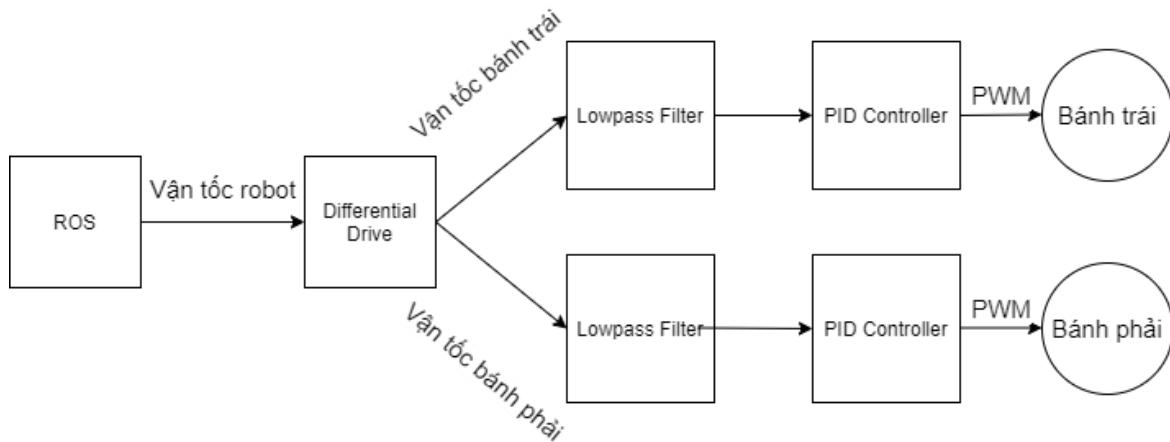
Khi có được vận tốc riêng cho hai bánh, để tài sử dụng thuật toán PID để xuất được xung PWM chính xác cho driver cầu H, với mục tiêu chính là hạn chế tối thiểu sai số xác lập e_{xl} và độ vọt lố POT. Vì hai yếu tố này gây ra sai số lớn trong khâu tính Odometry gửi về ROS, dẫn đến sai số lớn trong việc vẽ bản đồ và định vị robot trong bản đồ.

Bảng 3.3 Bảng thông tin tín hiệu bộ điều khiển PID

Tín hiệu đặt	$r(t)$	v_r hoặc v_l	Vòng/phút
Tín hiệu hồi tiếp	$v(t)$	\bar{v}_l hoặc \bar{v}_r	Vòng/phút
Tín hiệu điều khiển	$u(t)$	Chu kỳ điều khiển xung PWM	%

Giá trị vận tốc mong muốn trước khi đưa vào bộ điều khiển sẽ được cho qua bộ lọc thông thấp để tạo độ trễ, nhằm tránh tình trạng thuật toán trên ROS gửi vận tốc đặt có độ thay

đổi quá lớn, tương đương $e(k)$ lớn, dễ khiến cho tín hiệu điều khiển $u(k)$ bị vọt lố. Do robot được lái bằng hai bánh, sự vọt lố sẽ khiến cho robot bị lệch hướng, ảnh hưởng đến việc vẽ bản đồ và điều hướng.



Hình 3.26 Sơ đồ khái niệm điều khiển vận tốc hai bánh xe

Encoder nhóm sử dụng có thể xuất được khoảng 1500 xung/vòng khi được đọc ở chế độ x4, mà tốc độ di chuyển trung bình của robot vào khoảng 20 cm/s, vì vậy nhóm lựa chọn chu kỳ đọc Encoder $T_{encoder} = 60$ ms để sau mỗi chu kỳ, giá trị counter đọc về từ Encoder thay đổi khoảng 57 xung, nhờ đó, gai nhiễu Encoder sẽ không làm thay đổi quá lớn giá trị vận tốc của động cơ khi tính toán. Các giá trị cho các thông số của bộ lọc thông thấp, bộ điều khiển PID và các chu kỳ điều khiển được chọn như trong bảng dưới đây. Trong đó, các giá trị K_p, K_i, K_d được tinh chỉnh theo phương pháp sau đây².

- Bước 1: Đặt K_i, K_d ban đầu bằng 0. Tăng K_p đến khi POT đạt 10%.
- Bước 2: Tăng K_i đến khi $e_{xl} = 0$.
- Bước 3: Tăng K_d đến khi POT đạt nhỏ nhất.

Bảng 3.4 Bảng giá trị thông số điều khiển tốc độ động cơ

Tần số cắt bộ lọc thông thấp	ω_{LPF}	2.7 Hz
Chu kỳ điều khiển PID	T	60 ms
Chu kỳ đọc encoder	$T_{encoder}$	60 ms
Tần số xung PWM	T_{PWM}	10 khz

² Nguồn: Thầy Huỳnh Thái Hoàng, Bộ môn Điều khiển, Tự động, trường ĐH Bách Khoa TP.HCM

Hệ số khâu P	Kp	4.5
Hệ số khâu I	Ki	9
Hệ số khâu D	Kd	0.015

3.2.3. Chu kỳ tác vụ

Bảng 3.5 Bảng giá trị chu kỳ thực hiện tác vụ trong chương trình vi điều khiển

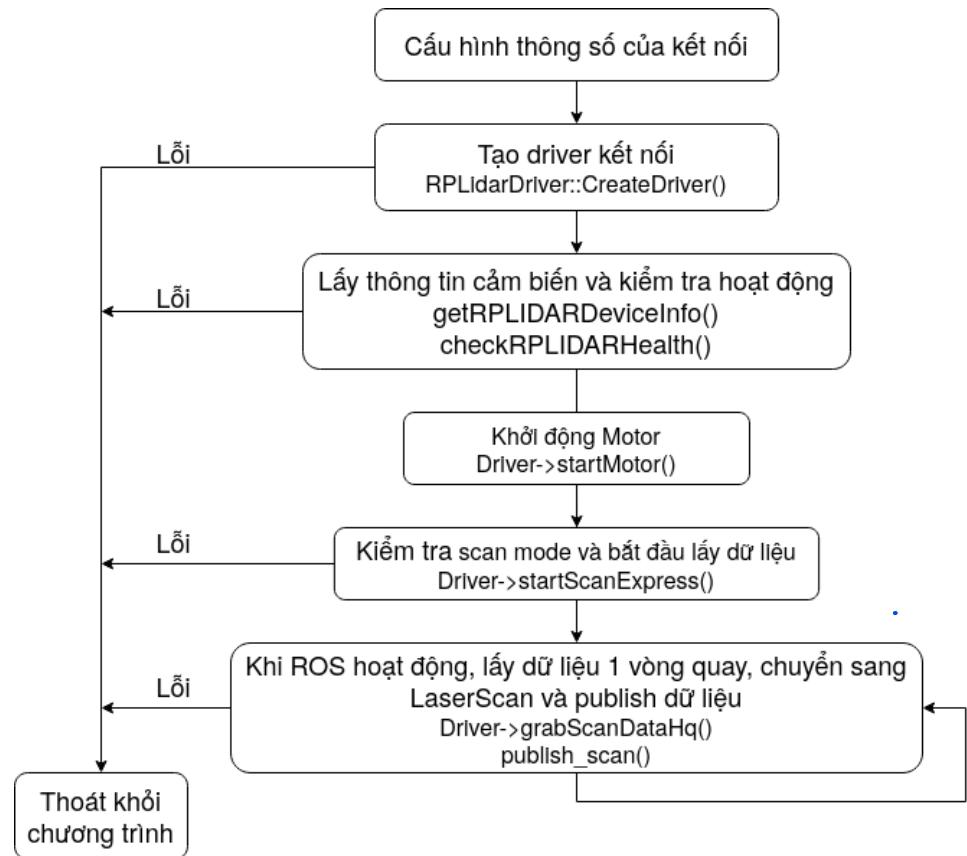
Chu kỳ điều khiển PID	T	60 ms
Chu kỳ đọc encoder	$T_{encoder}$	60 ms
Chu kỳ lấy mẫu quaternion từ bộ lọc Magdwick	$T_{Quaternion}$	60 ms
Chu kỳ tính toán odometry và publish topic /odom	T_{odom}	100 ms
Chu kỳ broadcast TF transform	T_{TF}	100 ms

3.3. Đọc dữ liệu cảm biến trên nền tảng ROS

3.3.1. Đọc dữ liệu Lidar

Cảm biến Lidar được nhóm sử dụng là RPLidar A1 của công ty Slamtec, hỗ trợ quét 360° như đã trình bày ở mục 3.1.1. Nhóm sử dụng công cụ SDK được hỗ trợ sẵn để giao tiếp với cảm biến và viết ROS package mang tên rplidar_ros phục vụ việc đọc scan data và chuyển đổi sang dạng dữ liệu LaserScan được tích hợp trong ROS, phục vụ các tác vụ sau này. Package này sử dụng các package phụ thuộc gồm roscpp, rosconsole, sensor_msgs và std_srvs.

Chuẩn giao tiếp được sử dụng để đọc dữ liệu scan là USB-Uart, tiến hành viết code như lưu đồ giải thuật dưới đây sử dụng SDK của RPLidar A1.



Hình 3.27. Lưu đồ giải thuật để đọc cảm biến Lidar

Sau khi hoàn thành package rplidar_ros và kết nối thành công với cảm biến rplidar, tiến hành viết file launch phục vụ việc kết nối có cấu hình các thông số kết nối, có tên là thesis_robot_rplidar.launch nằm trong package thesis_robot_bringup. Việc này hỗ trợ nhiều cho đề tài cần cấu hình kết nối, chế độ đọc dữ liệu scan khi có sự thay đổi của các tác vụ hoặc yêu cầu bắt buộc. Các thông số cấu hình kết nối được trình bày ở bảng dưới đây.

Bảng 3.6. Bảng giá trị thông số cấu hình cho giao tiếp giữa ROS và Lidar

Thông số	Giá trị mặc định	Giá trị cấu hình	Chú thích
serial_port	/dev/ttyUSB0	/dev/ttyUSB0	Port kết nối
serial_baudrate	115200	115200	Tốc độ baudrate
frame_id	laser_frame	lidar_link	“link” trong mô hình
inverted	false	false	Đảo chiều dữ liệu

angle_compensate	true	true	Scale lại dữ liệu
scan_mode		Boost	Lấy 8000 mẫu/s

Như câu hình trên, khi giao tiếp với cảm biến Lidar, nhóm thu được dữ liệu quét 360° về khoảng cách của môi trường xung quanh robot.

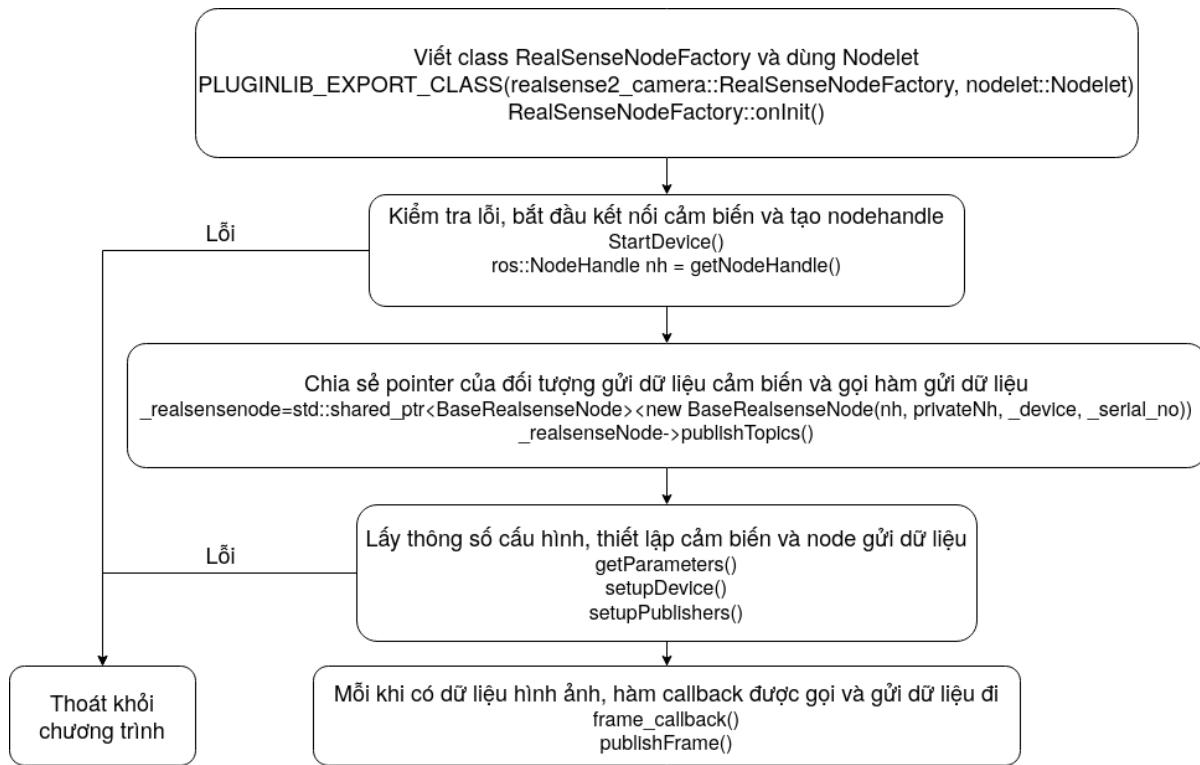
Bảng 3.7. Bảng thông tin về dữ liệu của Lidar trong ROS

Đặc tính	Giá trị
Topic	/scan
Dạng message	sensor_msgs/LaserScan
Tần số truyền	7Hz
Độ phân giải	0.5°
Khoảng cách đo	$0.15m < d < 3.5m$

3.3.2. Đọc dữ liệu Camera

Cảm biến Camera được nhóm sử dụng là Realsense Camera D435i, được sản xuất bởi Intel, như trình bày ở mục 3.1.1. Sản phẩm này cũng được hỗ trợ công cụ SDK, có đầy đủ các API phục vụ việc đọc dữ liệu các loại ảnh từ camera với tần số cao có thể lên đến 30Hz.

Cùng với các API đã được cung cấp của cảm biến, nhóm có sử dụng công cụ là package nodelets của ROS, đây là công cụ hỗ trợ chạy nhiều tiến trình (processes) trong cùng một thiết bị điện tử đang sử dụng ROS, sử dụng kỹ thuật chia sẻ pointer thay vì truyền lượng lớn dữ liệu gốc như hình ảnh hay pointcloud thông qua pluginlib (công cụ giúp sử dụng nodelets kể cả khi trong quá trình runtime của code). Để sử dụng công cụ hỗ trợ nodelets, sử dụng tính chất override của class để viết hàm `onInit()`, trong hàm này chứa code mà nhóm sẽ dùng để đọc dữ liệu từ cảm biến và publish trên nền tảng ROS. Tiến hành viết package `realsense2_camera` để sử dụng cảm biến camera, sử dụng các package phụ thuộc chính gồm `roscpp`, `image_transport`, `cv_bridge`, `nodelet`, `std_msgs` và thư viện SDK `librealsense2`. Chuẩn giao tiếp được sử dụng là USB.



Hình 3.28. Lưu đồ giải thuật đọc cảm biến Camera trên nền tảng ROS

Với nhu cầu sử dụng cảm biến Camera để thu thập ảnh độ sâu và ảnh màu của môi trường, nhóm tiến hành cấu hình thông số phù hợp để dữ liệu thu được ở tần số cao, độ phân giải hình ảnh đảm bảo đủ chất lượng để thực hiện tác vụ vẽ bản đồ và chỉ sử dụng những topic cần thiết nhằm tối ưu bộ nhớ. Những thông số cấu hình cần thiết cho đề tài được mô tả như Bảng 3.8.

Bảng 3.8. Một số thông số cấu hình camera thông qua ROS

Thông số	Giá trị cấu hình	Chú thích
depth_width	640	Chiều dài ảnh depth
depth_height	480	Chiều cao ảnh depth
enable_depth	true	Có publish ảnh depth
depth_fps	15	Tần số xuất ảnh depth
color_width	640	Chiều dài ảnh màu
color_height	480	Chiều cao ảnh màu
enable_color	true	Có publish ảnh màu

Bảng 3.9. Một số thông số cấu hình camera thông qua ROS (tiếp theo)

color_fps	15	Tần số xuất ảnh màu
enable_pointcloud	false	Không sử dụng pointcloud từ camera
align_depth	true	Căn chỉnh ảnh depth theo ảnh màu

Sau khi kết nối thành công với cảm biến với các thông số cấu hình lựa chọn, thực hiện viết file launch phục vụ việc khởi động kết nối có tên là thesis_robot_realsense.launch nằm trong package thesis_robot_bringup. Như trên, khi giao tiếp với cảm biến Camera, các topic trên nền tảng ROS chứa dữ liệu thông qua các message được truyền đạt với tần số 15Hz gồm:

- /camera/color/image_raw: ảnh màu của môi trường dạng sensor_msgs/Image chứa 3 kênh màu
- /camera/color/camera_info: thông tin về thời gian thu thập ảnh màu
- /camera/aligned_depth_to_color/image_raw: ảnh độ sâu của môi trường dạng sensor_msgs/Image với mỗi kênh chứa độ sâu giống nhau (3 kênh)
- /camera/aligned_depth_to_color/camera_info: thông tin về thời gian thu thập ảnh độ sâu

3.3.3. Giao tiếp rosserial giữa ROS và mạch STM32F4

Sau khi đã hoàn thành phần mềm nhúng trên mạch điều khiển ở mục 3.2, nhóm thực hiện kết nối giữa máy tính nhúng và mạch điều khiển thông qua chuẩn giao tiếp rosserial đã được đóng gói thành một ROS package được trình bày ở mục 2.1.4. Việc này giúp đồng bộ dữ liệu nhúng trên mạch điều khiển với các topic trên ROS, hoạt động như một node trong mô hình ROS, tiết kiệm thời gian và thuận lợi cho ROS vận hành. Để thực hiện được việc này, một lớp đệm giữa mạch điều khiển và ROS theo mô hình client-server được tích hợp vào ROS dưới dạng package rosserial_python mà nhóm đã viết với ngôn ngữ python để dễ dàng tiếp cận. Các package phụ thuộc gồm rospy, python3-serial, rosserial_msgs.

Với mạch điều khiển hoạt động như một server chờ lệnh (request), package rosserial_python thông qua chuẩn giao tiếp USB-Uart với baudrate 115200, thực hiện lần lượt các bước sau:

- Gửi lệnh đi và nhận về thông tin của publisher và subscriber trên mạch điều khiển
- Lấy thông tin về message, service của publisher và subscriber từ package phụ thuộc rospy
- Đồng bộ thông tin với ROS master
- Sử dụng các topic trên mạch điều khiển một cách hợp thức hóa trên nền tảng ROS

Sau khi đã kết nối thành công rosserial, tiến hành viết file launch với các thông số cấu hình lựa chọn phục vụ việc kết nối trở nên nhanh gọn thông qua một lần gọi trên ROS, file mang tên thesis_robot_rosserial.launch thuộc package thesis_robot_bringup.

