

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

HUỲNH TRUNG TÍN
PHAN THỊ THANH HIỀN

LUẬN VĂN TỐT NGHIỆP
XE TỰ HÀNH BÁM QUÝ ĐẠO, PHÁT HIỆN
VÀ TRÁNH VẬT CẢN SỬ DỤNG TRÍ TUỆ NHÂN TẠO

KỸ SƯ NGÀNH KỸ THUẬT ĐIỀU KHIỂN & TỰ ĐỘNG HÓA

TP. HỒ CHÍ MINH, 2020

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

HUỲNH TRUNG TÍN - 1613563
PHAN THỊ THANH HIỀN - 1611082

LUẬN VĂN TỐT NGHIỆP
XE TỰ HÀNH BÁM QUÝ ĐẠO, PHÁT HIỆN
VÀ TRÁNH VẬT CẢN SỬ DỤNG TRÍ TUỆ NHÂN TẠO
PATH TRACKING, OBJECT DETECTION AND AVOIDANCE
FOR SELF - DRIVING VEHICLE USING AI

KỸ SƯ NGÀNH KỸ THUẬT ĐIỀU KHIỂN & TỰ ĐỘNG HÓA

GIẢNG VIÊN HƯỚNG DẪN
TS. NGUYỄN VĨNH HẢO

TP. HỒ CHÍ MINH, 2020

TP.HCM, ngày....tháng.....năm.....

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

Tên luận văn:

**XE TỰ HÀNH BÁM QUÝ ĐẠO, PHÁT HIỆN VÀ TRÁNH VẬT CẨN
SỬ DỤNG TRÍ TUỆ NHÂN TẠO**

Nhóm Sinh viên thực hiện:

Huỳnh Trung Tín

1613563 TS. Nguyễn Vĩnh Hảo

Phan Thị Thanh Hiền

1611082 TS. Nguyễn Vĩnh Hảo

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 Trung Tín :...../10

Phan Thị Thanh Hiền :...../10

Cán bộ hướng dẫn

(Ký tên và ghi rõ họ tên)

TP.HCM, ngày....tháng.....năm.....

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

Tên luận văn:

**XE TỰ HÀNH BÁM QUÝ ĐẠO, PHÁT HIỆN VÀ TRÁNH VẬT CẨN
SỬ DỤNG TRÍ TUỆ NHÂN TẠO**

Nhóm Sinh viên thực hiện:

Huỳnh Trung Tín

1613563 TS. Nguyễn Trọng Tài

Phan Thị Thanh Hiền

1611082 TS. Nguyễn Trọng Tài

Đánh giá Luận văn

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 Trung Tín :...../10

Phan Thị Thanh Hiền :...../10

Cán bộ hướng dẫn

(Ký tên và ghi rõ họ tên)

Lời cảm ơn

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến quý thầy cô Trưởng Đại học Bách Khoa – Đại học Quốc gia TP.HCM nói chung, quý thầy cô bộ môn Điều khiển Tự động nói riêng, đã truyền dạy cho chúng em những kiến thức và kinh nghiệm quý báu để hoàn thành đề tài luận văn này.

Chúng em xin gửi lời cảm ơn đặc biệt đến thầy Nguyễn Vĩnh Hảo – Trưởng bộ môn Điều khiển Tự động. Sự giúp đỡ của thầy thực sự rất quan trọng để chúng em có thể hoàn thành đề tài luận văn này. Không những hỗ trợ về kiến thức, thầy còn tiếp thêm tinh thần, giúp chúng em vượt qua những khó khăn trong quá trình thực hiện. Có được sự dẫn dắt của thầy là một may mắn rất lớn trên con đường học tập, đồng thời là một bàn đạp tốt trên con đường phát triển sự nghiệp của chúng em sau này.

Cảm ơn gia đình đã chăm lo, hỗ trợ và tạo động lực cho chúng em trong suốt khoảng thời gian thực hiện đề tài. Cảm ơn gia đình đã nuôi dạy cả về thể chất lẫn tinh thần, đã tin tưởng và đầu tư cho chúng em theo đuổi con đường học tập này, tạo điều kiện cho chúng em có một tương lai tươi sáng phía trước.

Và cuối cùng, chúng em xin gửi lời cảm ơn đến bạn bè đã cùng học tập và làm việc, đặc biệt là các bạn, các anh tại phòng thí nghiệm 207B1 đã hỗ trợ, giúp đỡ chúng em tận tình trong những tháng ngày học tập và thực hiện đề tài luận văn này.

Chúng em xin chân thành cảm ơn!

TP. HCM, ngày 1 tháng 9 năm 2020

Sinh viên

TP. HCM, ngày 1 tháng 9 năm 2020

ĐỀ CƯƠNG CHI TIẾT

TÊN LUẬN VĂN: XE TỰ HÀNH BÁM QUỸ ĐẠO, PHÁT HIỆN VÀ TRÁNH VẬT CẨN SỬ DỤNG TRÍ TUỆ NHÂN TẠO

Cán bộ hướng dẫn: TS. Nguyễn Vĩnh Hảo

Thời gian thực hiện: Từ ngày 30/3/2020 đến ngày 1/9/2020

Sinh viên thực hiện: Huỳnh Trung Tín - 1613563

Phan Thị Thanh Hiền - 1611082

Nội dung đề tài:

Mục tiêu:

Xây dựng mô hình xe tự hành sử dụng GPS để bám quỹ đạo cho trước ứng dụng thuật toán Stanley Controller. Đồng thời, kết hợp thông tin từ stereo camera và trí tuệ nhân tạo để phát hiện và tránh vật cản. Thiết kế và xây dựng giao diện điều khiển để dễ dàng hơn trong quá trình sử dụng.

Phương pháp thực hiện:

- Thiết kế phần cứng, xây dựng mô hình xe tự hành 3 bánh.
- Nghiên cứu, xây dựng các giải thuật điều khiển, giải thuật phát hiện, theo dõi và tránh vật cản, đồng thời phân loại vật cản.
- Xây dựng giao tiếp giữa các module trong mô hình.

Kết quả mong đợi:

- Xây dựng thành công mô hình xe tự hành 3 bánh.
- Thực hiện thành công điều khiển mô hình bám quỹ đạo và tránh vật cản dừng yên.
- Phát hiện và phân loại vật cản: xe máy, ô tô, người, ...
- Lập trình giao diện điều khiển trực quan, dễ sử dụng.

Kế hoạch thực hiện

Huỳnh Trung Tín	Phan Thị Thanh Hiền
<p>Giai đoạn 1: Tiếp nhận dự án, tìm hiểu tổng thể cấu trúc robot</p> <ul style="list-style-type: none"> • Tìm hiểu thông tin và cách sử dụng từng module: H-Bridge, GPS NEO-M8P, Lora. • Tìm hiểu cách kết nối toàn bộ hệ thống phần cứng. • Xây dựng thêm các chi tiết phần cứng mới đồng thời sửa chữa lại các phần cứng đã cũ <p>Giai đoạn 2: Tìm hiểu các thuật toán bám quỹ đạo. Tìm hiểu các hệ tọa độ (Lat, Lng) và UTM, cách chuyển đổi.</p> <ul style="list-style-type: none"> • Tìm hiểu các thuật toán bám quỹ đạo phổ biến hiện nay: Pure Pursuit, Stanley Steering Controller, Model Predictive Control - MPC. • Chọn sử dụng thuật toán Stanley Steering vì: chi phí thấp, dễ thực hiện, độ chính xác ổn định, chấp nhận được. <p>Giai đoạn 3: Lập trình Firmware cho vi điều khiển và viết lại giao diện điều khiển.</p> <ul style="list-style-type: none"> • Xử lý tín hiệu cho từng cảm biến: IMU, GPS, Encoder. • Viết chương trình thực hiện 	<p>Giai đoạn 1: Giao tiếp, cấu hình và hiệu chỉnh camera.</p> <ul style="list-style-type: none"> • Tìm hiểu về BumbleBee2 stereo camera. • Cài đặt driver để giao tiếp với camera trên hệ điều hành ROS. • Tiến hành các bước cấu hình, hiệu chỉnh camera. <p>Giai đoạn 2: Xây dựng giải thuật phát hiện, theo dõi và xác định vị trí 3D của vật cản.</p> <ul style="list-style-type: none"> • Tìm hiểu các thuật toán phát hiện và theo dõi đối tượng. • Đánh giá, lựa chọn thuật toán phát hiện và theo dõi đối tượng phù hợp với mô hình. • Kết hợp các thuật toán đã chọn và thông tin hiệu chỉnh camera để xác định vị trí 3D của đối tượng. <p>Giai đoạn 3: Xây dựng giải thuật điều khiển mô hình tránh vật cản.</p> <ul style="list-style-type: none"> • Tìm hiểu, đánh giá và lựa chọn thuật toán tránh vật cản phù hợp với mô hình. • Tiến hành lập trình thuật toán, chạy mô phỏng và sửa lỗi. <p>Giai đoạn 4: Xây dựng giao tiếp.</p> <ul style="list-style-type: none"> • Xây dựng giao tiếp giữa các thuật