Bảng 3.10. Các topic giao tiếp giữa mạch điều khiển và ROS thông qua rosserial

	Tên topic	Dạng message	Tần số	Ý nghĩa
Publisher	/odometry	nav_msgs/Odometry	10Hz	Thông tin về vị trí robot
	/imu	sensor_msgs/Imu	10Hz	Thông tin về quaternion, vận tốc góc, gia tốc
Subscriber	/cmd_vel	geometry_msgs/Twist	Mỗi lần gửi	Vận tốc điều khiển

3.3.4. Khởi tạo kết nối đọc dữ liệu từ cảm biến và giao tiếp rosserial

Sau khi đã cấu hình thích hợp cho từng package để đọc cảm biến Camera và Lidar cũng như bắt đầu giao tiếp rosserial và đóng gói lại thành các file launch trên nền tảng ROS, nhóm thực hiện tích hợp các file lại thành một file launch duy nhất mang tên thesis_robot.launch nằm trong package thesis_robot_bringup. File này có chức năng gọi lần lượt các file launch thành phần để khởi tạo và duy trì kết nối trong một tab của terminal. Để việc bắt đầu kết nối không xảy ra lỗi cũng như xen lấn lên nhau thì nhóm sử dụng package hỗ trợ của ROS là

timed_roslaunch có chức năng hoãn việc chạy file launch sau một thời gian nhất định, từ đó giúp cho việc kết nối không xảy ra lỗi khi khởi động và ổn định trong quá trình sử dụng.

```
<node    pkg="timed_roslaunch"    type="timed_roslaunch.sh"    name="[name]_delay"
args="[seconds to delay] [package] [launch file]"/>
```

Với sự hỗ trợ từ package timed_roslaunch, nhóm phân chia việc khởi tạo kết nối theo thứ tự như sau, đảm bảo kết thúc khởi tạo mỗi kết nối mới bắt đầu chạy kết nối tiếp theo:

- Đọc dữ liệu Camera qua file thesis_robot_realsense.launch: Khởi tạo kết nối đầu tiên, không sử dụng timed_roslaunch
- Bắt đầu giao tiếp rosserial qua file thesis_robot_rosserial.launch: Khởi tạo thứ hai với thời gian trì hoãn là 10s của package timed_roslaunch
- Đọc dữ liệu Lidar qua file thesis_robot_rplidar.launch: Khởi tạo kết nối cuối cùng, sử dụng package timed_roslaunch với thời gian trì hoãn là 15s.

3.4. Tác vụ điều khiển robot trên ROS

Trên nền tảng ROS, package thesis_robot_teleop được thực hiện với mục tiêu gửi vận tốc điều khiển xuống mạch điều khiển nhằm đưa robot di chuyển đến các điểm mong muốn. Package này được viết bằng ngôn ngữ C++, điều khiển thông qua các phím nhấn để gửi vận tốc và được giới hạn phù hợp với mô hình robot. Vận tốc điều khiển gồm hai thành phần là vận tốc thẳng và vận tốc xoay, sau khi mạch điều khiển nhận được thì sẽ chuyển đổi sang vận tốc điều khiển hai động cơ theo mô hình differential drive.

Tiến hành viết package thesis_robot_teleop với các điều kiện về vận tốc phù hợp đặc tính robot và môi trường được nhóm lựa chọn như sau:

Bảng 3.11. Bảng thông số cấu hình điều khiển vận tốc trên ROS

Loại vận tốc	Thông số	Giá trị
Vận tốc thẳng	Step size	0.075 m/s
	Vận tốc tối đa	0.3 m/s
	Vận tốc tối thiểu	-0.3 m/s
Vận tốc xoay	Step size	0.075 rad/s
	Vận tốc tối đa	0.5 rad/s
	Vận tốc tối thiểu	-0.5 rad/s

Các phím điều khiển được lựa chọn gồm:

- Phím w/s: Đi tới/ Đi lùi
- Phím a/d: Xoay trái/ Xoay phải
- Phím s/space: Dừng lại

Các thông số ROS của package như sau:

Bảng 3.12. Bảng thông tin ROS của package điều khiển vận tốc

Thông số	Giá trị
Tên topic	/cmd_vel
Dạng message	geometry_msgs/Twist
Tần số	Mỗi lần nhấn phím

```

hxnhgia@CLAW:~$ rosrun thesis_robot teleop thesis_robot teleop
[ INFO] [1625559822.118096786]: 
Control Thesis Robot
-----
Moving around:
      w
a      s      d
      x
w/x : increase/decrease linear velocity ( 0.075/time)
a/d : increase/decrease angular velocity (0.075/time)

space key, s : force stop
-----
CTRL-C to quit

Current:      Linear_vel = 0.0 (cm/s)          Angular_vel = 0.00 (rad/s)
Current:      Linear_vel = 0.0 (cm/s)          Angular_vel = 0.00 (rad/s)

```

Hình 3.29. Khi chạy package điều khiển vận tốc trên ROS

3.5. Tính toán odometry từ dữ liệu Lidar

Trong quá trình thực hiện vẽ bản đồ cũng như định vị robot, nhóm phát hiện nhược điểm không thể khắc phục của IMU là nhiễu từ trường. Nhược điểm này làm robot không thể di chuyển gần những vật có từ trường và vẽ bản đồ sai cũng như định vị sai. Nhóm tiến hành tạo ra dữ liệu odometry từ dữ liệu laser scan của Lidar thông qua thuật toán Scan Matching được trình bày ở mục 2.4.

Nhiệm vụ tính toán odometry từ dữ liệu của lidar được đóng gói thành package `laser_scan_matcher`. Package được cấu hình subscribe vào topic là `/scan` chứa dữ liệu `LaserScan` với tần số 7Hz và publish ra topic `/odom` mang dữ liệu Odometry của robot với tần số 5Hz. Khi package này hoạt động thì việc xuất odometry từ mạch điều khiển phải được dừng.

3.6. Tác vụ vẽ bản đồ

3.6.1. Cấu hình thuật toán xây dựng bản đồ

Việc xây dựng bản đồ của nhóm được thực hiện thông qua package `rtabmap_ros` với sự hỗ trợ từ thư viện `rtabmap`. sử dụng thuật toán RTAB-Map đã trình bày ở mục 2.2. Trước khi vẽ bản đồ cần cấu hình một số thông số và dữ liệu đầu vào của RTAB-Map sao cho phù hợp với robot và môi trường hoạt động.

Bảng 3.13. Một số thông số cấu hình RTAB-Map

Thông số	Giá trị	Ý Nghĩa
subscribe_depth	true	Đăng ký nhận dữ liệu ảnh depth
subscribe_scan	true	Đăng ký nhận dữ liệu scan
map_always_update	true	Cập nhật bản đồ kể cả khi không di chuyển
Icp/CorrespondenceRatio	0.3	Tỉ lệ được xem là có biến đổi giữa hai dữ liệu scan
Vis/MinInliers	15	Số đặc trưng nhỏ nhất cần trích xuất để nhận biết sự thay đổi
Vis/InlierDistance	0.1	Khoảng cách giữa hai điểm đặc trưng
Rtabmap/TimeThr	0	Thời gian để chuyển các điểm từ WM sang LTM

Bảng 3.14. Một số thông số cấu hình RTAB-Map (tiếp theo)

Rtabmap/MemoryThr	0	Không giới hạn số điểm chứa bên trong WM
Rtabmap/DetectionRate	5	Quá trình cập nhật bản đồ diễn ra với tần số 5Hz
Mem/RehearsalSimilarity	0.3	Tỉ lệ để xem 2 điểm trong STM là giống nhau để cập nhật trọng số
GridGlobal/MinSize	20	Kích thước nhỏ nhất của mỗi cạnh bản đồ
Grid/FromDepth	false	Bản đồ 2D được tạo ra nhờ cảm biến Lidar

Với các cấu hình như trên, package rtabmap_ros sẽ sử dụng các đầu vào và cung cấp các đầu ra theo yêu cầu vẽ bản đồ.

Bảng 3.15. Ngõ vào và ngõ ra của package rtabmap_ros

	Tên topic	Dạng message	Tần số	Ý Nghĩa
Subscriber	scan	sensor_msgs/LaserScan	7Hz	Lấy dữ liệu scan từ Lidar
	camera/color/image_raw	sensor_msgs/Image	15Hz	Lấy dữ liệu ảnh màu từ Camera
	camera/color/camera_info	sensor_msgs/CameraInfo	15Hz	Lấy dữ liệu về thời gian của ảnh
	camera/aligned_depth_to_color/image_raw	sensor_msgs/Image	15Hz	Lấy dữ liệu về ảnh depth từ Camera
	odom	nav_msgs/Odometry	10Hz	Lấy dữ liệu odometry tính toán được của robot
Publisher	map	nav_msgs/OccupancyGrid	4Hz	Bản đồ GridMap 2D vẽ được
	rtabmap/pointcloud	rtabmap_ros/MapData	4Hz	Bản đồ pointcloud 3D vẽ được

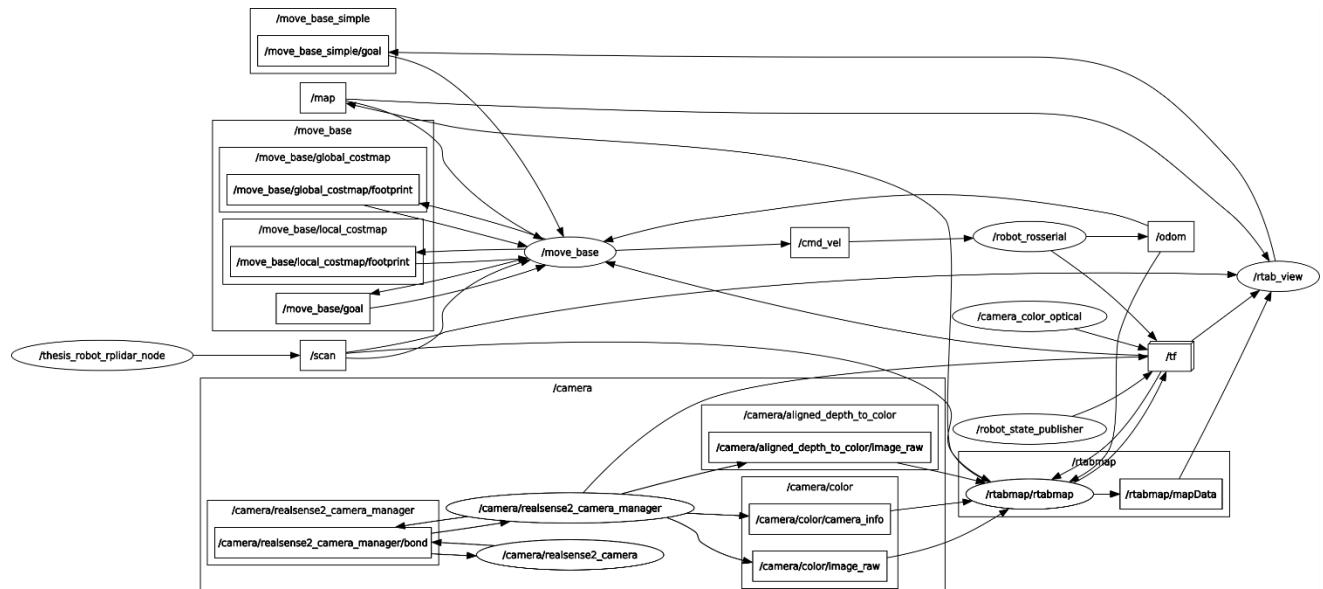
3.6.2. Khởi động vẽ bản đồ trên ROS

Sau khi đã lựa chọn các thông số cấu hình thích hợp của RTAB-Map, nhóm tiến hành viết file launch để vẽ bản đồ mang tên thesis_robot_rtabmap.launch nằm trong package

thesis_robot_bringup. File này có chức năng khởi động RTAB-Map thuộc package rtabmap_ros và thông số robot_description nằm trong file thesis_robot_remote.launch cũng như mở phần mềm quan sát Rviz nếu cần thiết sau khi đã đảm bảo có đầy đủ dữ liệu đầu vào là các subscriber được trình bày ở Bảng 3.15. Như vậy, trình tự thực hiện vẽ bản đồ sẽ gồm hai bước chính như sau:

- Chạy file khởi tạo kết nối cảm biến và rosserial có tên là thesis_robot.launch, có chức năng tạo kết nối và giữ kết nối tới các cảm biến và mạch điều khiển, duy trì việc thu thập dữ liệu và giao tiếp rosserial trong suốt quá trình vẽ bản đồ.
- Chạy file vẽ bản đồ tên là thesis_robot_rtabmap.launch để tải lên thông số cấu hình của robot cũng như rtab, khởi động thuật toán RTAB-Map và bắt đầu vẽ bản đồ.

Một cách tổng quan, việc kết nối giữa các node trong tác vụ vẽ bản đồ sử dụng RTAB-Map là khá phức tạp. Hình dưới đây minh chứng cho điều này.



Hình 3.30. Sự liên kết giữa các node thông qua topic trong tác vụ vẽ bản đồ

3.6.3. Lưu bản đồ 2D

Sau khi điều khiển robot di chuyển trong môi trường cần vẽ bản đồ và thu được bản đồ có độ chính xác tốt nhất thì cần lưu lại để phục vụ các tác vụ về sau. Việc lưu lại bản đồ được thực hiện thông qua lệnh (command) trên cửa sổ terminal.

“rosrun map_server map_saver [-f map_name]”

Trong đó map_name là tên bản đồ mà nhóm đặt tên. Sau khi lưu bản đồ lại thì thu được hai file mô tả bản đồ gồm file hình ảnh (.pgm) và file thông số (.yaml) của bản đồ.



Hình 3.31. File hình ảnh và file thông số của bản đồ 2D được vẽ

File hình ảnh mô tả không gian bị chiếm chỗ (occupancy) của môi trường. Theo hình ảnh trên, các pixel màu trắng là vùng không gian trống, các pixel màu đen là nơi có vật cản (như tường, bàn ghế, ...), các pixel màu xám là nơi chưa xác định. Đối với file thông số thì sẽ lưu trữ đường dẫn đến file hình ảnh, độ phân giải, vị trí gốc cũng như các ngưỡng của bản đồ.

3.7. Tác vụ định vị robot trong bản đồ

Sau khi đã vẽ được bản đồ và lưu lại, việc định vị robot trong bản đồ được tiến hành với package amcl. Package này sử dụng thuật toán AMCL được trình bày ở mục 2.3.3, thông qua ngôn ngữ C++ với mục tiêu xác định được vị trí (x, y, theta) của robot thông qua cảm biến về môi trường và bản đồ đã biết.

Bảng 3.16. Ngõ vào và ngõ ra của package amcl

	Tên topic	Dạng message	Tần số	Ý Nghĩa
Subscriber	map	nav_msgs/OccupancyGrid	10Hz	Bản đồ đã biết
	scan	sensor_msgs/LaserScan	7Hz	Dữ liệu scan từ Lidar
Publisher	amcl_pose	geometry_msgs/PoseWithCovarianceStamped	6Hz	Vị trí định vị của robot trong bản đồ

	particlecloud	geometry_msgs/PoseArray	6Hz	Tập hợp các particle của bộ lọc amcl
--	---------------	-------------------------	-----	--------------------------------------

Việc áp dụng và viết package amcl được thực hiện thông qua 5 bước chính gồm:

- Khởi tạo đối tượng cho class AmclNode

```
amcl_node_ptr.reset(new AmclNode());
```

- Cập nhật các thông số của thuật toán (được trình bày sau)
- Tạo node handle cùng 2 publisher và 3 service trên ROS, cấu hình các hàm callback

```
//2 publisher
pose_pub_ = nh_.advertise<geometry_msgs::PoseWithCovarianceStamped>("amcl_pose", 2, true);
particlecloud_pub_ = nh_.advertise<geometry_msgs::PoseArray>("particlecloud", 2, true);
//3 service
global_loc_srv_ = nh_.advertiseService("global_localization", &AmclNode::globalLocalizationCallback, this);
nomotion_update_srv_ = nh_.advertiseService("request_nomotion_update", &AmclNode::nomotionUpdateCallback, this);
set_map_srv_ = nh_.advertiseService("set_map", &AmclNode::setMapCallback, this);
```

- Khi nhận được map, hàm callback AmclNode::mapReceived() được gọi để khởi tạo bộ lọc amcl cùng vị trí khởi tạo ban đầu của robot

```
// Create the particle filter
pf_ = pf_alloc(min_particles_, max_particles_, alpha_slow_, alpha_fast_,
                (pf_init_model_fn_t)AmclNode::uniformPoseGenerator,
                (void *)map_);
updatePoseFromServer();
odom_ = new AMCL0dom();
```

- Khi nhận được dữ liệu từ Lidar, hàm callback AmclNode::laserReceived() được gọi để tính vị trí mới dùng bộ lọc amcl

```
tf2::toMsg(tf2::Transform::getIdentity(), ident.pose);
this->tf_->transform(ident, laser_pose, base_frame_id_);
odom_->UpdateAction(pf_, (AMCLSensorData*)&odata);
```

Các thông số của bộ lọc amcl được nhóm lựa chọn, sử dụng phù hợp với đặc tính của robot và môi trường hoạt động như bảng dưới đây.

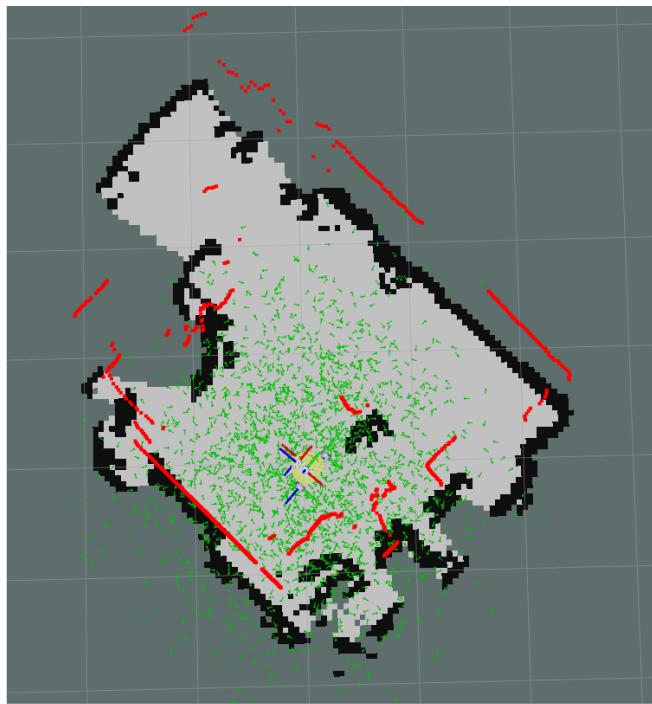
Bảng 3.17. Các thông số cấu hình bộ lọc AMCL

Thông số cấu hình	Giá trị
initial_pose_x	0.0
initial_pose_y	0.0

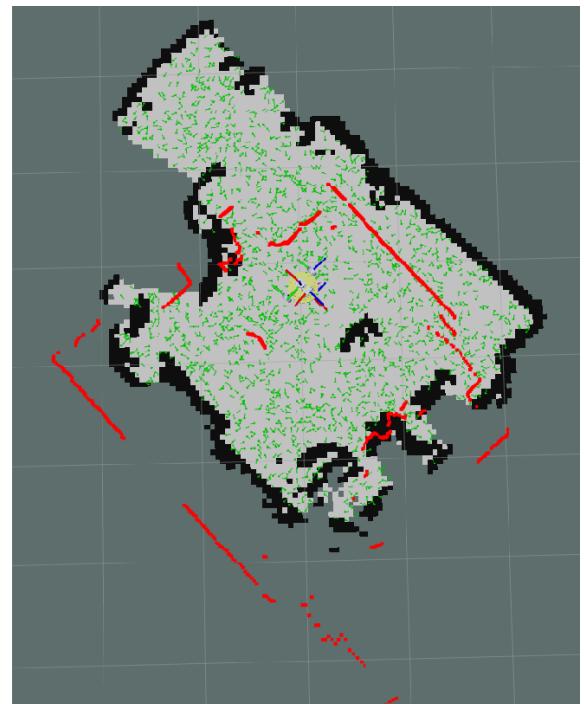
initial_pose_a	0.0
kdl_err	0.02
use_map_topic	true
base_frame_id	base_footprint
odom_frame_id	odom
scan	scan
min_particles	500
max_particles	3000
odom_model_type	diff

Sau khi hoàn thành package amcl, nhóm tiến hành thực hiện định vị cho robot trong một bản đồ đã vẽ. Tác vụ này đánh giá kết quả của việc kết hợp các package trên ROS và khả năng hoạt động của robot. Để tác vụ định vị có thể chạy và quan sát kết quả, nhóm thực hiện các bước sau:

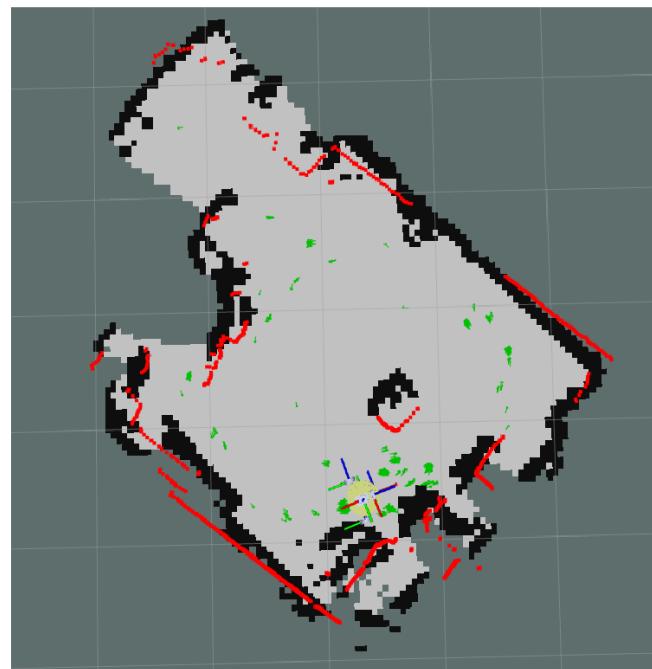
- Chạy file thesis_robot.launch để khởi tạo các kết nối cảm biến thu thập dữ liệu cần thiết cho tác vụ định vị của robot.
- Tải lên ROS thông số robot_description, đây là các thông số của mô hình quan sát của robot mà nhóm đã thiết kế trong mục 3.1.5. Thông qua việc này, plugin robot_model trong phần mềm Rviz có thông số quan sát của robot, từ đó giúp nhìn nhận và đánh giá hướng di chuyển cũng như tác động của môi trường lên robot thông qua việc quan sát robot. Việc này được thông qua việc thông qua một file đóng gói sẵn là thesis_robot_remote.launch.
- Tải lên bản đồ đã vẽ, đây là yêu cầu bắt buộc của tác vụ định vị. Có bản đồ, robot thu thập dữ liệu từ Lidar và dùng thuật AMCL để ước lượng vị trí robot trong bản đồ.
- Dùng package amcl để định vị thông qua thuật toán AMCL. Với đầu vào như trình bày ở bảng 3.13, đầu ra thu được vị trí của robot và tập hợp pointcloud vị trí trong bản đồ. Nhược điểm của package amcl là lần đầu tiên chạy thuật toán thì tập hợp pointcloud chưa xác định tốt vị trí và cần chạy service global_localization để tạo lại tập pointcloud ngẫu nhiên.



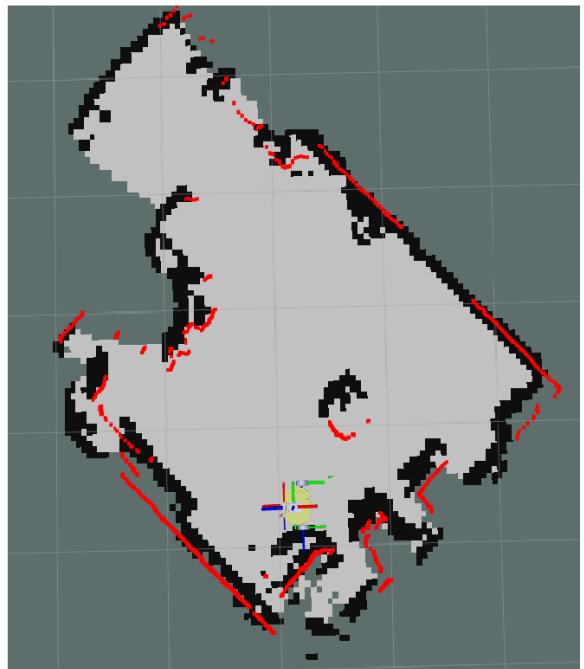
a) Vừa bắt đầu chạy bộ lọc AMCL



b) Chạy service global_localization



c) Đang xoay robot để tự định vị

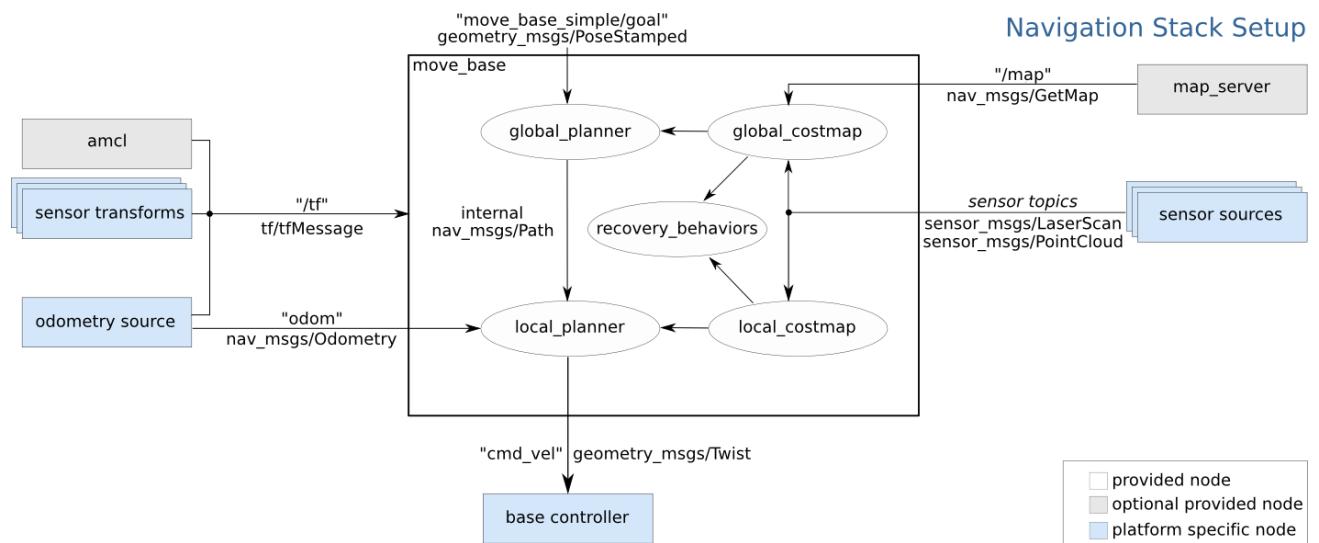


d) Hoàn thành quá trình định vị

Hình 3.32. Quá trình robot thực hiện tự định vị trên bản đồ với sự quan sát trên Rviz

3.8. Tác vụ điều hướng robot

Tác vụ này được thực hiện thông qua 2D Navigation Stack, một metapackage của ROS có chức năng lấy các thông tin dữ liệu về odometry, cảm biến và vị trí mục tiêu (goal pose) để tính toán ra vận tốc và gửi xuống mạch điều khiển để robot tự di chuyển. Điều kiện để chạy được Navigation Stack rất khắc khe, đầu tiên robot chạy trên nền tảng ROS có một tf tree cung cấp thông tin hoàn chỉnh việc chuyển đổi hệ trực, tiếp theo robot cần publish dữ liệu đúng kiểu message của package này với độ chính xác nhất định. Trong meta package này hỗ trợ các package phục vụ toàn bộ quá trình điều hướng của robot gồm amcl, map_server, move_base và một số package tùy chọn khác.



Hình 3.33. Mô hình Navigation stack trong ROS

Hình 3.33 thể hiện cái nhìn tổng quan về cách cài đặt cũng như những thành phần có trong Navigation Stack. Những thành phần bên trong move_base (màu trắng) là những thành phần bắt buộc, được cấu hình tùy theo yêu cầu. Những thành phần như amcl, map_server (màu xám) là những thành phần tùy chọn, còn những thành phần còn lại (màu xanh) là những thành phần phải có của robot. Tác vụ điều hướng được thực hiện và quan sát thông qua nhiều bước kết hợp. Đối với đề tài mà nhóm thực hiện, tác vụ điều hướng robot lấy dữ liệu đầu vào gồm odometry, laser scan, bản đồ và xuất ngõ ra là vận tốc điều khiển robot.

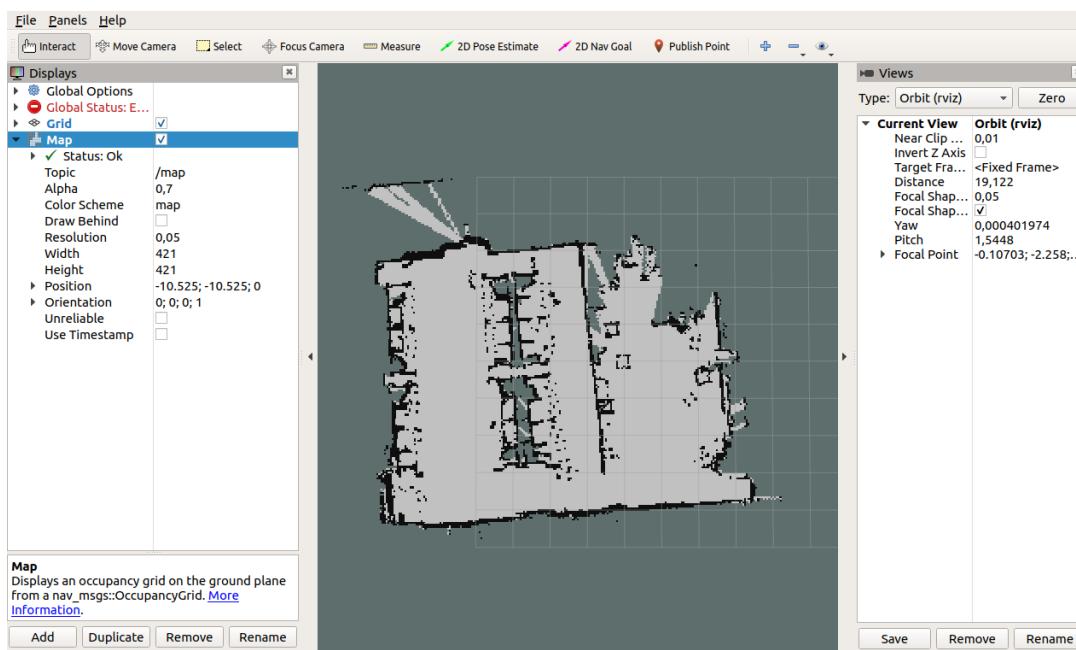
3.8.1. Khởi tạo kết nối cảm biến, rosserial và tải lên ROS thông số robot_description

Việc đầu tiên cần làm sử dụng file thesis_robot.launch để khởi tạo kết nối tới các cảm biến và rosserial, thu thập dữ liệu cần thiết phục vụ tác vụ điều hướng robot. Tiếp theo, tương tự như khi tác vụ định vị, điều kiện cũng là cần có mô hình quan sát và điều khiển trên phần mềm Rviz. Mô hình quan sát này đã được trình bày ở mục 3.1.5, đã được đóng gói thành một file launch và được gọi trong file launch của package sử dụng.

```
<include file="$(find thesis_robot_bringup/launch/thesis_robot_remote.launch")/>
```

3.8.2. Tải lên bản đồ đã vẽ

Tiến hành cập nhật bản đồ mà robot cần được điều hướng, được thực hiện thông qua package map_server. Mỗi bản đồ được lưu lại gồm 2 file là hình ảnh (.png) và cấu hình (.yaml) của bản đồ. Việc cập nhật này có thể được thực hiện thông qua lệnh (command) hoặc tích hợp trong file launch sử dụng.



Hình 3.34. Kết quả khi tải lên bản đồ đã vẽ

3.8.3. Định vị robot trong bản đồ

Tiến hành chạy tác vụ định vị robot như trình bày ở mục 3.6 để định vị robot trong bản đồ đã vẽ. Việc này là cần thiết để robot biết vị trí của bản thân trong môi trường khi đang điều

hướng. Tác vụ định vị được đóng gói thành một file launch, được gọi và sử dụng trong file launch cần thiết với các thông số cấu hình phù hợp cho nhiệm vụ mà robot thực hiện.

Bảng 3.18. Một số thông số cấu hình tác vụ định vị trong điều hướng robot

Thông số cấu hình	Giá trị
use_map_topic	true
base_frame_id	base_footprint
odom_frame_id	odom
scan	scan
min_particles	500
max_particles	3000
odom_model_type	diff

3.8.4. Tác vụ điều hướng với package move_base

Đây là thành phần cốt lõi trong Navigation Stack, được tích hợp sử dụng nhiều package hỗ trợ để phục vụ nhiệm vụ điều hướng. Các package quan tâm được sử dụng trong move_base gồm costmap_2d, global_planner và base_local_planner. Mỗi package nhỏ này phục vụ một nhiệm vụ nhất định được trình bày sau. Một số thông số cơ bản của package move_base được cấu hình như dưới đây.

Bảng 3.19. Thông số cấu hình cơ bản của move_base

Thông số	Giá trị	Ý nghĩa
map_topic	map	Topic lấy dữ liệu về bản đồ
odom_topic	odom	Topic lấy dữ liệu về odometry của robot
cmd_vel_topic	cmd_vel	Topic xuất giá trị vận tốc điều khiển robot
move_forward_only	true	Đặt điều kiện robot chỉ di chuyển về phía trước

3.8.4.1. Cấu hình costmap

Costmap_2D là một package để tạo bản đồ chi phí (costmap) từ việc lấy dữ liệu của cảm biến về môi trường và bản đồ đã biết, xây dựng nên bản đồ lưới về các nơi bị chiếm chỗ và tăng vùng chi phí trong dựa trên nơi bị chiếm chỗ và bản kính tăng chi phí do nhóm định nghĩa. Bản đồ này chứa thông tin về vật cản cũng như những nơi mà robot không thể di chuyển tới, với các lớp dữ liệu khác như bản đồ tĩnh là một lớp, vật cản động là một lớp giúp robot xử lý và di chuyển thông minh. Robot được định hướng đường đi toàn cục cho đường đi dài và cục bộ trong không gian nhỏ tương ứng với costmap toàn cục và cục bộ. Mỗi costmap có những thông số cấu hình chung và riêng, được đóng gói thành các file yaml.

Bảng 3.20. Một số thông số cấu hình chung của costmap

Thông số	Giá trị	Ý nghĩa
obstacle_range	3.0	khoảng cách tối đa cảm biến đọc được và cập nhật về môi trường trên costmap
robot_radius	0.15	Bán kính robot tính từ tâm
inflation_radius	0.3	Khoảng cách nhỏ nhất giữa tâm robot và vật cản
observation_sources	scan	Dữ liệu từ cảm biến cập nhật costmap
scan	{ sensor_frame: lidar_link, data_type: LaserScan, topic: scan, marking: true, clearing: true }	Thông tin về hệ tọa độ và topic của cảm biến để đồng bộ với hệ tọa độ chung trên bản đồ.

Global Costmap

Đây là costmap toàn cục giúp robot điều hướng đường đi dài, tìm đường đi tối ưu toàn bản đồ.

Bảng 3.21. Thông số cấu hình Costmap toàn cục

Thông số	Giá trị	Ý nghĩa
global_frame	map	Hệ tọa độ toàn cục

robot_base_frame	base_footprint	Hệ tọa độ gắn với robot
update_frequency	5.0	Tần số cập nhật costmap (Hz)
publish_frequency	4.0	Tần số cập nhật thông tin (Hz)
static_map	true	Dùng map đã biết hay không

Local Costmap

Đây là costmap cục bộ được cập nhật liên tục trong một khoảng không gian nhất định, giúp robot xác định và tránh vật cản động.

Bảng 3.22. Một số thông số cấu hình Local Costmap

Thông số	Giá trị	Ý nghĩa
global_frame	map	Hệ tọa độ toàn cục
robot_base_frame	base_footprint	Hệ tọa độ gắn với robot
update_frequency	5.0	Tần số cập nhật costmap (Hz)
publish_frequency	4.0	Tần số cập nhật thông tin (Hz)
static_map	true	Dùng map đã biết hay không
rolling_window	true	Giữ costmap ở trung tâm khi robot di chuyển hay không
width	2	Chiều dài costmap cục bộ (m)
height	2	Chiều rộng costmap cục bộ (m)
resolution	0.05	Độ phân giải (m)

3.8.4.2. Cấu hình planner

Sau khi đã có costmap, việc lên kế hoạch cho đường đi của robot được thực hiện thông qua hai hoạch định đường đi là toàn cục (global planner) và cục bộ (local planner). Mỗi planner có chức năng riêng của mình, dựa trên costmap tương ứng với mục tiêu tránh vật cản và di chuyển robot đến vị trí đích.

Global Planner

Việc hoạch định đường đi toàn cục cho robot sử dụng thuật toán Dijkstra được trình bày ở mục 2.5.1, với mục tiêu tìm đường đi ngắn nhất từ vị trí robot đến vị trí đích dựa trên bản đồ đã biết. Cấu hình global planner phù hợp với đặc tính robot và môi trường hoạt động là việc cần thiết. Việc cấu hình sử dụng thuật toán được thiết lập thông qua thông số `use_dijkstra` khi cài đặt giá trị true.

Local Planner

Dựa trên local costmap luôn được cập nhật, local planner sử dụng thuật toán DWA được trình bày ở mục 2.5.2 để hoạch định đường đi tránh vật cản động khi robot di chuyển. Các thông số cấu hình được lựa chọn sao cho phần cứng robot đáp ứng được mục tiêu đặt ra.

Bảng 3.23. Một số thông số cấu hình Local Planner

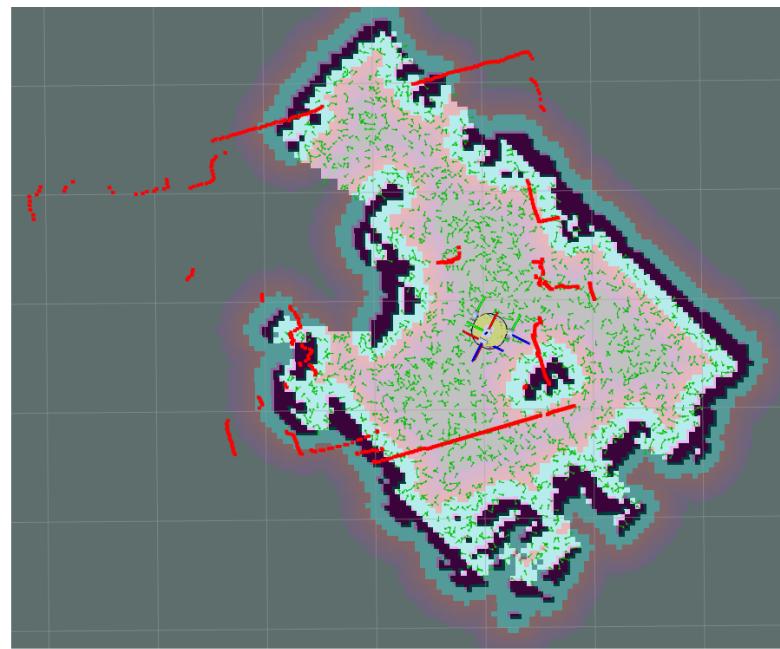
Thông số	Giá trị	Ý nghĩa
<code>max_vel_x</code>	0.15 m/s	Vận tốc tối đa của robot theo trục x
<code>min_vel_x</code>	-0.15 m/s	Vận tốc tối thiểu của robot theo trục x
<code>max_vel_y</code>	0 m/s	Vận tốc tối đa của robot theo trục y
<code>min_vel_y</code>	0 m/s	Vận tốc tối thiểu của robot theo trục y
<code>max_vel_trans</code>	0.15 m/s	Vận tốc tịnh tiến tối đa
<code>min_vel_trans</code>	0.10 m/s	Vận tốc tịnh tiến tối thiểu
<code>max_vel_theta</code>	1.5 rad/s	Vận tốc xoay tối đa
<code>min_vel_theta</code>	0.5 rad/s	Vận tốc xoay tối thiểu
<code>acc_lim_x</code>	2.0m/s ²	Gia tốc thẳng theo trục x
<code>acc_lim_y</code>	0 m/s ²	Gia tốc thẳng theo trục y
<code>acc_lim_theta</code>	3.0 rad/s ²	Gia tốc góc
<code>xy_goal_tolerance</code>	0.05 m	Sai số cho phép về tọa độ so với vị trí đích
<code>yaw_goal_tolerance</code>	0.17 rad	Sai số cho phép về góc so với vị trí đích

Ở đây, các thông số của trục y được thiết lập về giá trị 0 do robot hoạt động theo mô hình differential drive, không có khả năng di chuyển được theo trục y.