<p>giải thuật điều khiển cho từng khối: Map Planning, Stanley, Fuzzy PI, PID.</p> <ul style="list-style-type: none"> Thiết kế giao diện điều khiển trên máy tính điều khiển và giám sát mô hình từ xa, thông qua module RF LoRa. Xây dựng Frame truyền nhận dữ liệu với giao thức Stop-and-Wait. Thực hiện truyền thông giữa các khối cho hệ thống robot, bao gồm giữa máy tính nhúng và vi điều khiển, vi điều khiển và giao diện. <p>Giai đoạn 4: Thực nghiệm chạy mô hình và tinh chỉnh tham số cho các bộ điều khiển.</p> <ul style="list-style-type: none"> Lựa chọn các bộ số K_p, K_i, K_d cho từng động cơ. Lựa chọn các hệ số K_e, K_{edot} và các hàm thành viên cho bộ điều khiển Fuzzy PI. Thử nghiệm và chọn các hệ số K, K_{soft} cho bộ điều khiển Stanley. <p>Giai đoạn 5: Thực nghiệm chạy ngoài trời với nhiều quỹ đạo cho mô hình và khảo sát các đáp ứng của các bộ điều khiển trên thực tế, phân tích và đánh</p>	<p>toán trên hệ điều hành ROS.</p> <ul style="list-style-type: none"> Xây dựng việc truyền và nhận dữ liệu giữa máy tính nhúng và vi điều khiển STM32F407 thông qua giao tiếp UART. <p>Giai đoạn 5: Quan sát kết quả thực nghiệm, thay đổi các thông số và đánh giá thuật toán.</p> <ul style="list-style-type: none"> Tiến hành chạy thực nghiệm, thu thập số liệu để kiểm tra và đánh giá thuật toán. Điều chỉnh thông số của thuật toán cho phù hợp với mô hình.
--	---

giá dữ liệu thu thập.

- Tiến hành chạy mô hình ngoài thực tế trên nhiều quỹ đạo khác nhau. Mỗi quỹ đạo chạy nhiều vận tốc khác nhau và các cấp vận tốc hỗn hợp.
- Trong quá trình chạy, dữ liệu được thu thập từng giây gửi về cho giao diện và lưu ra file dữ liệu.
- Viết chương trình python xử lý và phân tích dữ liệu thu thập. Vẽ biểu đồ và đánh giá sai số.

Xác nhận của Cán bộ hướng dẫn

(Ký tên và ghi rõ họ tên)

TP. HCM, ngày 12 tháng 7 năm 2020

Sinh viên

(Ký tên và ghi rõ họ tên)

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. - Chủ tịch.
2. - Thư ký.
3. - Ủy viên.
4. - Ủy viên.
5. - Ủy viên.

MỤC LỤC

TÓM TẮT LUẬN VĂN	1
CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI	3
1.1. Giới thiệu xe tự hành.....	3
1.2. Một số ứng dụng thực tiễn	4
1.2.1. Robot giao hàng Starship	4
1.2.2. Robot vận chuyển hàng trong nhà kho Amazon	6
1.3. Mục tiêu của đề tài	7
1.4. Cấu trúc tổng quát của hệ thống.....	8
1.5. Sơ lược nội dung của luận văn.....	9
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT.....	11
2.1. Điều khiển bám quỹ đạo.....	12
2.1.1. Thuật toán Cubic Spline Trajectories	12
2.1.2. Thuật toán bám quỹ đạo Stanley Steering Controller.....	14
2.1.3. Bộ điều khiển PI mờ	16
2.2. Điều khiển tránh vật cản	21
2.2.1. Mô hình MobileNet SSD	21
2.2.2. Thuật toán theo vết đối tượng MOSSE.....	31
2.2.3. Thuật toán Block Matching	33
2.2.4. Thuật toán tránh vật cản	36
CHƯƠNG 3. THIẾT KẾ VÀ THI CÔNG PHẦN CỨNG	45
3.1. Sơ đồ kết nối phần cứng.....	45
3.1.1. Sơ đồ khối công suất.....	45
3.1.2. Sơ đồ kết nối các thành phần.....	46
3.2. Tổng quan các thiết bị	47
3.2.1. Vi xử lý STM32F407	47
3.2.2. Module định vị GPS NEO – M8P	49

3.2.3.	Module cảm biến gia tốc và góc quay IMU.....	51
3.2.4.	Module thu phát sóng RF Lora 433MHz.....	53
3.2.5.	Mạch cầu H.....	54
3.2.6.	Máy tính nhúng.....	55
3.2.7.	Stereo camera Bumblebee2	56
3.3.	Thi công phần cứng	57
3.3.1.	Mô hình robot.....	57
3.3.1.	Trạm Base GPS.....	58

CHƯƠNG 4. THIẾT KẾ HỆ THỐNG..... 59

4.1.	Giải thuật điều khiển	60
4.1.1.	Khối Referenced Planning Map	60
4.1.2.	Khối Stanley Steering Controller	61
4.1.3.	Khối Fuzzy Controller	64
4.1.4.	Khối PID Motor	69
4.2.	Hệ thống ROS node và giải thuật tránh vật cản.....	71
4.2.1.	Camera1394stereo Node	72
4.2.2.	Stereo_image_proc Node	72
4.2.3.	Object Detection Node.....	75
4.2.4.	Object Avoidance Node	79
4.2.5.	SerialPort node	83
4.3.	Giao diện điều khiển trên máy tính	85
4.3.1.	Giao diện cấu hình thông số tổng quát.....	85
4.3.2.	Giao diện cấu hình IMU	86
4.3.3.	Giao diện điều khiển góc (Fuzzy test)	87
4.3.4.	Giao diện điều khiển path tracking.....	87