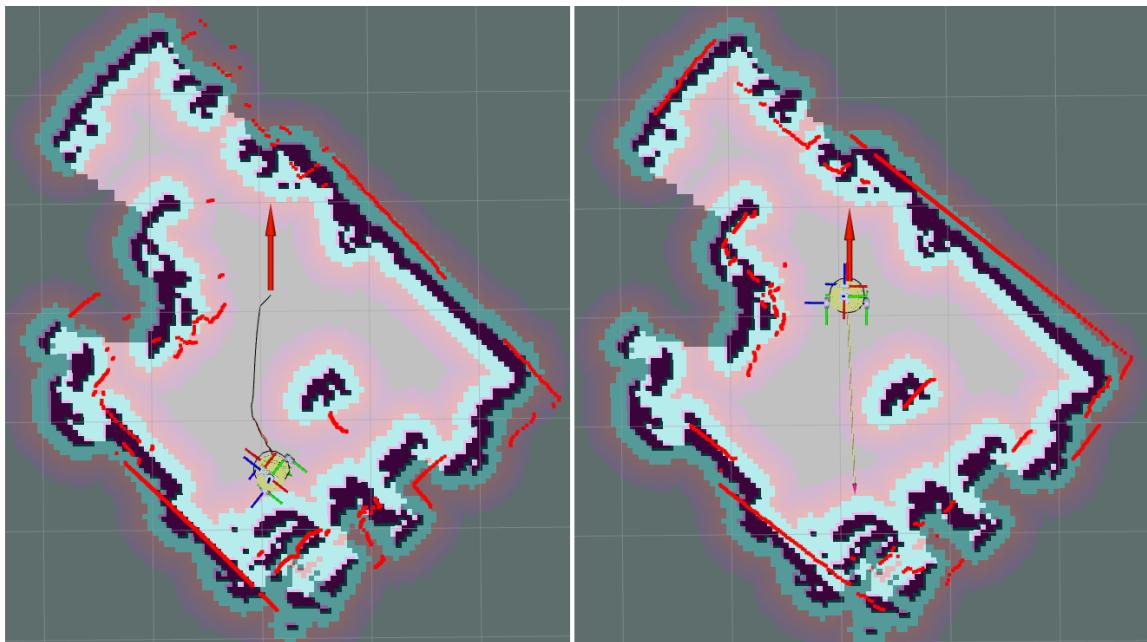
3.8.5. Tổng hợp quá trình điều hướng robot

Sau khi hoàn thành việc thiết lập tác vụ điều hướng cho robot, tiến hành kiểm tra sự đúng đắn của tác vụ thông qua việc quan sát tất cả các node hoạt động trong ROS. Các node đã subscribe vào các topic cần thiết và hoạt động đúng yêu cầu đề ra.

Trong quá trình điều hướng cho robot, xảy ra 3 trường hợp mà robot hoạch định đường đi khác nhau gồm không vật cản, có vật cản tĩnh và có vật cản động. Việc thực hiện hoạch định này được thể hiện thông qua sự quan sát trên Rviz, với các đường đi hoạch định quan sát được.



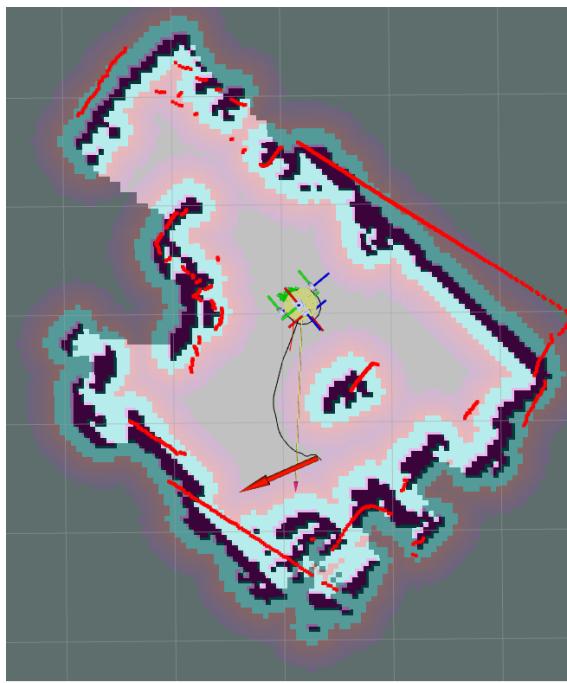
a) Bắt đầu quá trình điều hướng, chưa định vị



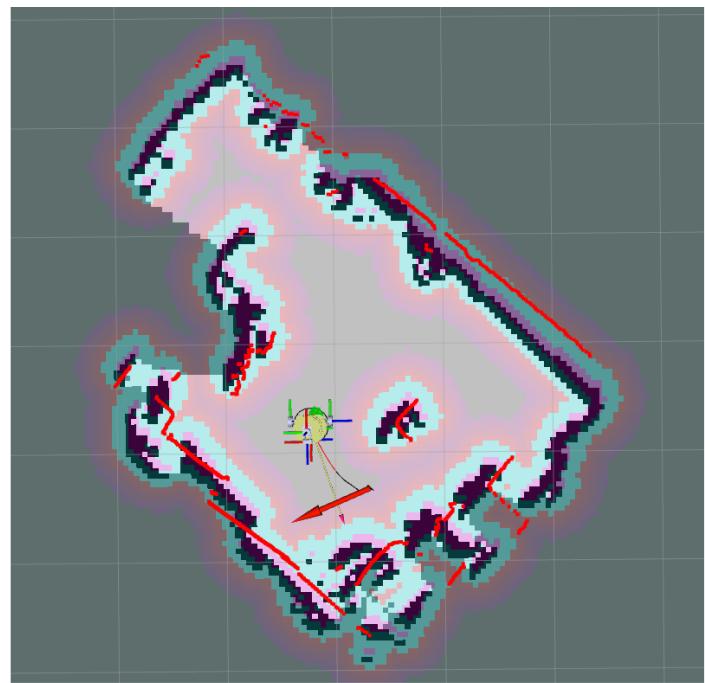
b) Đã định vị, chọn điểm mục tiêu và
hoạch định đường đi

c) Đến mục tiêu và dừng lại

Hình 3.35. Kết quả điều hướng robot khi không có vật cản và quan sát trên Rviz



a) Hoạch định đường đi khi có vật cản tĩnh trong bản đồ



b) Di chuyển theo hoạch định



c) Đến điểm mục tiêu và dừng lại

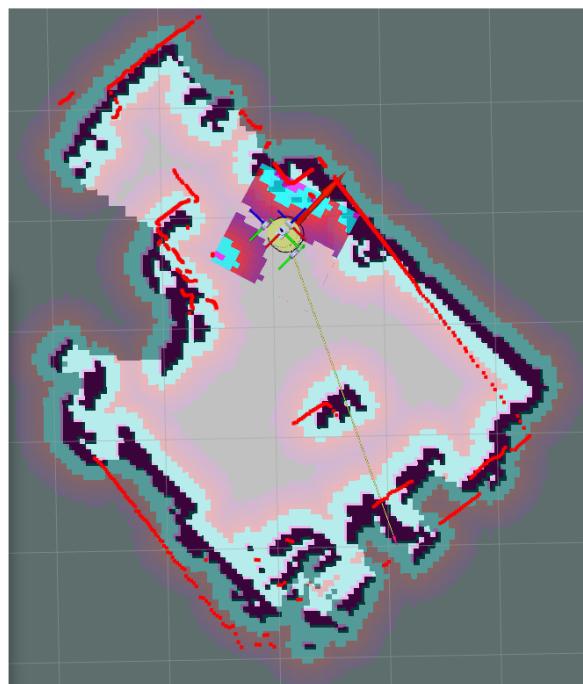
Hình 3.36. Kết quả điều hướng robot khi có vật cản tĩnh và quan sát trên Rviz



a) Chọn điểm mục tiêu và hoạch định đường đi toàn cục



b) Xuất hiện vật cản động trong dynamic window và hoạch định đường đi cục để vượt qua



c) Vượt qua vật cản động và đến điểm mục tiêu

Hình 3.37. Kết quả điều hướng robot khi có vật cản động và quan sát trên Rviz

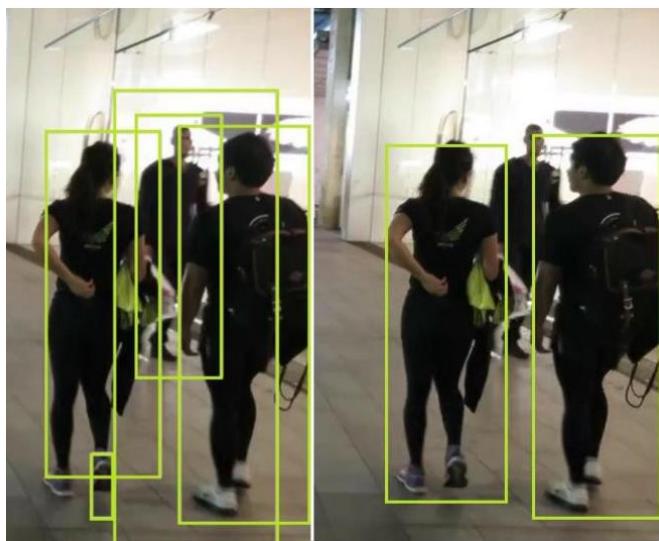
3.9. Thuật toán bám theo người (Human Following)

Bài toán Human Following thường được xem như một bài toán Object Tracking vì để có thể bám theo người, cần theo dõi được quỹ đạo di chuyển của người. Phương pháp Object Tracking sử dụng trong đề tài là phương pháp Tracking-by-Detection, tức là người trong ảnh sẽ được theo dõi dựa vào các thuật toán phát hiện vật thể trong ảnh (Object Detection), cụ thể là phát hiện người.

Camera sử dụng cho các tác vụ xử lý ảnh trong đề tài cũng là camera dùng cho việc vẽ bản đồ được đặt trên robot, và nằm ở vị trí có chiều cao không quá đầu gối người trưởng thành. Vì thế, khi người đứng ở khoảng cách hợp lý nhất cho việc nhận diện, camera chỉ lấy được hình ảnh đến 2/3 thân người về phía dưới (từ thắt lưng xuống đến hai chân).

3.9.1. Các khó khăn thường gặp trong bài toán

Occlusion: thuật toán phải nhận dạng được người cần bám theo khi đó xuất hiện trở lại sau vài khung ảnh bị khuất, và liên kết được người đó với các thông tin tracking trước đó, cũng như các đặc trưng quỹ đạo di chuyển cũ.



Hình 3.38 Vấn đề occlusion trong Human Tracking

Non stationary camera: quỹ đạo được dự đoán của vật thể có thể bị ảnh hưởng khi camera di chuyển. Một thuật toán giải quyết tốt được vấn đề này sẽ giúp ích cho các ứng dụng robot tự hành.

3.9.2. Mục đích sử dụng các phương pháp

Đề tài không sử dụng các bộ tracker như KCF (Kernelized Correlations Filtering) hay CSRT (The channel and Spatial Reliability Tracker), DeepSORT ... vì các lý do sau:

- Thuật toán KCF [16] không sử dụng deep learning, ảnh đầu vào là ảnh trắng đen, đặc trưng trích xuất là đặc trưng HOG. HOG là đặc trưng chuyên dùng cho việc phát hiện biên, và nhận dạng hình dáng. Nhưng trong điều kiện đã nêu, phần thân dưới của người hầu như đều có hình dạng giống nhau.
- Thuật toán như CSRT [17] có sử dụng mô hình Object Detection đã được huấn luyện sẵn là Faster RCNN. Tuy nhiên qua kiểm thử trong điều kiện thực tế của đề tài, bộ tracker này cho kết quả bounding box không sát với người cần theo dõi (cho ra tọa độ của người không chính xác), dẫn đến khó khăn trong việc bám theo người khi robot di chuyển. Vì vậy, CSRT được sử dụng để hỗ trợ quá trình bám người.
- DeepSORT là thuật toán khá nặng về khói lượng tính toán, không phù hợp triển khai trên môi trường robot của đề tài.

Human Detection với mô hình MobileNet SSD

Có rất nhiều thuật toán thực hiện tác vụ nhận diện hình dáng người như phương pháp Haar Cascade Classifier. Nhưng nhóm chọn mô hình CNN MobileNet SSD cho thuật toán Object Detection xương sống trong đề tài này vì những lý do sau:

- Qua kiểm thử, các đặc trưng trích xuất bởi phương pháp Haar Cascade cho việc nhận dạng cơ thể người không dùng mô hình CNN, cho kết quả không tốt và kém ổn định.
- Một mô hình deep neural network phát hiện người được huấn luyện trên tập dữ liệu lớn sẽ giúp phát hiện được chính xác người xuất hiện trong ảnh trong điều kiện một phần thân trên người đã bị khuất mất.
- MobileNet SSD là mô hình phù hợp cho các ứng dụng trên hệ thống nhúng, cũng là hệ thống vận hành robot (mục 2.6).
- Class *Person* là một nhãn class nằm trong tập dữ liệu mà mô hình này được huấn luyện, vì thế ta có thể sử dụng mô hình đã được huấn luyện sẵn (pretrained model) cho việc phát hiện người trong ảnh.

Mô hình MobileNet SSD được sử dụng trong đề tài là phiên bản dùng framework Caffe được huấn luyện bởi *chuanqi305*³ trên tập dataset VOC0712. Mô hình huấn luyện sẵn này được lưu dưới dạng một file cấu trúc mô hình “*MobileNetSSD_deploy.caffemodel*” và một file trọng số “*MobileNetSSD_deploy.prototxt.txt*”, có thể được dùng để phát hiện 20 loại vật thể và background (vùng nền). Class *Person* có vị trí là 15 trong vector **labels**.

Module DNN (Deep Neural Network) của OpenCV hỗ trợ nhiều framework deep learning như Caffe, Tensorflow, Pytorch, ... Mô hình MobileNet SSD có thể dễ dàng được load vào ứng dụng của người dùng bằng module DNN.

```
Net = cv2.dnn.readNetFromCaffe("../MobileNetSSD_deploy.prototxt.txt",  
"../MobileNetSSD_deploy.caffemodel")
```

CUDA là một ứng dụng, platform, mô hình được tạo bởi NVIDIA, cho phép truy cập trực tiếp vào các tập lệnh ảo (virtual instruction set) và phần tử tính toán song song (parallel computational elements) của GPU, giúp dễ dàng sử dụng resource của GPU cho các phép tính mà không cần đến các kỹ năng lập trình đồ họa nâng cao. Module DNN hỗ trợ lựa chọn cấu hình cho CUDA để sử dụng khả năng tính toán của GPU trên máy tính cho việc thực hiện các phép tính của mô hình. Máy tính nhúng Jetson Nano có tích hợp GPU của NVIDIA, vì thế chỉ cần cài đặt các thư viện và driver cần thiết, cấu hình cho module “*dnn*” của OpenCV, là đã có thể tăng tốc độ xử lý của mô hình nhờ GPU.

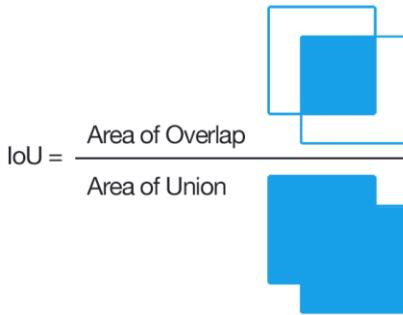
```
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)  
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)
```

Tính toán hệ số tỉ lệ IOU

Những bounding box chứa người được phát hiện trong ảnh cần phải được tính toán IOU để xem có nằm trùng một phần với tracking box (bounding box chứa người đang được bám theo (target person) trong khung ảnh cũ) hay không. Nếu hệ số tỉ lệ IOU lớn hơn một mức ngưỡng, bounding box đó được xem là chứa target person, và track box được cập nhật mới.

³ Nguồn: <https://github.com/chuanqi305/MobileNet-SSD>

IOU là tỉ lệ trùng nhau của hai bounding box, được tính bằng tỉ lệ giữa phần giao nhau với phần hợp nhau của hai bounding box này.



Hình 3.39 Cách tính toán IOU giữa hai bounding box

Tính toán đặc trưng bằng Kmeans-Clustering

Trong quá trình bám theo người, sẽ có nhiều trường hợp mô hình Object Detection không đủ khả năng quyết định target person trong ảnh và cập nhật track box mới như:

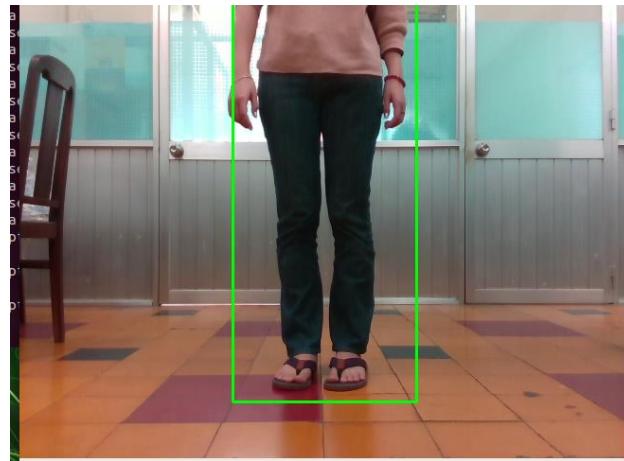
- Có nhiều hơn một bounding box có IOU với track box vượt ngưỡng đã định.
- Khi target person di khuất khỏi ảnh, robot cần di chuyển để bám theo. Khi có người xuất hiện trở lại trong ảnh, lúc này không thể nào biết được liệu người đó có phải target person không khi chỉ sử dụng phương pháp IOU.

Vì vậy, cần phải có sự trích lấy đặc trưng của target person ban đầu, cũng như của các bounding box đang được phân tích trong các trường hợp trên để thực hiện việc so sánh.

Trong bounding box chỉ có phần thân dưới của người, hầu hết có hình dáng giống nhau, việc sử dụng một mô hình CNN để huấn luyện nhận dạng được target person là rất tốn kém thời gian xử lý và khối lượng tính toán. Vì thế, nhóm chọn đặc trưng màu sắc làm đặc điểm nhận dạng giữa các bounding box. Lúc này, phương pháp Kmeans-Clustering được sử dụng trên pixel map ba kênh màu của ảnh trong bounding box với $K = 3$ để phân màu trong ảnh thành ba cụm và sử dụng nó làm đặc trưng. Phương pháp Euclidean Distance được sử dụng để tính độ tương quan giữa hai đặc trưng. Người đầu tiên được phát hiện trong khung ảnh khi chạy thuật toán sẽ được trích lấy đặc trưng làm đặc trưng tham chiếu $feature_{reference}$.

$$distance = ||feature_{new} - feature_{reference}||_2^2$$

Bounding box trước khi được trích đặc trưng cần được crop đi $\frac{1}{4}$ diện tích từ rìa vào để loại bỏ bớt background, nhờ vậy đặc trưng sẽ chỉ bao gồm màu sắc quần áo của người trong bounding box.

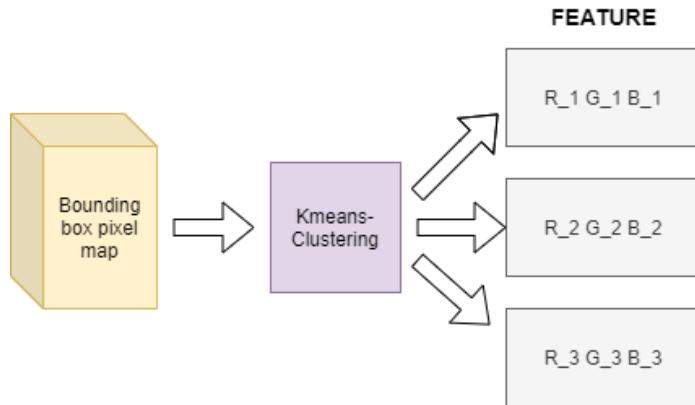


Hình 3.40 Bounding box chứa người cần trích xuất đặc trưng.



Hình 3.41 Bounding box sau khi đã được crop để dùng trích xuất đặc trưng

Ma trận đặc trưng có kích thước 3×3 , với mỗi hàng gồm ba phần tử (R, G, B) đại diện một màu sắc được chọn làm center của cụm. Ba hàng tương đương với ba màu sắc center.