CHƯƠNG 5. KẾT QUẢ THỰC NGHIỆM..... 91

5.1.	Kết quả phát hiện, thep dõi và xác định tọa độ của vật cản	91
5.1.1.	Kết quả phát hiện vật	91
5.1.2.	Kết quả theo dõi vật.....	93

5.1.3. Kết quả xác định tọa độ của vật.....	95
5.1.4. Tốc độ xử lý	97
5.2. Các kết quả điều khiển	98
5.2.1. Kết quả lọc nhiễu và xử lý Encoder	98
5.2.2. Kết quả điều khiển vận tốc PID.....	99
5.2.3. Kết quả điều khiển góc Fuzzy.....	101
5.2.4. Kết quả điều khiển bám quỹ đạo	102
5.2.5. Kết quả điều khiển mô hình tránh vật cản	119
CHƯƠNG 6. ĐÁNH GIÁ, KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	121
6.1. Đánh giá kết quả.....	121
6.2. Kết luận	123
6.3. Hướng phát triển	123
TÀI LIỆU THAM KHẢO	124

DANH MỤC HÌNH ẢNH

Hình 1.1 Mô hình xe tự hành [1].....	4
Hình 1.2 Robot giao hàng của công ty Startship Technologies [2].....	5
Hình 1.3 Robot đang lấy hàng trong các nhà kho của Amazon [3].....	6
Hình 1.4 Cấu trúc tổng quát hệ thống	8
Hình 2.1 Sơ đồ các thuật toán sử dụng trong mô hình	11
Hình 2.2 Quỹ đạo tuyến tính và đáp ứng góc	12
Hình 2.3 Path planning sử dụng thuật toán Cubic Spline Trajectories	13
Hình 2.4 Quỹ đạo và đáp ứng góc sau khi sử dụng Cubic Spline	14
Hình 2.5 Mô hình sử dụng trong thuật toán Staley Controller.....	15
Hình 2.6 Sơ đồ khối bộ điều khiển PID	16
Hình 2.7 Sơ đồ khối cấu trúc bộ điều khiển Fuzzy	19
Hình 2.8 Sơ đồ khối thiết kế bộ điều khiển PI mờ	20
Hình 2.9 Phương pháp giải mờ trọng tâm.....	21
Hình 2.10 Cấu trúc mạng MobileNet.....	22
Hình 2.11 Depthwise Separable Convolution.....	23
Hình 2.12 Depthwise convolution và Pointwise convolution	24
Hình 2.13 So sánh MobileNet với các mô hình khác.....	26
Hình 2.14 Cấu trúc mạng SSD	26
Hình 2.15 SSD sử dụng nhiều bounding box.....	27
Hình 2.16 So sánh giữa các mạng phát hiện đối tượng	30
Hình 2.17 Nguyên lý hoạt động stereo camera.....	33
Hình 2.18 Đường epipolar	35
Hình 2.19 Nguyên lý cơ bản của thuật toán Block Matching	36
Hình 2.20 Chuyển động của mô hình với các góc lái khác nhau.....	37
Hình 2.21 Hệ tọa độ của mô hình	37
Hình 2.22 Cách xác định $d(\theta, obs)$	39
Hình 2.23 Cách xác định θ_{min} và θ_{max}	40
Hình 2.24 Cách chọn $d\theta, obs$ xấp xỉ cho từng khoảng	41
Hình 2.25 Một số kết quả mô phỏng thuật toán tránh vật cản.....	44

Hình 3.1 Sơ đồ kết nối các khối nguồn, cung cấp công suất cho hệ robot.....	45
Hình 3.2 Components Connection Diagram	46
Hình 3.3 Board STM32F407 Discovery	47
Hình 3.4 Mô hình RTK GPS NEO – M8P	49
Hình 3.5 GPS Antenna	50
Hình 3.6 IMU sử dụng cảm biến MEMS ADIS16488	51
Hình 3.7 Module truyền nhận không dây RF LoRa 433MHz.....	53
Hình 3.8 Driver điều khiển động cơ sử dụng mạch cầu H HI216	54
Hình 3.9 Nano – HR650 – R11	55
Hình 3.10 BumbleBee2 stereo camera	56
Hình 3.11 Mô hình robot.....	58
Hình 3.12 Mô hình trạm Base GPS.....	58
Hình 4.1 Sơ đồ khối giải thuật điều khiển cho hệ robot.....	59
Hình 4.2 Dữ liệu mẫu thuật toán Cubic Spline	61
Hình 4.3 Giải thuật tính góc lái Stanley.....	62
Hình 4.4 Giải thuật cho bộ điều khiển Fuzzy.....	64
Hình 4.5 Tập mờ cho biến ngõ vào e và e	66
Hình 4.6 Tập mờ cho biến ngõ ra u	67
Hình 4.7 Sơ đồ khối thiết kế bộ điều khiển PID	69
Hình 4.8 Sơ đồ các node	71
Hình 4.9 Ảnh camera trước khi hiệu chỉnh.....	72
Hình 4.10 Stereo_image_proc node	73
Hình 4.11 Ảnh camera sau khi hiệu chỉnh.....	75
Hình 4.12 Giải thuật phát hiện, xác định vị trí và tốc độ vật cản.....	76
Hình 4.13 Cách tính IoU	78
Hình 4.14 Ví dụ phát hiện vật cản và bản đồ chênh lệch	78
Hình 4.15 Giải thuật tránh vật cản.....	80
Hình 4.16 Sơ đồ truyền thông Serial Node	84
Hình 4.17 Giao diện cấu hình thông số	85
Hình 4.18 Giao diện cấu hình và calib cảm biến IMU	86