Hình 3.42 Sơ đồ khái trich xuất đặc trưng của ảnh trong bounding box

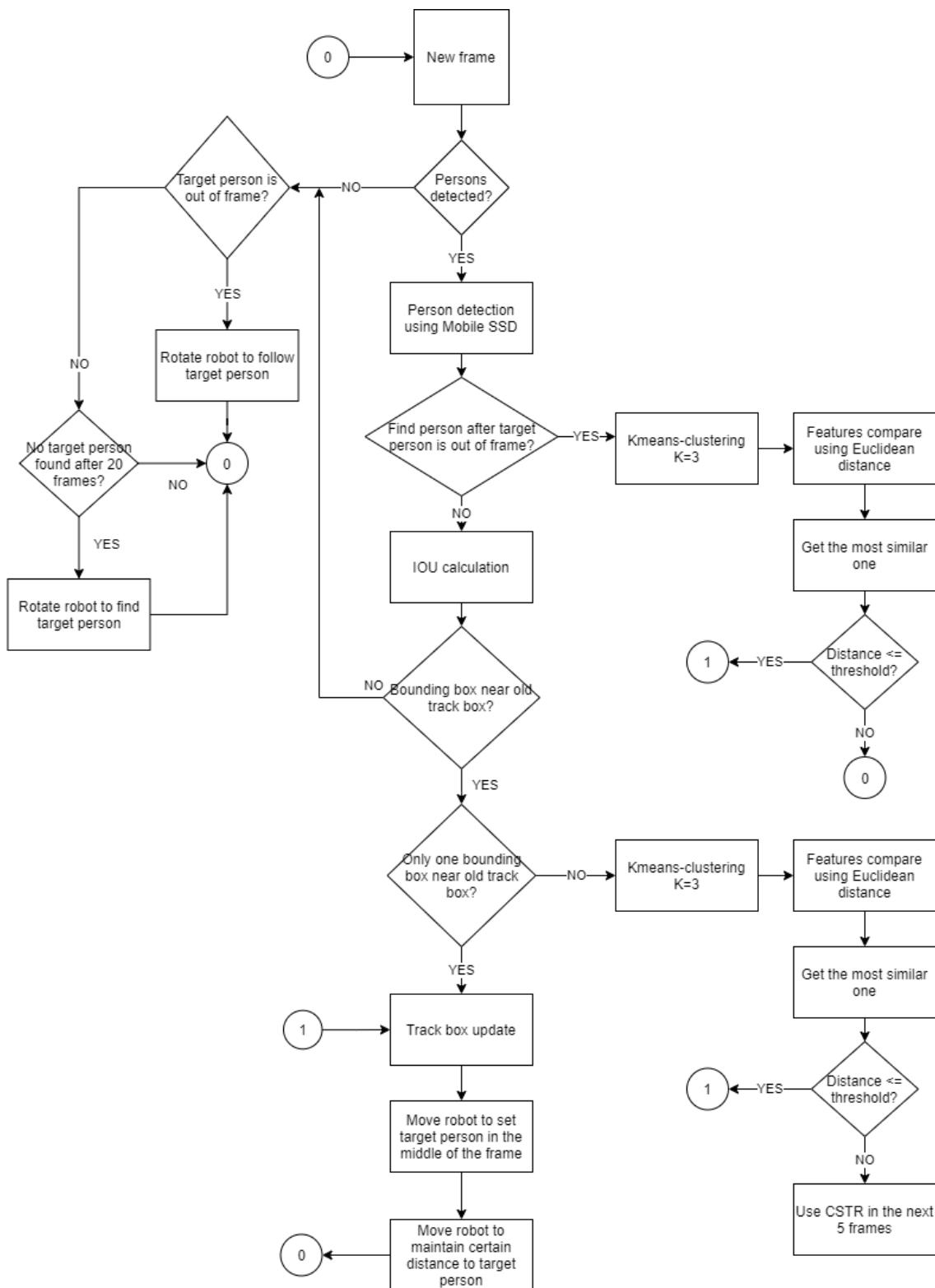
$$\mathbf{feature} = \begin{bmatrix} Color1_R & Color1_G & Color1_B \\ 0.75Color2_R & 0.75Color2_G & 0.75Color2_B \\ 0.75Color3_R & 0.75Color3_G & 0.75Color3_B \end{bmatrix}$$

$$distance = \min(distance_1, distance_2, \dots, distance_6)$$

Hai cụm màu có ít điểm dữ liệu con nhất trong ma trận **feature** sẽ được gán trọng số 0.75 để giảm độ ảnh hưởng lên kết quả.

Tuy nhiên, ở những góc độ khác nhau, các cụm màu sẽ có số điểm dữ liệu khác nhau. Cụm có nhiều điểm dữ liệu nhất lúc ban đầu chưa hẳn vẫn sẽ vẫn chiếm số điểm dữ liệu lớn nhất ở các khung ảnh sau đó. Vì thế, khi sử dụng cho phép so sánh, các hàng của ma trận được luân phiên đổi thứ tự để tìm được 6 giá trị *distance* tương ứng. Giá trị nhỏ nhất sẽ được sử dụng làm thông số so sánh ngưỡng để đảm bảo các màu gần giống nhau nhất sẽ được so sánh với nhau.

3.9.3. Xây dựng giải thuật



Hình 3.43 Sơ đồ giải thuật tóm tắt của thuật toán bám theo người

Sơ đồ giải thuật trên Hình 3.43 được trình bày rõ như sau, với các thông số ngưỡng được chọn từ thực nghiệm, trong đó, mỗi khung ảnh được lấy dưới dạng các ma trận điểm ảnh do Realsense public lên topic “/camera/color/image_raw” trên ROS:

- Bước 1: Với mỗi khung ảnh mới, áp dụng mô hình MobileNet SDD để tìm người. Nếu $score \geq 0.5$ thì bounding box được chấp nhận. Nếu không có bounding box nào được tìm thấy, thực hiện bước 2. Nếu có bounding box được tìm thấy, thực hiện bước 3.
- Bước 2: Xem xét vị trí của track box cũ xem có nằm sát với biên trái hoặc phải của khung ảnh không? Nếu có, xem rằng target person đã đi khuất khỏi khung ảnh, cho robot xoay theo hướng đó để tìm người, và quay lại bước 1. Nếu không, quay lại bước 1. Nếu không có target person xuất hiện trong 20 khung ảnh liên tục thì cho xoay robot để tìm người trước khi quay lại bước 1s.
- Bước 3: Nếu đây là những bounding box được tìm thấy sau khi target person bị xem là đi khuất khỏi ảnh, trích xuất ma trận $feature_{new}$ và so sánh với $feature_{reference}$ để lấy hệ số bất tương đồng $distance$. Nếu không, thực hiện bước 5.
- Bước 4: Nếu chỉ có một bounding box được tìm thấy, bounding box sẽ được xem là chứa target person nếu $distance \leq 70$. Nếu có nhiều hơn một bounding box được tìm thấy, bounding box có $distance$ nhỏ nhất và ≤ 70 sẽ được xem là chứa target person. Nếu không có bounding box nào đạt chuẩn, tiếp tục xoay robot để tìm target person và quay lại bước 1. Nếu không, cho robot dừng xoay và thực hiện bước 7.
- Bước 5: Tính toán tỉ số IOU của mỗi bounding box được tìm thấy, nếu $IOU \leq 0.3$ sẽ được xem là có khả năng cao trùng với track box cũ, và sẽ được lưu lại. Nếu không có bounding box nào gần với track box cũ, quay lại bước 2. Nếu có nhiều hơn một bounding box gần với track box cũ, thực hiện bước 6. Nếu có duy nhất một bounding box gần với track box cũ, thực hiện bước 7.
- Bước 6: Tìm $distance$ giữa $feature_{new}$ và $feature_{reference}$ của mỗi bounding box được lưu ở bước 5. Chọn bounding box có $distance$ bé nhất và ≤ 70 là bounding box có chứa target person. Nếu không đạt chuẩn, bộ tracker CSRT sẽ được sử dụng trong 5 khung ảnh kế tiếp để hỗ trợ thuật toán. Sau đó quay lại bước 1.

- Bước 7: Cập nhật track box mới với bounding box duy nhất được lưu.
- Bước 8: Di chuyển robot sao cho bounding box chứa target person nằm ở giữa khung ảnh, và khoảng cách từ target person đến robot nằm trong khoảng từ 1.4 m đến 1.6 m. Khoảng cách từ target person đến robot được trích xuất từ camera Realsense D435, mà trận ảnh depth được Realsense public lên topic “/camera/aligned_depth_to_color/image_raw”. Quay lại bước 1.

Bảng 3.24 Bảng thông số threshold (ngưỡng) cho thuật toán bám theo người.

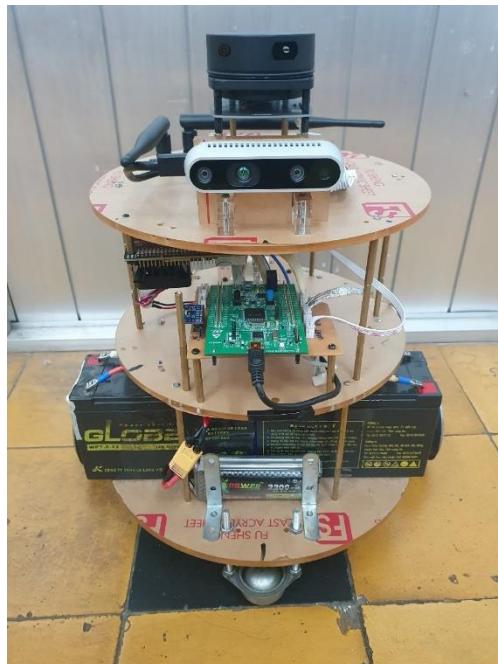
Detection score threshold	0.5
IOU threshold	0.3
Features distance threshold	70
Distance threshold	Min: 1.4 m; Max: 1.6m

CHƯƠNG 4. KẾT QUẢ THỰC HIỆN

4.1. Kết quả xây dựng mô hình robot

4.1.1. Mô hình robot thực tế

Sau khi thực hiện vẽ mô hình robot trên SolidWorks, nhóm tiến hành cắt laser và lắp ráp. Kết quả robot thu được như hình dưới đây, gồm 3 tầng như đã thiết kế ở mục 3.1.



Hình 4.1. Mô hình robot thực tế

4.1.2. Thông số mô hình và nhận xét

Bảng 4.1. Thông số của mô hình robot

Loại thông số	Thiết kế (đơn vị)	Thực tế (đơn vị)
Chiều cao	60 cm	66 cm
Đường kính	35 cm	42 cm
Cân nặng	0.6 kg	4 kg
Chiều cao mỗi tầng	20 cm	20 cm

Mô hình có các thông số thực tế được trình bày ở Bảng 4.1. Mô hình robot thực tế được chỉnh sửa trong quá trình thực hiện, nâng cấp cải tiến và thay đổi một số thông số đã thiết kế, và đã đáp ứng được các yêu cầu đề ra gồm khả năng di chuyển, khả năng chịu tải và khả năng quan sát môi trường.

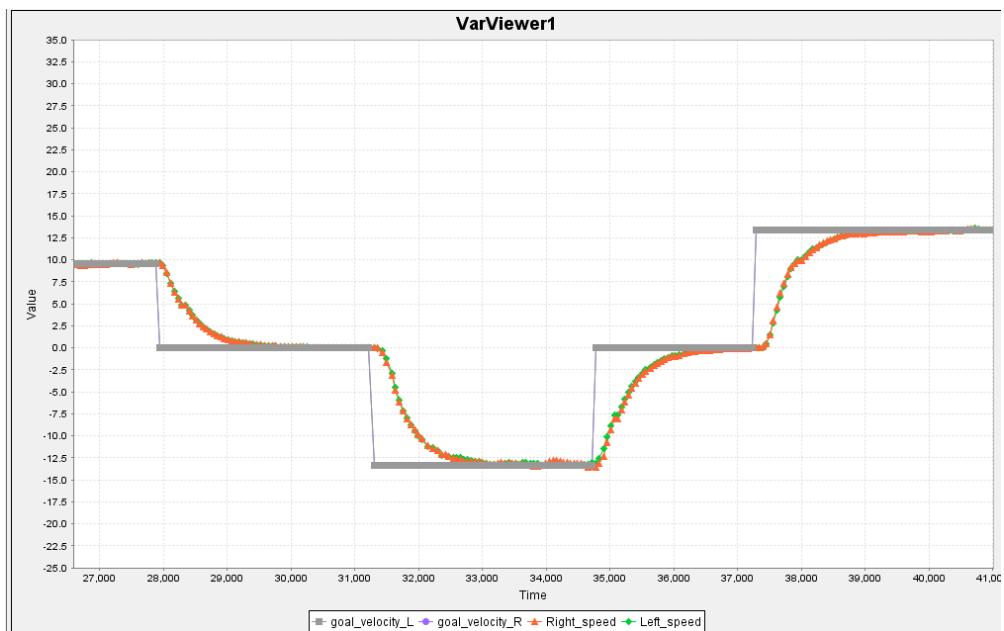
4.2. Kết quả điều khiển tốc độ động cơ có tải

Với các thông số điều khiển và giá trị hệ số bộ điều khiển được chọn như ở mục 3.2.2, khi vẽ đồ thị đáp ứng cho hai bánh xe, ta nhận thấy tuy thời gian xác lập khá lớn (khoảng 1.5 s), tín hiệu điều khiển vẫn khá ổn định và không có vọt lồ cho cả hai bánh xe. Thời gian xác lập lớn một phần là vì bộ lọc thông thấp được thêm vào trước tín hiệu đặt của bộ PID.

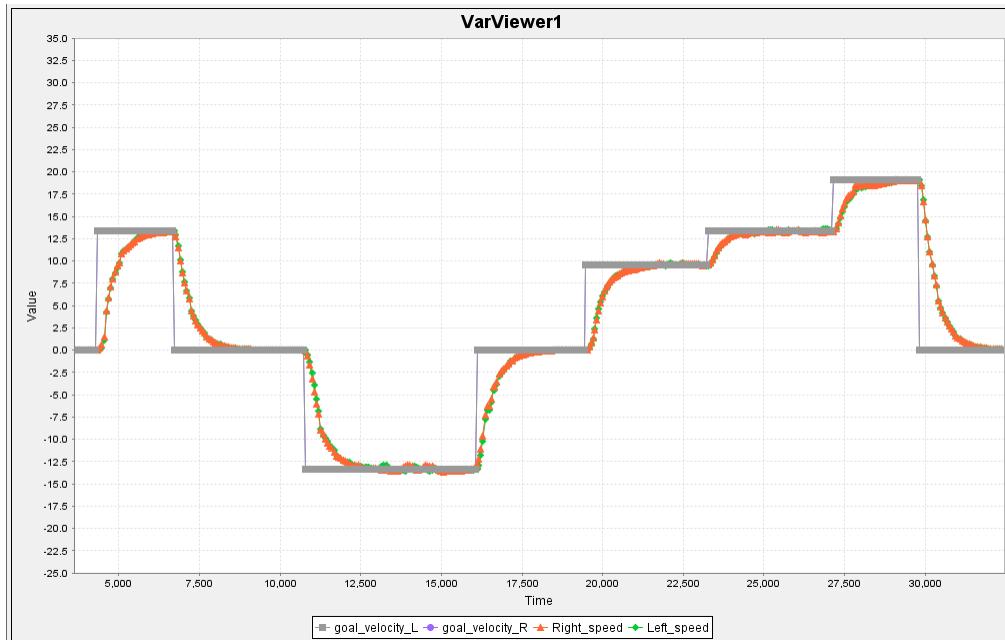
Đáp ứng của động cơ phải (điểm màu đỏ) dao động nhiều hơn động cơ trái theo nhóm nhận xét trong quá trình thử nghiệm đọc encoder, là vì xung encoder của động cơ phải có nhiều gai nhiễu hơn, nhóm xem đây là giới hạn phần cứng.

Bảng 4.2 Bảng kết quả điều khiển tốc độ động cơ có tải bằng PID

Thời gian xác lập	T_{xl}	1.5 s
Độ vọt lồ	POT	0
Sai số xác lập	e_{xl}	0
Thời gian lên	T_{len}	1 s

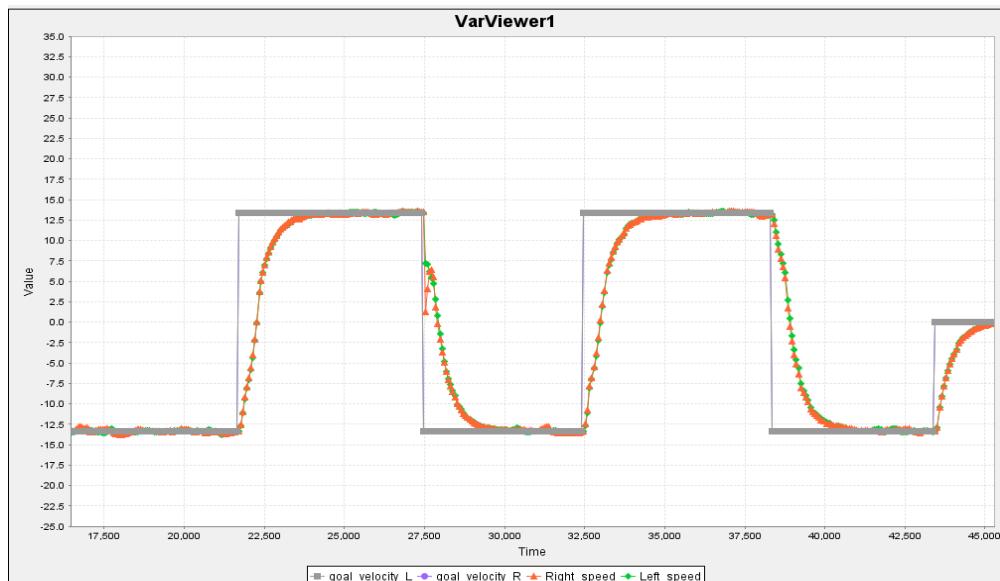


Hình 4.2 Đồ thị đáp ứng điều khiển tốc độ hai động cơ có tải bằng PID (đơn vị trực x: s – y: m/s)



Hình 4.3. Đồ thị đáp ứng điều khiển tốc độ hai động cơ có tải bằng PID với $r(t)$ ở nhiều mức khác nhau (đơn vị trực x: s – y: m/s)

Khi cho động cơ đảo chiều liên tục, thời gian xác lập không bị giảm và đáp ứng vẫn tốt, tuy rằng có thể xảy ra dao động trong khoảng thời gian rất ngắn như từ thời điểm 27.5 s đến thời điểm 27.8 s như Hình 4.4.



Hình 4.4 Đồ thị đáp ứng điều khiển tốc độ hai động cơ có tải bằng PID với $r(t)$ luân phiên đảo dấu (đơn vị trực x: s – y: m/s)

4.3. Kết quả tính toán odometry

4.3.1. Cách thức thực hiện

Tại một môi trường không cần xác định trước, tiến hành khởi động cảm biến của robot và giao thức rosserial, được trình bày từ mục 3.1 đến 3.5. Môi trường được lựa chọn thực hiện là phòng trọ có diện tích $11.5m^2$. Sau đó, thông qua chuẩn kết nối từ xa UDP giữa ROS trên máy tính nhúng và ROS trên laptop, thực hiện điều khiển robot đi đến vị trí bất kỳ. Sau khi điều khiển robot đến vị trí mong muốn, sử dụng các dụng cụ có sẵn tiến hành đo đặc khoảng cách giữa vị trí đầu và vị trí hiện tại của robot theo 2 trục x và y (Trục x hướng về phía trước, trục y hướng về phía bên trái của robot) cũng như đo góc lệch theta giữa 2 vị trí của robot. Việc tính toán odometry của robot có thể đạt được thông qua hai phương pháp gồm:

- Tính toán từ cảm biến IMU và Encoder thông qua mạch điều khiển STM32: Việc tính toán này được trình bày ở mục 3.2.1, sử dụng kết hợp dữ liệu IMU và Encoder, thông qua bộ lọc Madgwick và tính ra odometry. Sau đó, thông qua chuẩn giao tiếp rosserial, mạch điều khiển STM32 gửi odometry lên ROS trên máy tính nhúng với tần số 10Hz.
- Tính toán thông qua thuật toán scan_matching được trình bày ở mục 2.4 sau khi xây dựng phần mềm trên nền tảng ROS: Thuật toán này sử dụng dữ liệu laser từ Lidar để đối chiếu với môi trường, ước lượng ra vị trí robot và tính odometry xuất ra với tần số 10Hz. Tất cả các bước được đóng gói thành package laser_scan_matcher.