Hình 4.19 Giao diện điều khiển Manual dựa trên góc của cảm biến IMU	87
Hình 4.20 Giao diện chế độ Auto	87
Hình 4.21 Các tính năng chính ở chế độ Auto.....	88
Hình 4.22 Nút SETTING	89
Hình 4.23 Nút PROCESSING MAP.....	90
Hình 4.24 Nút TRANSFER MAP.....	90
Hình 5.1 Ví dụ hình ảnh bị lóa sáng.....	92
Hình 5.2 Ví dụ bounding box không bao hết vật	93
Hình 5.3 Kết quả theo dõi vật đứng yên.....	93
Hình 5.4 Một số kết quả theo dõi vật chuyển động	94
Hình 5.5 Tín hiệu đọc về từ Encoder không tải, PWM=20%.....	98
Hình 5.6 Tín hiệu đọc về từ Encoder có tải, PWM=20%.....	99
Hình 5.7 Đáp ứng điều khiển vận tốc M1 không tải, Kp=0.66 Ki=1.3 Kd=0.0005	99
Hình 5.8 Đáp ứng điều khiển vận tốc M2 không tải, Kp=0.5 Ki=1.35 Kd=0.0002 ..	100
Hình 5.9 Đáp ứng điều khiển vận tốc M1 có tải, Kp=0.66 Ki=1.3 Kd=0.0005	100
Hình 5.10 Đáp ứng điều khiển vận tốc M2 có tải, Kp=0.5 Ki=1.35 Kd=0.0002.....	100
Hình 5.11 Đáp ứng với $\Delta\varphi = +60^\circ$	102
Hình 5.12 Đáp ứng thực tế quỹ đạo 1, $V = 0.2$ (m/s)	103
Hình 5.13 Mô phỏng quỹ đạo 1 trên hệ tọa độ UTM.....	103
Hình 5.14 Góc tiếp tuyến của quỹ đạo so với trục x trên hệ tọa độ UTM	104
Hình 5.15 Đồ thị thể hiện đặc tính cong của quỹ đạo.....	104
Hình 5.16 Đồ thị efa thể hiện sai lệch vị trí robot so với quỹ đạo, $V = 0.2$ (m/s)	105
Hình 5.17 Đồ thị đáp ứng vận tốc, $V = 0.2$ (m/s)	105
Hình 5.18 Đáp ứng góc và quỹ đạo từ A đến B, $V = 0.2$ m/s.....	106
Hình 5.19 Đồ thị sai số quỹ đạo của robot, $V = 0.3$ (m/s)	107
Hình 5.20 Đồ thị đáp ứng vận tốc, $V = 0.3$ (m/s)	107
Hình 5.21 Đáp ứng góc và quỹ đạo từ A đến B, $V = 0.3$ m/s.....	108
Hình 5.22 Đồ thị sai số quỹ đạo của robot, $V = 0.5$ (m/s)	109
Hình 5.23 Đồ thị đáp ứng vận tốc, $V = 0.5$ (m/s)	109
Hình 5.24 Đáp ứng góc và quỹ đạo từ A đến B, $V = 0.5$ m/s.....	110