Thực hiện thu thập số liệu như trên 5 lần để đánh giá kết quả.

4.3.2. Odometry từ mạch điều khiển

Bảng 4.3 Kết quả đo odometry thông qua IMU và Encoder

Lần thử nghiệm	Tọa độ x (m)		Tọa độ y (m)		Góc yaw (rad)	
	Thực tế	Tính toán	Thực tế	Tính toán	Thực tế	Tính toán
Lần 1	1.07	1.0985	0.125	0.1001	5.55	5.5271
Lần 2	0.78	0.7474	0.82	0.8437	0.6109	0.6777
Lần 3	0.13	0.1225	1.2	1.1764	1.1694	1.1615
Lần 4	1.2	1.2306	1.28	1.2541	1.57	1.5664
Lần 5	0.48	0.4664	1.15	1.1967	3.6652	3.7237
Sai số RMS	0.0247		0.0727		0.0412	

4.3.3. Odometry thông qua dữ liệu từ Lidar

Bảng 4.4. Kết quả đo odometry thông qua dữ liệu từ Lidar

Lần thử nghiệm	Tọa độ x (m)		Tọa độ y (m)		Góc yaw (rad)	
	Thực tế	Tính toán	Thực tế	Tính toán	Thực tế	Tính toán
Lần 1	1.1	1.0982	0.3	0.2743	0.6632	0.622
Lần 2	0.75	0.7258	1.2	1.2090	1.57	1.5202
Lần 3	1.56	1.5144	1.2	1.2474	0.8727	0.8532
Lần 4	1.15	1.0602	2.18	2.1551	3.0718	3.1572
Lần 5	0.35	0.3030	1.3	1.3612	1.92	1.9481
Sai số RMS	0.0509		0.0384		0.0503	

4.3.4. Nhận xét và đánh giá

Sau quá trình thu thập số liệu, sai số của odometry được tính toán ra với giá trị nhỏ. Cả hai phương pháp đều cho sai số tọa độ vào khoảng 5cm và sai số góc vào khoảng 3°. Như vậy, dù bị hạn chế về phần cứng (IMU nhiều, encoder có xung thấp, lidar nhiều) thì việc tính toán odometry được đánh giá là hoàn thành tốt mục tiêu đặt ra với sai số nhỏ chấp nhận được. Kết quả odometry thu được như trên đáp ứng được yêu cầu thực hiện tác vụ vẽ bản đồ và các tác vụ về sau của robot.

4.4. Kết quả vẽ bản đồ với RTAB-Map

4.4.1. Môi trường và mục đích thử nghiệm

Nhóm thực hiện thử nghiệm trong môi trường phòng lab và phòng thí nghiệm 207B3 có nhiều vật cản như bàn, ghế, xe mô hình, ... và vật cản di động như người di chuyển. Sau đó, nhóm tiến hành lại trong môi trường là căn phòng với kích thước 11.5m². Robot được đánh giá khả năng vẽ bản đồ 2D và 3D, khả năng tránh vật cản bao gồm cả tĩnh và động, khả năng tìm đường đi đến đích và khả năng bám theo người. Môi trường đánh giá được sử dụng là không gian phòng thí nghiệm 207B3 trường ĐH Bách Khoa TP.HCM và không gian một căn phòng trọ, tương đương với hình ảnh thực tế ở Hình 4.5 và Hình 4.6.



Hình 4.5. Môi trường vẽ bản đồ và thực hiện các tác vụ là phòng thí nghiệm 207B3 ĐH Bách Khoa TP.HCM



Hình 4.6. Môi trường vẽ bản đồ và thực hiện các tác vụ là một căn phòng

Nhóm thực hiện vẽ map 2D và 3D của môi trường được miêu tả ở mục 4.4.1 bằng thuật toán Rtab-Map. Sau khi có bản đồ, nhóm thực hiện so sánh với hình dạng và kích thước của môi trường thực tế. Bản đồ 2D được sử dụng trong các tác vụ định vị robot, tìm đường đến đích và bám theo người sau này.

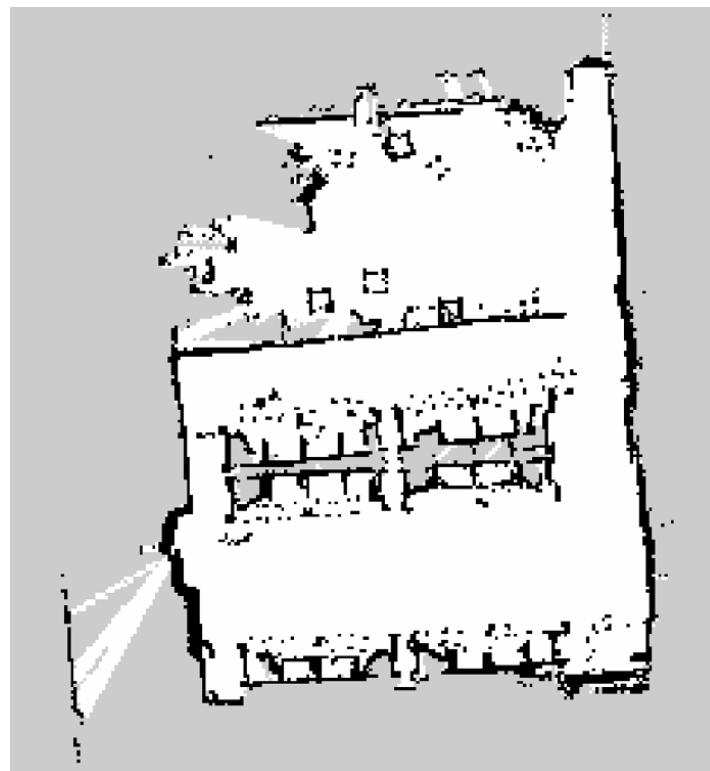
Có 3 phương pháp điều khiển robot để vẽ bản đồ:

- Điều khiển trực tiếp thông qua máy tính nhúng trên robot
- Điều khiển thông qua một máy tính làm server được kết nối với robot qua chuẩn udp
- Sử dụng node thesis_robot_explorer để robot tự hành và tìm đường đi vẽ bản đồ

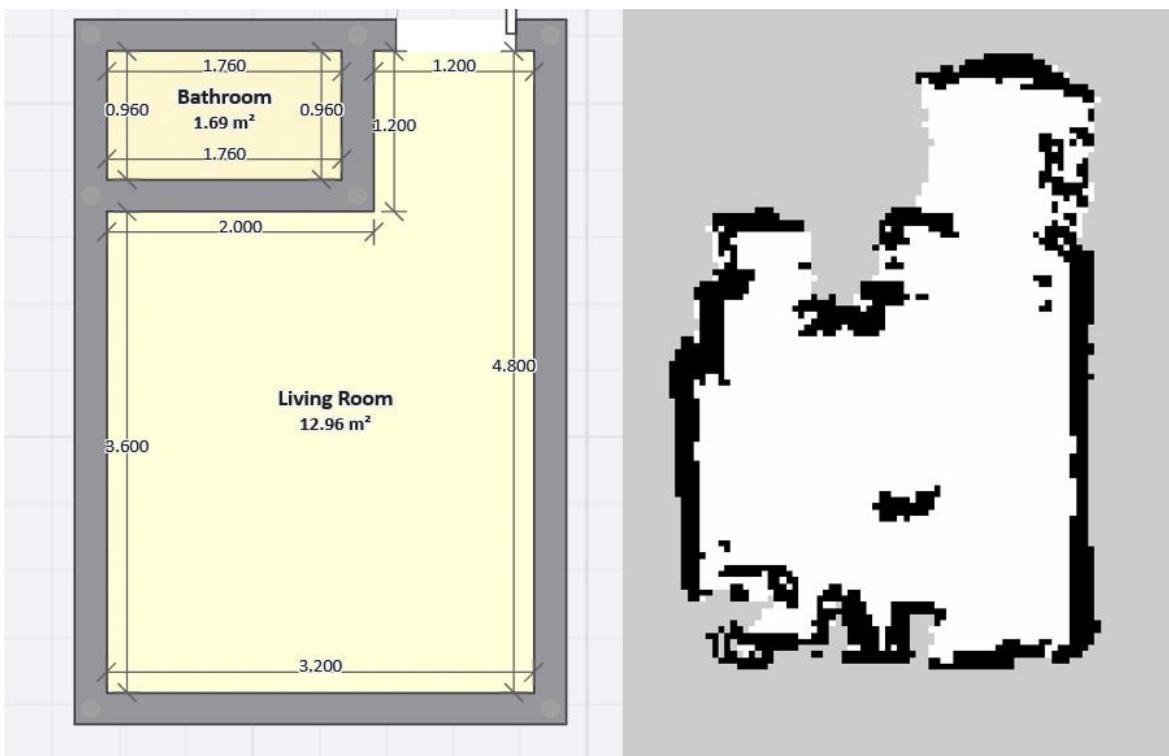
4.4.2. Kết quả vẽ bản đồ 2D

Sau khi điều khiển robot di chuyển khắp môi trường như được nêu trong mục 4.4.1 thì nhóm thu được hai bản đồ 2D như

Hình 4.7 và Hình 4.8 tương ứng với hai môi trường thử nghiệm. Hai hình này là bản vẽ của môi trường dùng để đối chiếu hình dạng và kích thước với bản đồ, trong đó, phần màu đen là vật cản, phần màu trắng là nơi robot có thể di chuyển.



Hình 4.7. Kết quả vẽ bản đồ 2D tại phòng thí nghiệm 207B3 ĐH Bách Khoa TP.HCM

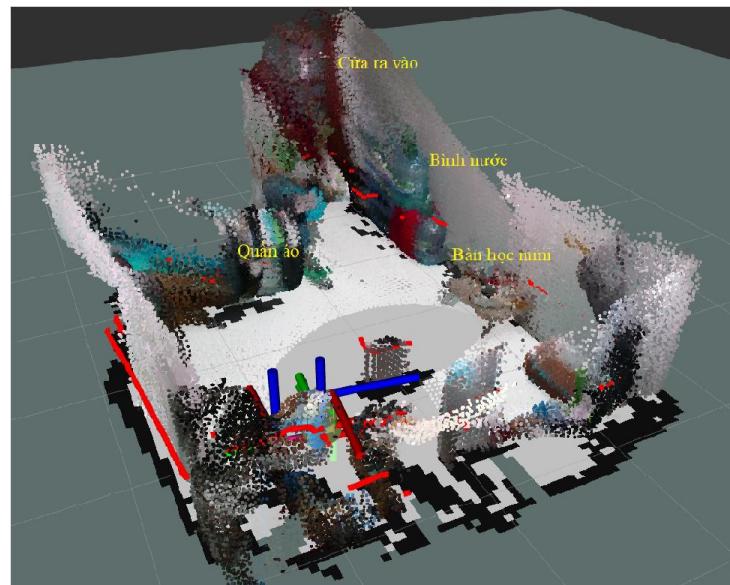


Hình 4.8. Kết quả vẽ bản đồ 2D tại căn phòng trọ và kích thước thật của nó

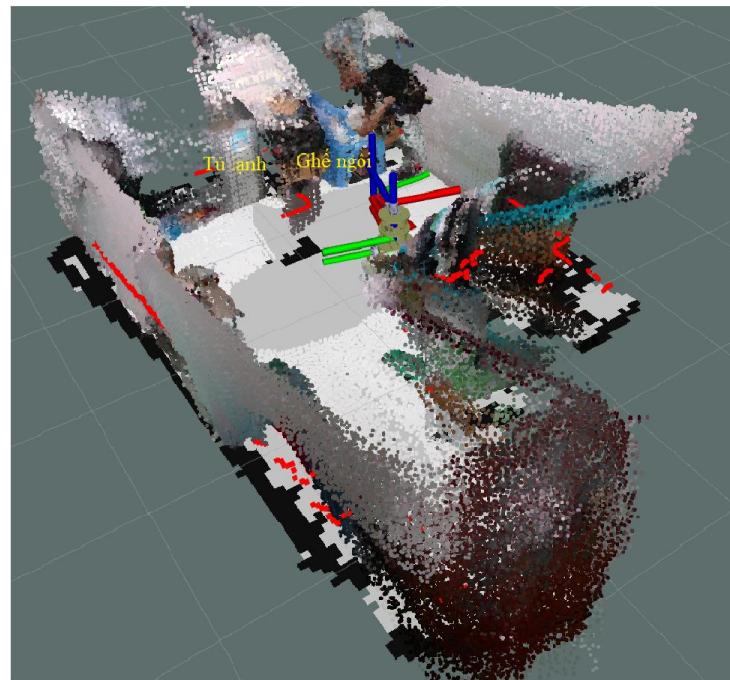
Với môi trường là phòng thí nghiệm 207B3 ĐH Bách Khoa TP.HCM, nhóm sử dụng phương pháp tính toán odometry thông qua thuật toán scan matching dùng dữ liệu từ Lidar và vẽ bản đồ với kích thước lớn, có nhiều vật cản, thể hiện rõ ràng kết quả vẽ bản đồ dùng thuật toán RTAB-Map. Tuy nhiên nhóm mới chỉ vẽ bản đồ, chưa kịp đo đặc số liệu để đánh giá cũng như thực hiện các tác vụ về sau. Còn đối với môi trường là căn phòng trọ, nhóm sử dụng odometry tính toán từ cảm biến IMU và Encoder của robot, được mạch điều khiển STM32 xử lý và gửi lên ROS thông qua giao tiếp rosserial.

4.4.3. Kết quả vẽ bản đồ 3D

Cùng với bản đồ 2D, nhóm thu được bản đồ 3D của môi trường cần vẽ như hình dưới đây. Bản đồ 3D là pointcloud chứa các đặc trưng SIFT được trích xuất từ môi trường cần vẽ. Đối với bản đồ 3D, nhóm chưa tìm được phương pháp đánh giá sai số phù hợp, chỉ đánh giá sự giống nhau tương đối giữa bản đồ và thực tế thông qua người đánh giá.



a) Góc nhìn thứ nhất của bản đồ 3D



b) Góc nhìn thứ hai của bản đồ 3D

Hình 4.9. Kết quả vẽ bản đồ 3D tại căn phòng trọ

4.4.4. Nhận xét và kết luận

Bản đồ 2D thu được đã vẽ lại được hình dạng của môi trường bao gồm cả vật cản tĩnh. Tuy nhiên, bản đồ thu được có sai lệch với môi trường thực tế như bảng dưới đây.

Bảng 4.5. Kết quả kích thước bản đồ 2D vẽ được tại căn phòng tro

Tên chi tiết	Giá trị thực (m)	Giá trị đo từ bản đồ (m)	Sai số (m)	Phần trăm sai số
Chiều dài phòng	4.8	4.8541	0.1459	3.04%
Chiều rộng phòng	3.2	3.3232	0.0232	0.7%
Chiều rộng cửa	0.3	0.2845	0.0155	5.17%

Bản đồ 3D khái quát hóa được đặc trưng của môi trường, có thể hình dung được môi trường thực tế thông qua quan sát bản đồ. Thông qua 10 người quan sát và đánh giá được nhóm khảo sát, bản đồ có độ chính xác tầm 70%, có thể giúp người nhìn hình dung được hình dạng của môi trường đã vẽ ngay lập tức.

Tác vụ vẽ bản đồ vẫn còn gặp nhiều sai số từ chủ quan đến khách quan do giới hạn phần cứng và môi trường cần vẽ. Đặc biệt cần nói đến là nhiều từ trường khiến cho IMU ghi nhận từ trường bị lệch tại các vị trí khác nhau, làm cho odometry tính toán được khi robot di chuyển có sai số và bản đồ vẽ ra sẽ bị lệch đi nhiều. Từ đó nhóm thực hiện thêm phương pháp tính toán odometry dùng thuật toán scan matching để loại bỏ nhiễu từ trường, tuy nhiên phương pháp này tồn tại hạn chế là không thể sử dụng cùng tác vụ định vị. Nhóm quyết định sử dụng odometry tính từ IMU và Encoder để vẽ bản đồ có kích thước nhỏ và dùng odometry tính từ thuật toán scan matching để vẽ bản đồ có kích thước lớn. Mặc dù tồn tại những hạn chế như trên, robot vẫn hoàn thành mục tiêu đặt ra, các kết quả thu được vẫn có tính hữu dụng cao cho các tác vụ sau và mang lại nhiều số liệu quan trọng của môi trường.

4.5. Kết quả định vị robot trong bản đồ

4.5.1. Thu thập số liệu

Để đánh giá khả năng định vị vị trí robot trong bản đồ, nhóm thực hiện thí nghiệm khởi tạo robot trong môi trường đã vẽ bản đồ 2D. Môi trường này gồm các vật cản tương tự như khi vẽ bản đồ và điều khiển robot xoay vòng tròn 360° để robot lấy mẫu và thực hiện định vị. Nhóm thực hiện khởi tạo robot tại 5 vị trí bất kỳ và tại mỗi vị trí tiến hành định vị robot 5 lần để tính sai số và đánh giá.

Sau khi nhóm thực hiện các bước trên thì thu được kết quả như sau:

Bảng 4.6. Kết quả định vị của robot

Lần thử nghiệm	Tọa độ x (m)		Tọa độ y (m)		Góc yaw (rad)	
	Thực tế	Tính toán	Thực tế	Tính toán	Thực tế	Tính toán
Lần 1	0.2	0.1825	0.1	0.1211	0.7854	0.8883
Lần 2	1.2	1.0715	0.4	0.3921	1.0472	0.9120
Lần 3	0.4	0.3765	1.3	1.2844	5.4105	5.5142
Lần 4	0.5	0.5041	0.5	0.4815	1.92	2.2088
Lần 5	0.7	0.6769	0.3	0.2784	1.3963	1.3280
Sai số RMS	0.0599		0.0177		0.1598	

4.5.2. Đánh giá và nhận xét

Sau quá trình thu thập số liệu của tác vụ định vị tại các vị trí bất kỳ trong bản đồ đã vẽ, sai số tính toán được là bé hơn 6cm đối với tọa độ x và y và bé hơn 10° đối với góc yaw. Tác vụ định vị phụ thuộc vào bản đồ đã vẽ có chính xác không, dữ liệu laser quét các vật cản trong bản đồ luôn có nhiều đi cùng và môi trường định vị có khác môi trường đã vẽ hay không, đã tạo ra sai số lớn đối với tác vụ này. Do hạn chế về phần cứng (nhóm chỉ sử dụng RPLidar A1) nên chưa thể khắc phục được sai số trên. Cùng với điều trên là hạn chế về môi trường nếu có quá ít đặc trưng nhận biết, robot sẽ gặp khó khăn trong việc định vị. Đồng thời, nếu bản đồ vẽ quá lớn thì robot sẽ gặp trường hợp định vị robot ngay giữa bản đồ, nơi có nhiều vị trí bị chiếm trên bản đồ nhất, làm cho việc định vị trở nên bất khả thi, nhóm vẫn chưa tìm được phương án phù hợp cho trường hợp này.

4.6. Kết quả điều hướng robot

Đối với nhiệm vụ điều hướng robot, nhóm tiến hành lựa chọn điểm đầu và điểm đích trên bản đồ đã vẽ để robot di chuyển, từ đó thu thập số liệu về vị trí của robot và đánh giá sai số. Môi trường này sẽ gồm ba trường hợp là không vật cản, có vật cản tĩnh và có vật cản động. Tương tự như khi lấy số liệu định vị robot, nhóm sẽ thực hiện 5 lần tại các vị trí ngẫu nhiên trên bản đồ bao gồm cả 3 trường hợp nêu trên để đánh giá khách quan kết quả đã đạt được, từ đó tính toán sai số và đánh giá.

4.6.1. Kết quả thu thập số liệu

Bảng 4.7. Kết quả điều hướng robot

Lần thử nghiệm	Tọa độ x (m)		Tọa độ y (m)		Góc yaw (rad)	
	Mục tiêu	Đạt được	Mục tiêu	Đạt được	Mục tiêu	Đạt được
Lần 1	0.0531	0.0359	0.3673	0.3346	2.3538	2.4085
Lần 2	1.2509	1.2259	0.0057	0.0453	1.5961	1.6148
Lần 3	0.3936	0.3768	0.7178	0.6958	1.2865	1.3185
Lần 4	0.8309	0.7659	0.9733	1.0076	0.3412	0.0551
Lần 5	2.5142	2.4907	0.6618	0.7064	1.4223	1.5111
Sai số RMS	0.0346		0.0355		0.1372	

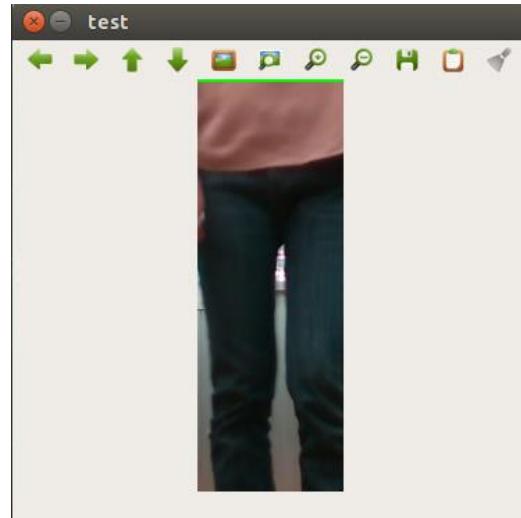
4.6.2. Đánh giá và nhận xét

Robot đã hoàn thành tác vụ điều hướng với sai số tọa độ bé hơn 4cm và sai số góc bé hơn 8° , đạt được mục tiêu điều hướng chính xác đến mục tiêu cần đi đến. Riêng vì hạn chế đã nêu ở tác vụ định vị robot nên ở tác vụ điều hướng sẽ có khi xảy ra trường hợp robot không thể hoạch định đường đi chính xác do dữ liệu laser quét các vật cản xung bị nhiễu và vận tốc của robot sai mong muốn (hay nói cách khác là do nhiễu encoder dẫn đến vận tốc feedback và vận tốc điều khiển sai). Đồng thời, do môi trường hoạt động nhỏ nên điều chỉnh lại kích thước costmap dẫn đến có khi không thể hoạch định đường đi cục bộ khi robot gặp vật cản động. Tác vụ này vẫn còn xuất hiện một ít sai sót và cần khởi động lại hệ thống, nhưng nhìn chung, việc điều hướng robot đã đạt được kết quả mong muốn với sai số đủ nhỏ.

4.7. Kết quả thuật toán bám theo người

4.7.1. Kết quả thu thập số liệu

Với người được chọn lấy đặc trưng tham chiếu, và là target person như Hình 4.10, một ma trận $\mathbf{Colors}_{reference}$ được tìm nhờ phương pháp Kmeans-Clustering, mỗi hàng của ma trận là giá trị ba kênh màu (R, G, B) của từng cụm màu center. Ma trận này sẽ được dùng để tính toán ma trận $\mathbf{feature}_{reference}$ như được nêu ở mục 3.9.2. Hình 4.11 biểu diễn ba màu sắc tương ứng trong ma trận $\mathbf{Colors}_{reference}$.



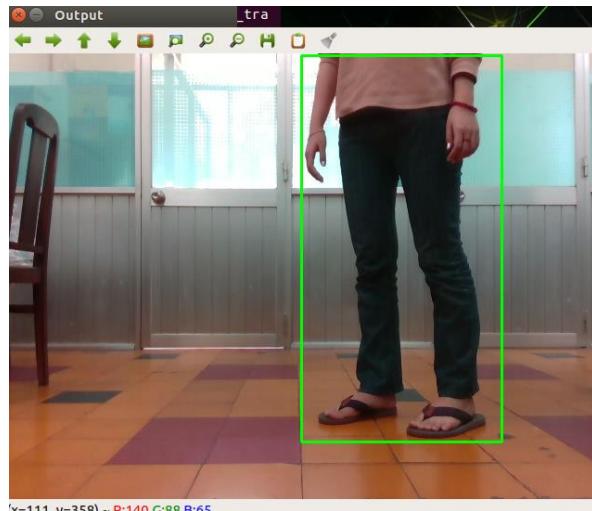
Hình 4.10 Ảnh được lấy trích đặc trưng tham chiếu

$$Colors_{reference} = \begin{bmatrix} 18.583 & 25.399 & 25.277 \\ 153.678 & 119.128 & 113.841 \\ 83.882 & 72.085 & 63.039 \end{bmatrix}$$



Hình 4.11 Ba màu center của đặc trưng tham chiếu

Khi cho target person đi khuất khỏi vùng nhìn thấy của camera, robot sẽ xoay để tìm lại người. Nếu người xuất hiện như trong Hình 4.12, với ma trận $Colors_1$ và ba cụm màu được biểu diễn như hình Hình 4.13, sự khác nhau giữa ma trận $feature$ của người này với $feature_{reference}$ là $distance_1 = 52.825$. $distance_1 < 70$, vì vậy người này được nhận diện là target person.



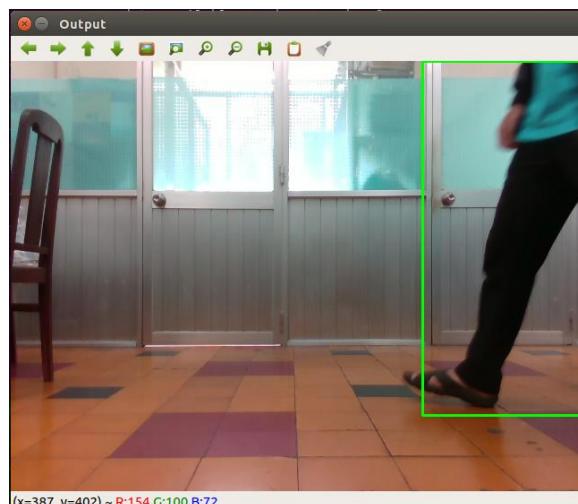
Hình 4.12 Ảnh người được nhận diện là target person

$$\mathbf{Colors}_1 = \begin{bmatrix} 22.603 & 29.124 & 32.626 \\ 123.106 & 99.065 & 96.604 \\ 135.992 & 162.664 & 161.905 \end{bmatrix}$$



Hình 4.13 Ba màu center đặc trưng của người được nhận diện là target person

Nếu người xuất hiện như Hình 4.14, với ma trận \mathbf{Colors}_2 và ba cụm màu được biểu diễn như hình Hình 4.15, với $distance_2 = 155.675$. $distance_2 > 70$, vì vậy người này không được nhận diện là target person. Robot sẽ tiếp tục xoay để tìm người.



Hình 4.14 Ảnh người không được nhận diện là target person

$$Colors_2 = \begin{bmatrix} 10.082 & 12.737 & 10.919 \\ 19.358 & 153.081 & 153.004 \\ 48.842 & 62.370 & 63.234 \end{bmatrix}$$



Hình 4.15 Ba màu center đặc trưng của người không được nhận diện là target person

Nếu người xuất hiện như Hình 4.16, có màu sắc quần áo tương đồng với target person ở Hình 4.10, với ma trận $Colors_3$ và ba cụm màu được biểu diễn như Hình 4.17, sự khác nhau giữa ma trận **feature** của người này với **feature_{reference}** là $distance_3 = 63.507$. $distance_2 < 70$, vì vậy người này được nhận diện nhầm thành target person dù không phải là target person.



Hình 4.16 Ảnh người được nhận diện sai thành target person

$$Colors_3 = \begin{bmatrix} 47.821 & 49.156 & 53.946 \\ 180.090 & 91.745 & 112.988 \\ 61.114 & 70.067 & 75.112 \end{bmatrix}$$



Hình 4.17 Ba màu center đặc trưng của người được nhận diện sai thành target person

Để đánh giá khả năng nhận diện target person của thuật toán, thực hiện đánh giá trên 3 target person khác nhau và lấy 60 lần dự đoán như Bảng 4.8. Target person 1 được dự đoán ở lần 1 – 20 (lượt 1), target person 2 từ lần 21 – 40 (lượt 2), target person 3 từ lần 41 – 60 (lượt 3). Phép tính F1-score được sử dụng cho việc đánh giá. F1-score được xem là trung bình của Precision và Recall, nó nằm trong khoảng [0,1], khi có giá trị càng cao thì chất lượng thuật toán càng tốt. F1-score được tính bởi công thức:

$$\frac{2}{F_1} = \frac{1}{\text{precision}} + \frac{1}{\text{recall}} \text{ hay } F_1 = 2 \frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Trong đó, Precision được định nghĩa là tỉ lệ mẫu được xác định đúng (True Positive) trong số những mẫu được thuật toán xác định là target person (True Positive + False Positive). Và Recall được định nghĩa là tỉ lệ mẫu được xác định đúng (True Positive) trong số những mẫu thực sự là target person (True Positive + False Negative). Precision cao đồng nghĩa với việc độ chính xác của các mẫu được xác định bởi thuật toán là cao. Recall cao đồng nghĩa với việc True Positive Rate cao, tức tỉ lệ bỏ sót các mẫu thực sự là target person là thấp.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Thuật toán nhận diện sai 17 lần. Trong đó, 7 lần thuật toán không nhận diện lại được target person (False Negative – FN), và 10 lần thuật toán nhận diện nhầm người không phải target person thành target person (False Positive – FP). Số lần phân biệt đúng là 43. Số lần nhận diện đúng target person là 30 (True Positive - TP).

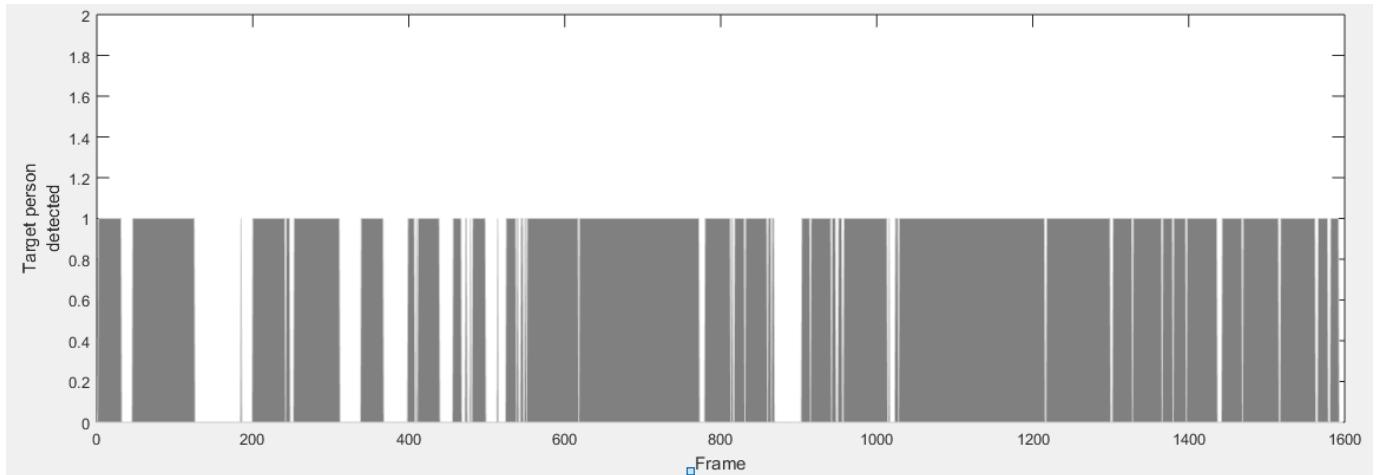
Bảng 4.8 Kết quả nhận diện target person ở 60 lần dự đoán (target person: 1/ không phải target person: 0)

Lần	Nhận	Dự đoán	Lần	Nhận	Dự đoán	Lần	Nhận	Dự đoán
1	1	1	21	1	1	41	0	1
2	1	1	22	1	1	42	0	0
3	0	0	23	0	1	43	1	1
4	1	1	24	1	0	44	1	1
5	0	0	25	1	0	45	0	0
6	1	1	26	1	1	46	1	1
7	1	1	27	0	1	47	0	1
8	1	1	28	0	1	48	1	1
9	1	1	29	0	0	49	1	1
10	0	1	30	1	0	50	0	0
11	1	1	31	1	1	51	1	1
12	1	1	32	1	0	52	1	1
13	1	1	33	0	0	53	0	0
14	0	0	34	1	0	54	1	1
15	1	1	35	1	1	55	1	1
16	0	0	36	0	1	56	1	1
17	1	0	37	1	0	57	0	1
18	1	1	38	1	1	58	0	0
19	0	1	39	0	1	59	1	1
20	1	1	40	1	1	60	0	0
FN = 7 FP = 10 TP = 30 Precision = 75% Recall = 81.08% F1-score = 77.922%								

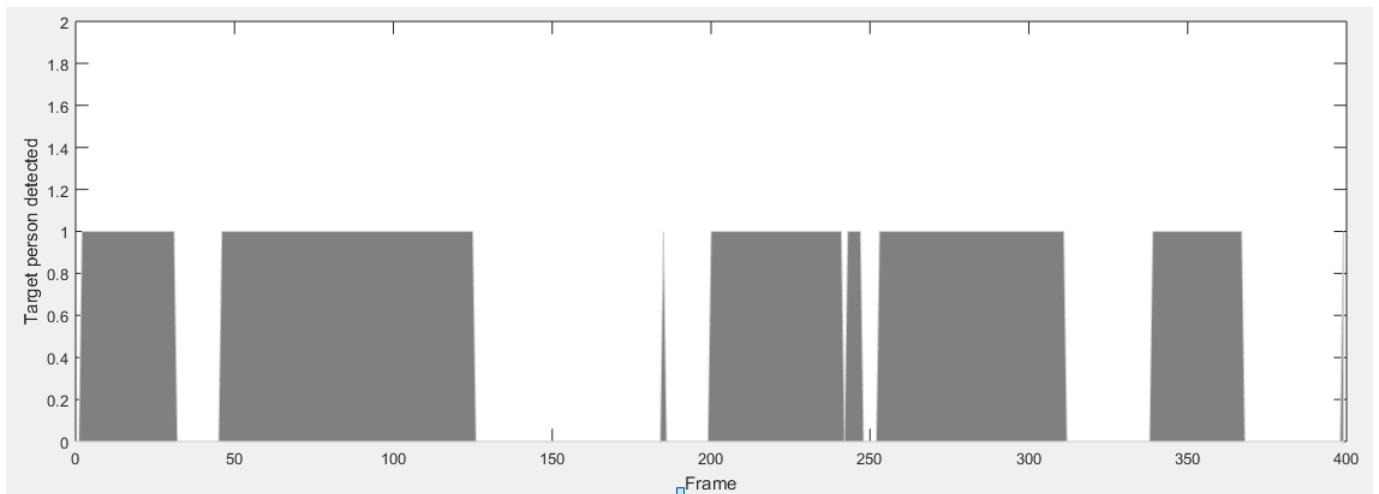
Theo Bảng 4.8, có 6 trong 7 False Negative thuộc lượt 2, nhóm đưa ra giải thích như sau: đặc trưng tham chiếu được lấy ở target person 2 không tốt vì ảnh được mang trích đặc trưng tham chiếu có thể chứa nhiều background; ngoài ra, ở những góc độ, hướng ánh sáng khác nhau, quần áo của target person 2 mang màu sắc không tương đồng.

Các tín hiệu ở mỗi khung ảnh được lưu lại ở dạng file .csv và vẽ ra trong phần mềm Matlab để đánh giá khả năng thành công bám theo target person của thuật toán (bao gồm False

Positive). Với tốc độ xử lý của thuật toán vào khoảng 7 FPS, trong 1600 khung ảnh liên tục (khoảng 4 phút), những khoảng thời gian bị mất dấu target person là ngắn và không đáng kể như Hình 4.18. Đồ thị trong hình Hình 4.18 được phóng to trong hình Hình 4.19.



Hình 4.18 Đồ thị thể hiện khả năng bám theo target person mỗi khung ảnh trong 1600 khung.



Hình 4.19 Đồ thị thể hiện khả năng bám theo target person mỗi khung ảnh trong 400 khung.

Bảng 4.9 Đánh giá kết quả thuật toán bám theo người

Tốc độ xử lý	≈ 7 FPS
F1-score	$\approx 77.922\%$
Số khung ảnh tối đa liên tiếp mất dấu target person	≈ 65 khung
Thời gian tối đa liên tiếp mất dấu target person	≈ 10 s

4.7.2. Đánh giá và nhận xét

Tuy số liệu đánh giá cho ra kết quả F1-score khá tốt, tuy nhiên, việc đánh giá còn mang tính chủ quan vì chỉ được thực hiện trong một môi trường duy nhất. Kết quả sẽ phụ thuộc rất nhiều vào điều kiện ánh sáng, chất lượng của tấm ảnh đầu tiên được lấy trích đặc trưng tham chiếu, cũng như là màu sắc quần áo target person và những người xung quanh đang mặc.

Việc sử dụng màu sắc làm đặc trưng nhận diện có hai nhược điểm là: một người khác có màu sắc quần áo tương đồng với target person sẽ bị nhận diện nhầm thành target person; quần áo của target person nếu nhìn từ các hướng khác nhau, hoặc dưới điều kiện ánh sáng khác nhau có màu sắc không tương đồng thì sẽ không được nhận diện là cùng một người.

Việc camera được đặt ở vị trí quá thấp gây khó khăn cho tác vụ phát hiện người của mô hình CNN. Hình ảnh nửa thân dưới của người khó có thể được nhận diện chính xác thành một người.

Thời gian xử lý của thuật toán khá chậm (7 FPS), nên thuật toán sẽ gặp khó khăn trong việc bám theo target person di chuyển với tốc độ nhanh. Thuật toán có tốc độ xử lý chậm đầu tiên là vì khả năng của máy tính nhúng không đủ đáp ứng thực hiện thời gian thực các phép tính cho mô hình CNN. Ngoài ra, thuật toán được viết trên ngôn ngữ Python với khả năng tối ưu lệnh và phép tính thấp so với các ngôn ngữ như C, thời gian thực hiện vì vậy cũng chậm hơn.

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Nhóm đã thực hiện tất cả các quá trình gồm thiết kế và xây dựng phần cứng, lập trình firmware, xây dựng phần mềm trên nền tảng ROS, phát triển các tác vụ của xe tự hành và tích hợp thị giác máy thực hiện nhiệm vụ bám theo người. Trong quá trình nghiên cứu thực hiện của mình, nhóm đã rút được các kết quả mong muốn cũng như các hạn chế gặp phải, từng bước tìm các khắc phục và đã hoàn thành xây dựng robot tự hành với các tác vụ đề ra ban đầu. Nhóm rút ra kết luận và các đánh giá về kết quả như sau:

- Về thiết kế phần cứng, nhóm lựa chọn thiết kế mô hình dạng hình tròn giúp robot có khả năng di chuyển linh hoạt và ổn định cho môi trường trong nhà. Nhóm đã sử dụng máy tính nhúng để tối ưu hóa không gian của robot, sắp xếp các linh kiện cảm biến một cách tối ưu và tận dụng tối đa hai mặt của mỗi lớp của robot. Tuy nhiên, thiết kế này gặp phải một số hạn chế là việc đi dây kết nối trong robot giữa các tầng là phức tạp. Đồng thời robot chạy bằng acquy nên khối lượng robot nặng hơn dự tính rất nhiều cũng như thời gian sử dụng robot bị hạn chế, cần thao acquy để sạc thường xuyên.
- Về khả năng lập bản đồ của robot, qua quá trình phân tích đánh giá số liệu của bản đồ thu được thì nhóm nhận thấy rằng bản đồ có kích thước tương đối chính xác so với môi trường thực tế, đặc biệt là bản đồ 3D giúp nhận thức về môi trường. Tuy nhiên, việc xây dựng bản đồ cũng có những hạn chế nhất định. Đầu tiên, hạn chế về phần cứng là IMU và Encoder cũng như nhiễu của môi trường làm robot vẽ bản đồ bị lệch khi di chuyển đến các điểm khác nhau, đồng thời hạn chế tốc độ di chuyển của robot để có thể lập được một bản đồ như yêu cầu. Tiếp theo, cảm biến Lidar tính toán khoảng cách thông qua tia laser, nên với môi trường có vật liệu phản xạ hoặc xuyên thấu thì việc lập bản đồ không khả thi. Cuối cùng, hạn chế về phần xử lý hay nói cách khác là thuật toán Rtab-Map phức tạp và sử dụng hết hiệu năng của máy tính nhúng, nhưng chỉ có thể cập nhật bản đồ với tần số gần 4Hz. Hạn chế này có thể được khắc phục thông qua việc tối ưu hóa qua các bước không cần thiết của thuật toán hoặc thay vào đó là máy tính nhúng có khả năng xử lý cao hơn.