Hình 5.25 Đồ thị sai số quỹ đạo của robot, $V = 0.7$ (m/s)	111
Hình 5.26 Đồ thị đáp ứng vận tốc, $V = 0.7$ (m/s)	111
Hình 5.27 Đáp ứng góc và quỹ đạo từ A đến B, $V = 0.7m/s$	112
Hình 5.28 Đồ thị sai số quỹ đạo của robot, $V = 1.25$ (m/s)	113
Hình 5.29 Đồ thị đáp ứng vận tốc, $V = 1.25$ (m/s)	113
Hình 5.30 Đáp ứng góc và quỹ đạo từ A đến B, $V = 1.25m/s$	114
Hình 5.31 Đồ thị sai số quỹ đạo của robot, $V = [0.3, 0.5, 0.7]$ (m/s)	115
Hình 5.32 Đồ thị đáp ứng vận tốc, $V = [0.3, 0.5, 0.7]$ (m/s)	116
Hình 5.33 Đáp ứng góc và quỹ đạo từ A đến B, $V = [0.3, 0.5, 0.7]$ m/s	116
Hình 5.34 Đồ thị sai số quỹ đạo của robot, $V = [0.3, 0.7, 0.3, 0.8]$ (m/s)	117
Hình 5.35 Đồ thị đáp ứng vận tốc, $V = [0.3, 0.7, 0.3, 0.8]$ (m/s)	118
Hình 5.36 Đáp ứng góc và quỹ đạo từ A đến B, $V = [0.3, 0.7, 0.3, 0.8]$ m/s	118
Hình 5.37 Một số kết quả điều khiển tránh vật cản	120

DANH MỤC BẢNG

Bảng 2.1 Ảnh hưởng của các hệ số lén bộ điều khiển PID.....	18
Bảng 2.2 Các lớp của tập dữ liệu VOC0712	31
Bảng 3.1 Thông số kỹ thuật vi xử lí STM32F407	48
Bảng 3.2 Thông số kỹ thuật của Module GPS	49
Bảng 3.3 Thông số kỹ thuật của GPS Antenna	50
Bảng 3.4 Thông số kỹ thuật của Module IMU	51
Bảng 3.5 Frame truyền dữ liệu của IMU	52
Bảng 3.6 Thông số kỹ thuật Module thu phát RF LoRa	53
Bảng 3.7 Thông số kỹ thuật Mạch cầu H HI216	54
Bảng 3.8 Thông số kỹ thuật của máy tính nhúng	55
Bảng 3.9 Thông số kỹ thuật của BumbleBee2 stereo camera	57
Bảng 4.1 Các quy tắc mờ	68
Bảng 4.2 Các lớp đối tượng sử dụng trong đề tài.....	77
Bảng 5.1 Kết quả phát hiện đối tượng	91
Bảng 5.2 Kết quả xác định vị trí vật cản	96
Bảng 5.3 Tốc độ xử lý của thuật toán phát hiện, theo dõi vật cản	97
Bảng 5.4 Kết quả điều khiển bám quỹ đạo với $V = 0.2 \text{ m/s}$	106
Bảng 5.5 Kết quả điều khiển bám quỹ đạo với $V = 0.3 \text{ m/s}$	108
Bảng 5.6 Kết quả điều khiển bám quỹ đạo với $V = 0.5 \text{ m/s}$	110
Bảng 5.7 Kết quả điều khiển bám quỹ đạo với $V = 0.7 \text{ m/s}$	112
Bảng 5.8 Kết quả điều khiển bám quỹ đạo với $V = 1.25 \text{ m/s}$	114
Bảng 5.9 Kết quả điều khiển bám quỹ đạo với vận tốc cố định	115
Bảng 5.10 Kết quả điều khiển bám quỹ đạo, $V = [0.3, 0.5, 0.7] \text{ (m/s)}$	117
Bảng 5.11 Kết quả điều khiển bám quỹ đạo, $V = [0.3, 0.7, 0.3, 0.8] \text{ (m/s)}$	119

DANH MỤC TỪ VIẾT TẮT

Từ khóa	Tên tiếng anh
AI	Artificial Intelligence
CNN	Convolution Neural Network
FPS	Frame Per Second
mAP	mean Average Precision
IoU	Intersection over Union
ReLU	Rectified Linear Units
ROS	Robot Operating System
SAD	Sum of Absolute Difference
RMS	Root Mean Square
YOLO	You Only Look Once
R – CNN	Region – based Convolution Neural Network
UTM	Universal Transverse Mercator
DMS	Degree Minute Second
DD	Decimal Degree

TÓM TẮT LUẬN VĂN

Luận văn trình bày quá trình xây dựng và lập trình mô hình xe tự hành 3 bánh bám quỹ đạo và tránh vật cản. Chúng tôi sử dụng module GPS RTK để xác định vị trí, vi xử lý STM32F407 để điều khiển mô hình sử dụng thuật toán Stanley Controller, kết hợp module LoRa để giao tiếp giữa trạm Base và Rover cũng như giữa mô hình và máy tính. Bộ điều khiển PID và PI mờ được sử dụng để đạt kết quả điều khiển tốt nhất.

Trong quá trình bám quỹ đạo, mô hình sử dụng thông tin từ BumbleBee2 stereo camera để phát hiện các vật cản phía trước (bao gồm người và các phương tiện), theo dõi và xác định vị trí 3D của vật cản. Dựa vào đó, mô hình được điều khiển để tránh vật cản khi cần thiết. Bộ theo dõi MOSSE của thư viện OpenCV và mô hình phát hiện đối tượng MobileNet SSD được sử dụng. Thuật toán tránh vật cản được xây dựng dựa trên phương pháp The Curvature – Velocity Method. Các thuật toán này được xây dựng trên nền tảng hệ điều hành ROS.