- Về khả năng định vị và điều hướng robot, sau quá trình kết hợp các tác vụ thì robot đã hoàn thành việc tự hành đi đến vị trí mục tiêu với khả năng tránh vật cản tự động. Trong quá trình điều hướng, robot đã tự hành và di chuyển đến mục tiêu thành công, cùng với đó là khả năng tránh vật cản. Tuy nhiên trong quá trình thực hiện lấy kết quả, nhóm nhận thấy rằng robot vẫn còn nhiều hạn chế, đáng kể đến là việc hoạch định đường đi chính xác nhưng robot không đi theo đường đi đề ra do việc xác định vật cản từ cảm biến Lidar có sai số. Cùng với hạn chế trên là khi robot bị bao vây bởi vật cản động thì việc hoạch định đường đi cục bộ bất khả thi, robot phải thực hiện xóa costmap và hoạch định lại đường đi, việc này còn tốn thời gian. Đồng thời, ảnh hưởng của sai số dữ liệu từ Lidar và sai số bản đồ thể hiện rõ rệt khi quan sát robot vừa di chuyển vừa định vị thì dữ liệu LaserScan bị lệch so với bản đồ đã vẽ. Ngoài ra, có một số trường hợp robot không thể đi đến điểm mục tiêu do không thể xác định vật cản thấp hơn Lidar cũng như sai số vận tốc từ nhiều encoder, dẫn đến việc định vị thất bại và phải khởi động lại hệ thống. Các hạn chế này cần thêm thời gian để nhóm nghiên cứu và tìm phương pháp xử lý khắc phục.
- Về khả năng nhận diện và bám theo người, giới hạn ở khả năng xử lý của máy tính nhúng, việc sử dụng mô hình CNN và việc thuật toán được viết trên ngôn ngữ Python có tính tối ưu kém khiến cho thuật toán có tốc độ xử lý chậm và chỉ đạt 7 FPS. Mô hình CNN tuy đã được cấu hình để triển khai trên GPU của máy tính nhúng, tuy nhiên vẫn là nguyên nhân chính yếu làm thuật toán xử lý chậm. Thuật toán chung được chính sinh viên thiết kế và triển khai còn nhiều hạn chế về mặt logic và tối ưu. Việc sử dụng màu sắc làm đặc trưng nhận diện target person có rất nhiều hạn chế như được nêu ở mục 4.7.2: một người khác có màu sắc quần áo tương đồng với target person sẽ bị nhận diện nhầm thành target person; quần áo của target person nếu nhìn từ các hướng khác nhau, hoặc dưới điều kiện ánh sáng khác nhau có màu sắc không tương đồng thì sẽ không được nhận diện là cùng một người.

5.2. Hướng phát triển

Để hoàn thiện một robot tối ưu, hoạt động tốt và ổn định trong nhiều điều kiện khác nhau như văn phòng, bệnh viện, siêu thị thì nhóm cần phải nghiên cứu phát triển thêm về lâu dài, tìm cách tối ưu hóa mô hình cũng như các phương pháp xử lý thích hợp:

- Cải thiện thiết kế phần cứng cao hơn và chịu lực tốt hơn để robot có thể di chuyển và vận chuyển hàng hóa hoặc đồ dùng của người sử dụng. Thay nguồn pin acquy bằng nguồn pin lipo dung lượng lớn để tạo nhiều không gian trong robot hơn và sử dụng với thời gian dài hơn.
- Tích hợp việc kết nối IoT và tạo ứng dụng trên điện thoại, nơi mà người dùng có thể quan sát và điều khiển từ xa thông qua internet. Việc này yêu cầu nhóm nghiên cứu thêm để tạo các tính năng tương tác giữa người dùng và robot thông qua giao diện điều khiển được yêu cầu.
- Tối ưu hóa việc lập bản đồ thông qua việc loại bỏ các bước tính toán chưa cần thiết trong thuật toán, sử dụng IMU có độ phân giải cao và sai số thấp, không chịu ảnh hưởng nhiều của từ trường để có thể tính toán odometry một cách hoàn hảo, là tiền đề quan trọng để xây dựng bản đồ chính xác.
- Cải thiện tác vụ bám theo người thông qua việc lựa chọn một mô hình hoặc thuật toán phát hiện người đơn giản nhưng tối ưu hơn, không cần thiết phải là mô hình CNN. Tìm ra đặc trưng khác tốt hơn cho việc nhận diện target person, sử dụng các phương pháp như: lắp thêm một camera riêng cho tác vụ này sao cho việc lật camera nghiêng lên trên không ảnh hưởng đến tác vụ bản đồ, nhờ vậy có thể sử dụng được đặc trưng mạnh mẽ của gương mặt người; dùng hình dán có biểu tượng cụ thể dán lên target person làm đặc trưng để hỗ trợ thuật toán khi cần thiết. Sử dụng phần cứng máy tính nhúng có khả năng xử lý tốt hơn. Chỉnh sửa lại thuật toán sao cho tối ưu hơn về mặt logic lẫn thời gian xử lý, cũng như chuyển đổi sang một ngôn ngữ khác như C, để tăng thời gian thực thi chương trình.
- Tích hợp phương pháp nhận diện vật thể và tác vụ điều hướng, từ đó nhóm nghiên cứu xây dựng phần mềm giúp robot hoạt động trong siêu thị, tránh vật cản động một cách

linh hoạt và đưa người dùng đi đến nơi để vật phẩm cần tìm một cách nhanh nhất. Tác vụ này cần xây dựng mô hình nhận diện vật thể số lượng lớn, thu thập các số liệu cần thiết về lâu dài nhưng tính khả thi trong thực tế cao.

- Xây dựng phương án robot nhận thức phần trăm pin của bản thân và tự sạc khi hết pin.

PHỤ LỤC

PHỤ LỤC I

Hệ tọa độ NED và hệ tọa độ IMU

Hệ tọa độ trong Hình 2.24 là hệ tọa độ NED (n-frame), có trục x hướng theo hướng Bắc địa lý, trục y hướng theo hướng Đông địa lý và trục z hướng xuống tâm trái đất.

Hệ tọa độ vật thể là hệ tọa độ gắn trên cảm biến IMU (b-frame), có trục x là hướng chuyển động tới của vật thể, trục z hướng xuống, và trục y được xác định theo quy tắc tam diện thuận.

Tùy trường trái đất trong n-frame

Tại mọi nơi trên trái đất, tùy trường có thể được biểu diễn trong n-frame bằng một vector 3 chiều (\mathbf{h}_0). Vector tùy trường không chỉ thẳng về cực Bắc địa lý mà chỉ về cực Bắc từ và được đặc trưng bằng: biên độ tùy trường \mathbf{F}_0 (thường được đo ở đơn vị nano Tesla – nT hoặc Gauss, thường nằm trong khoảng 25000nT – 65000nT), độ từ khuynh \mathbf{I} (góc tạo thành bởi vector tùy trường trái đất với mặt phẳng nằm ngang tại điểm khảo sát), độ từ thiên \mathbf{D} (góc tạo thành giữa hướng Bắc địa lý và hướng Bắc từ).

$$\mathbf{h}_0 = F \begin{bmatrix} \cos(I) \cos(D) \\ \cos(I) \sin(D) \\ \sin(I) \end{bmatrix}$$

Tùy trường trái đất trong b-frame

Vector tùy trường \mathbf{h} trong hệ tọa độ vật thể là:

$$\mathbf{h} = \mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\varphi)\mathbf{h}_0$$

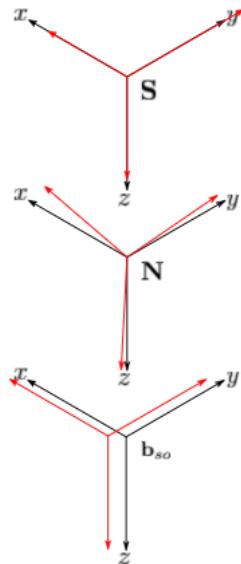
Trong đó, các ma trận \mathbf{R} là các ma trận xoay quanh ba trục của hệ tọa độ vật thể, và tích 3 ma trận \mathbf{R} này cũng là ma trận DCM biến đổi từ hệ tọa độ NED sang hệ tọa độ vật thể. Có thể dễ dàng thấy được, quỹ tích các mẫu thu được khi xoay IMU 360 quanh tất cả các trục là đường thẳng đi qua tâm là một mặt cầu F .

$$\mathbf{h}^T \mathbf{h} = \dots = F^2$$

Equation 0.1

Sai số thiết bị

Sai số thiết bị có thể được mô hình hóa bởi ba thành phần: scale factor là một ma trận đường chéo \mathbf{S} , sự không trực giao của các trục cảm biến \mathbf{N} và offset \mathbf{b}_{so} .



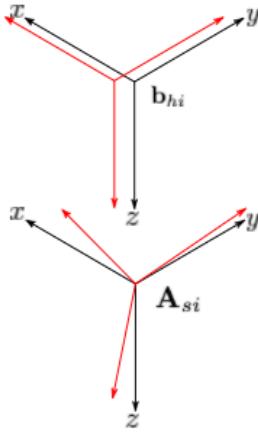
Hình 0.1 Mô tả sai số thiết bị

Nhiều từ trường

Nhiều từ trường có thể được mô hình hóa như sau, với \mathbf{b}_{hi} là hard iron và \mathbf{A}_{si} là soft iron:

$$\mathbf{b}_{hi} = [b_{hi_x} \ b_{hi_y} \ b_{hi_z}]^T$$

$$\mathbf{A}_{si} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$



Hình 0.2 Mô tả nhiễu từ trường

Trong khi \mathbf{h} là giá trị đúng, giá trị từ trường đo được từ cảm biến khi có nhiễu có thể dễ dàng được viết như sau:

$$\mathbf{h}_m = \mathbf{SN}(\mathbf{A}_{si}\mathbf{h} + \mathbf{b}_{hi}) + \mathbf{b}_{so}$$

$$\mathbf{h}_m = \mathbf{A}\mathbf{h} + \mathbf{b}$$

Equation 0.2

Với:

$$\mathbf{A} = \mathbf{S}\mathbf{N}\mathbf{A}_{si}$$

$$\mathbf{b} = \mathbf{S}\mathbf{N}\mathbf{b}_{hi} + \mathbf{b}_{so}$$

Có thể chứng minh được rằng, sự thay đổi tuyến tính của \mathbf{h} ở phương trình trên khiến quỹ tích các mẫu của \mathbf{h}_m nằm trên một mặt ellipsoid. Vì thế, vấn đề hiệu chỉnh cảm biến có thể được giải quyết bằng phương pháp ellipsoid fit.

Quadric

Quadric là các mặt phẳng có thể được biểu diễn bằng một đa thức bậc 2 trên các biến x, y, z. Nó bao gồm các mặt cầu, ellipsoid và paraboloic, ... Phương trình tổng quát của một quadric \mathbf{S} là:

$$S : ax^2 + by^2 + cz^2 + 2fyz + 2gxz + 2hxy + px + qy + rz + d = 0$$

Phương trình trên có thể được viết dưới dạng ma trận như sau:

$$S : x^T \mathbf{M}x + 2x^T \mathbf{n} + d = 0$$

Equation 0.3

Trong đó,

$$\mathbf{x} = [x \ y \ z]^T$$

$$\mathbf{M} = \begin{bmatrix} a & h & g \\ h & b & f \\ g & f & c \end{bmatrix}, \mathbf{n} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

Giả sử, một mặt có bán kính là 1, với tâm nằm tại gốc tọa độ sẽ có:

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{n} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, d = -1$$

Phương trình mặt S còn có thể được viết lại dưới dạng ma trận thuận túy:

$$S : \mathbf{x}^T \mathbf{Q} \mathbf{x} = 0$$

Với,

$$\mathbf{x} = [x \ y \ z \ 1]^T, \mathbf{Q} = \begin{bmatrix} \mathbf{M} & \mathbf{n} \\ \mathbf{n}^T & d \end{bmatrix}$$

Mối liên hệ giữa \mathbf{A} , \mathbf{b} và các thông số của một mặt ellipsoid

Ta giả sử các giá trị dự đoán của \mathbf{A} và \mathbf{b} lần lượt là $\widehat{\mathbf{A}}$ và $\widehat{\mathbf{b}}$. Khi đó, bát cú kết quả từ trường sau khi calib nào đều là \mathbf{h}_n , giả sử \mathbf{h}_n có giá trị giống với giá trị đúng \mathbf{h} , ta có phương trình liên hệ giữa giá trị đo và giá trị đúng như sau:

$$\mathbf{h} = \mathbf{A}^{-1}(\mathbf{h}_m - \mathbf{b})$$

Equation 0.4

Khi kết hợp Equation 0.1 với Equation 0.2, ta có phương trình:

$$\mathbf{h}_m^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_m - 2 \mathbf{h}_m^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{b} + \mathbf{b}^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{b} - F^2 = 0$$

Phương trình trên có thể được viết lại dưới dạng một mặt quadric dựa trên Equation 0.3:

$$\mathbf{h}_m^T \mathbf{M} \mathbf{h}_m + \mathbf{h}_m^T \mathbf{n} + d = 0$$

Trong đó,

$$\mathbf{M} = \mathbf{A}^{-T} \mathbf{A}^{-1}$$

$$\mathbf{n} = -2 \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{b}$$

$$\mathbf{d} = \mathbf{b}^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{b} - F^2$$

Equation 0.5

Như ở trên đã đề cập, mặt quadric này sẽ là một mặt ellipsoid. Vì vậy, ta cần sử dụng một thuật toán ellipsoid fitting để tìm được xấp xỉ các thông số $\widehat{\mathbf{M}}$, $\widehat{\mathbf{n}}$, \widehat{d} của mặt ellipsoid này. Từ đó ta có thể tìm được $\widehat{\mathbf{A}}^{-1}$ và $\widehat{\mathbf{b}}$. Giá trị F có thể được đặt là 1 để mặt cầu có được sau khi calib sẽ có bán kính là 1. Kết hợp với Equation 0.5, ta tìm được:

$$\widehat{\mathbf{b}} = -\widehat{\mathbf{M}}^{-1} \widehat{\mathbf{n}}$$

$$\widehat{\mathbf{A}}^{-1} = \frac{F}{\sqrt{\widehat{\mathbf{n}}^T \widehat{\mathbf{M}}^{-1} \widehat{\mathbf{n}} - \widehat{d}}} \widehat{\mathbf{M}}^{\frac{1}{2}}$$

PHỤ LỤC II

Bảng 0.1 Thông số kỹ thuật RPLidar A1

Thông số kỹ thuật	Chi tiết
Nguồn cấp	5V
Tầm đo	12m
Tần số lấy mẫu	Max: 10Hz (8000 samples)
Độ phân giải	1°

Bảng 0.2 Thông số kỹ thuật Camera Realsense D435

Thông số kỹ thuật	Chi tiết
Nguồn cấp	5V
Tốc độ khung ảnh	Max: 30FPS
Khoảng độ sâu đo được	Min: 0.105m
Độ phân giải video	Max: 1920x1080

Bảng 0.3 Thông số kỹ thuật Jetson Nano

Thông số kỹ thuật	Chi tiết
Nguồn cấp	5V, 4A
GPU	Maxwell 128-core
RAM	4GB 64-bit LPDDR4
CPU	Quad-core ARM Cortex-A57
Cổng kết nối, hiển thị	Ethernet, Camera, USB 3.0 2.0, HDMI
I/O giao tiếp	GPIO, I2C, UART, SPI

Bảng 0.4 Bảng thông số kỹ thuật MPU9250

Thông số kỹ thuật	Chi tiết
Nguồn cấp	2.4-3.6V, max: 3.5 mA
Khoảng giá trị Gyroscope	Max: ± 2000 °/sec, Min: ± 250 °/sec
Khoảng giá trị Accelerometer	Max: ± 16 g, Min: ± 2 g
Khoảng giá trị Magnetometer	± 4800 μ T
Giao thức đọc dữ liệu	I2C/SPI

Bảng 0.5 Thông số kỹ thuật vi điều khiển STM32F407VG

Thông số kỹ thuật	Chi tiết
Nguồn cấp	5V/3V
CPU	ARM Cortex-M4 + DSP Core
Tần số clock	Max: 168Mhz
Bộ nhớ Flash	1024KB
RAM	192KB

Bảng 0.6 Thông số kỹ thuật cầu H HI216

Thông số kỹ thuật	Chi tiết
Nguồn cấp	12V – 48V DC
Dòng liên tục	15A
Dòng định	20A
Công suất tối đa	600W
Điện áp kích	3.3V – 5.5V
Tần số hoạt động tối đa	100Khz

Bảng 0.7 Thông số kỹ thuật động cơ DCM50-775

Thông số kỹ thuật	Chi tiết
Nguồn cấp động cơ	12V DC
Công suất	60W
Nguồn cấp Encoder	5V DC
Hệ số chia hộp số	29:1
Tốc độ động cơ qua hộp số	Max: 80 rpm
Số xung encoder trên một vòng	13ppr

TÀI LIỆU THAM KHẢO

- [1] M. Labb , F. Michaud, Preprint: RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online, *Journal of Field Robotics*, 2019.
- [2] S.Thrun, W.Burgard, D.Fox, Probabilistic Robotics, Page 93-95: Probabilistic Kinematics, 1999-200.
- [3] S.Thrun, W.Burgard, D.Fox, Probabilistic Robotics, Page 121-148: Measurements, 1999-200.
- [4] S.Thrun, W.Burgard, D.Fox, Probabilistic Robotics, Page 77-89: Particle Filter, 1999-200.
- [5] D.Fox, "Adapting the Sample Size in Particle Filters Through KLD-Sampling," *The International Journal of Robotics Research*, 2003.
- [6] G.Peng. W.Zheng, Z.Lu, J.Liao, L.Hu, G.Zhang, D.He, "An Improved AMCL Algorithm Based on Laser Scanning Match in a Complex and Unstructured Environment," vol. 2018, p. 11, 2018.
- [7] S.Kardy, A.Abdallah, C.Joumaa, "On The Optimization of Dijkstra's Algorithm".
- [8] D.Fox, W.Burgard, S.Thrun, "The Dynamic Window Approach to Collision," *IEEE Robotics & Automation Magazine*, vol. Dynamic Window Algorithm, pp. Page 8-21, 1997.
- [9] D.Fox, W.Burgard, S.Thrun, "The Dynamic Window Approach to Collision," *IEEE Robotics & Automation Magazine*, vol. Motion Equations, pp. Page 5-6, 1997.
- [10] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg, "SSD: Single Shot MultiBox Detector," *arXiv*, 2016.
- [11] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision," *arXiv*, 2017.
- [12] V. H. Tiep, "Blog Machine Learning cơ bản Bài 4: K-means Clustering," 2017. [Trực tuyến]. Available: <https://machinelearningcoban.com/2017/01/01/kmeans/>.

- [13] E. Testlab, "A way to calibrate a magnetometer," 2019. [Online]. Available: <https://teslabs.com/articles/magnetometer-calibration/>.
- [14] Qingde Li, J.G. Griffiths, "Least squares ellipsoid specific fitting," 2004.
- [15] S. O. Madgwick, "Internal Report: An efficient orientation filter for inertial and inertial/magnetic sensor arrays," X-IO Technology, 2010.
- [16] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista, "High-Speed Tracking with Kernelized Correlation Filters," in *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 2014.
- [17] Alan Lukežič, Tomáš Vojíř, Luka Čehovin, Jiří Matas, Matej Kristan, "Discriminative Correlation Filter with Channel and Spatial Reliability," *International Journal of Computer Vision*, 2019.