Luận văn đã xây dựng thành công mô hình xe tự hành 3 bánh di chuyển với vận tốc tối đa 0.9m/s, sai số bám quỹ đạo từ 4 – 15cm và có khả năng tránh các vật cản khi cần thiết. Để thuận tiện hơn trong quá trình điều khiển mô hình, giao diện người dùng trực quan và dễ sử dụng trên máy tính được xây dựng.

ABSTRACT

This thesis illustrates the process of building and programming a 3-wheel autonomous vehicle model to follow a desired trajectory and avoid objects. We use GPS RTK module to determine the model's position and STM32F407 microprocessor to control the model. Besides, LoRa modules are used to communicate between the GPS Base and Rover stations and between the model and the computer. The main path tracking algorithm is Stanley Controller whereas the PID and PI Fuzzy controller is used to achieve the best control results.

During the path tracking process, the model uses information from the BumbleBee2 stereo camera to detect objects ahead (including people and vehicles), track and determine the 3D position of them. Then the model is controlled to avoid obstacles as needed. The OpenCV library's MOSSE tracker and the MobileNet SSD object detection model are used. The obstacle avoidance algorithm is built on the basis of The Curvature - Velocity Method. These algorithms are implemented on ROS.

We have successfully built a 3-wheel autonomous vehicle model that moves with a maximum speed of 0.9m/s, a path tracking error of 4 – 15 centimeters and is able to avoid obstacles when necessary. For more convenience during model control, we have built an intuitive and easy-to-use computer user interface.

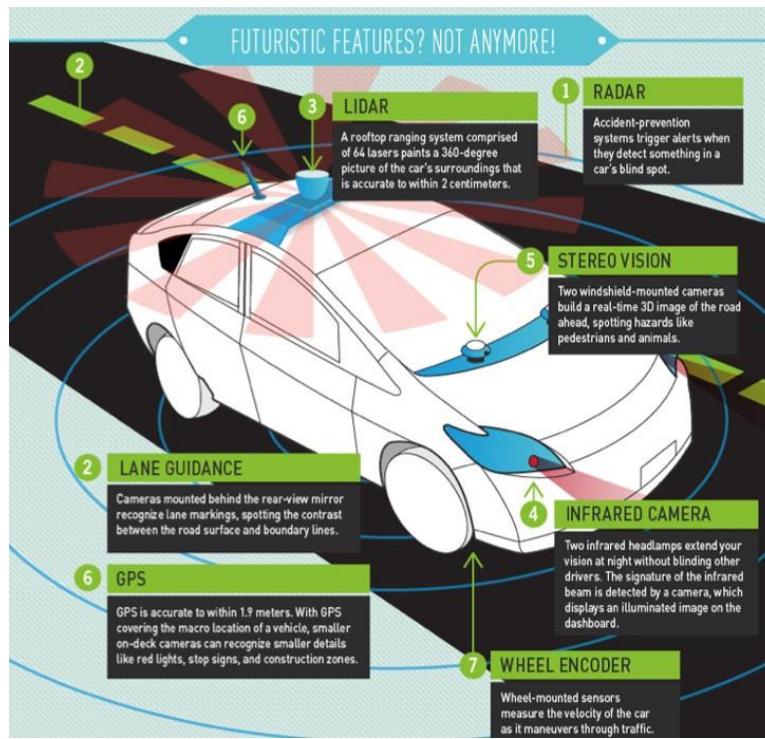
Chương 1. TỔNG QUAN VỀ ĐỀ TÀI

1.1. Giới thiệu xe tự hành

Hiện nay, đã có rất nhiều tập đoàn sản xuất xe hơi và công nghệ lớn trên thế giới đã tham gia vào cuộc chạy đua phát triển xe hơi công nghệ tự lái thông minh (gọi tắt là xe tự lái, xe tự hành) mà không cần đến bất cứ can thiệp nào từ bàn tay của con người, trong đó có những tên tuổi nổi bật như Tesla, Daimler, Google, ...

Theo Sridhar Lakshmanan, một chuyên gia về xe hơi tự lái, giáo sư về kỹ thuật của trường Đại học Michigan-Dearborn, một chiếc ô tô chỉ có thể tự lái được nếu nó đáp ứng được các tiêu chuẩn sau:

- Có hình dạng giống những chiếc ô tô thông thường.
- Được trang bị hệ thống nhận diện các biến động trên đường: GPS sẽ xác định nhiệm vụ của xe tự lái bằng các thiết lập điểm đầu và điểm cuối của hành trình dựa trên tính năng dẫn đường của Google Maps. Một hệ thống công nghệ hỗ trợ khác như radar, laser, camera để phát hiện và xử lý các tình huống bất ngờ. Trong đó, camera được dùng để hệ thống máy tính bên trong xe có thể nhìn thấy tình trạng xung quanh xe, còn radar sẽ giúp ô tô nhìn được đường phía trước (khoảng cách 100m), laser được dùng để quét các hiện tượng xảy ra xung quanh liên tục và gửi đến hệ thống máy tính.
- Có hệ thống chuyển các thông tin từ hai hệ thống trên thành hành động thực tế trên đường.



Hình 1.1 Mô hình xe tự hành [1]

1.2. Một số ứng dụng thực tiễn

1.2.1. Robot giao hàng Starship

Dịch vụ robot giao hàng của công ty Starship Technologies đã “nở rộ” tại hàng chục thành phố trên thế giới nhằm phục vụ người tiêu dùng ở nhà do đại dịch viêm đường hô hấp cấp COVID-19, giúp thiểu rủi ro lây nhiễm dịch bệnh cho cả người mua hàng và nhân viên giao hàng.



Hình 1.2 Robot giao hàng của công ty Startship Technologies [2]

Robot tự hành với thiết kế 6 bánh xe, vận tốc di chuyển 6 km/h, có thể vận chuyển cùng lúc tối đa 3 túi hàng, có thể tự điều chỉnh hướng di chuyển trên đường phố cũng như vỉa hè.

Robot Starship đi vào hoạt động chính thức kể từ đầu tháng 4 này tại cửa hàng rượu Broad Branch, khi cửa hàng ở một góc phố ngoại ô thủ đô Washington của Mỹ này buộc phải đóng cửa do diện tích quá nhỏ, không đủ điều kiện đảm bảo quy định giãn cách xã hội để ngăn chặn sự lây lan của dịch COVID-19.

Chủ cửa hàng Broad Branch, Tracy Stannard, cho biết 10 robot do Starship Technologies quản lý giúp vận chuyển hàng theo đặt hàng của người dân trong khu phố. Các robot "trợ thủ" này đảm trách giao hàng 50% trong số 60 - 70 đơn hàng mỗi ngày của Broad Branch.

Ngoài Starship Technologies, một số ít công ty khác cũng đã từng bước thâm nhập thị trường giao hàng độc đáo nói trên. Công ty khởi nghiệp tại Thung lũng Silicon, Nuro, gần đây đã hợp tác với Tập đoàn bán lẻ Kroger bắt đầu cung cấp dịch vụ giao hàng ở khu vực Houston bằng robot tự hành R2 do Nuro sản xuất.

Robot của Nuro có thể di chuyển với vận tốc lên tới 40 km/giờ và có khả năng chuyên chở khoảng 190 kg hàng hóa. Nuro đang mở rộng dịch vụ giao hàng trên và đã được chính quyền bang California cho phép robot của hãng hoạt động tại các tuyến đường công cộng. Robot giao hàng của công ty khởi nghiệp Postmate cũng đã xuất hiện trên nhiều con phố ở California. Trong khi đó, Tập đoàn bán lẻ Amazon đang thử nghiệm hoạt động giao hàng với các robot tự hành tương tự.

1.2.2. Robot vận chuyển hàng trong nhà kho Amazon

Một năm trước, một người lao động tại Amazon.com như Rejinaldo Rosales (34 tuổi) phải đi bộ hàng dặm mỗi ca để lấy từng mặt hàng và chuẩn bị cho việc vận chuyển. Nay giờ, gã khổng lồ trong làng thương mại điện tử có thể tự hào rằng, họ đã đẩy mạnh hiệu quả và giúp cho đôi chân của người lao động được nghỉ ngơi, bằng cách triển khai hơn 15.000 robot ở khắp các tầng của nhà kho rộng lớn.

Hiện có hơn 1.500 nhân viên toàn thời gian làm việc tại trung tâm Tracy rộng 1,2 triệu feet (hơn 365Km) vuông - tương đương với 28 sân bóng đá. Đồng thời, tại đây còn có khoảng 3.000 robot có thể lướt đi nhanh chóng quanh nhà kho, các robot này điều hướng bằng cách quét mã dán trên sàn nhà. Hệ thống sử dụng mã vạch để theo dõi hàng hóa trên mỗi kệ, do đó, robot có thể lấy đúng mặt hàng mang tới cho công nhân.



Hình 1.3 Robot đang lấy hàng trong các nhà kho của Amazon [3]

Mỗi "công nhân robot" Kiva nặng khoảng 145 kg, chạy dưới sàn nhà kho và có nhiệm vụ di chuyển các kệ hàng (nặng khoảng 340 kg và cao hơn 1 người trưởng thành) một cách trình tự, khoa học và chính xác cho cảm tưởng như đang lạc vào một nhà máy trong phim khoa học viễn tưởng vậy. Con người chỉ cần chất các món đồ lên kệ hàng và việc còn lại sẽ do robot Kiva đảm nhiệm. Tất cả đã tạo nên một khung cảnh hết sức nhộn nhịp và sống động. Amazon cho biết rằng hiện tại, hơn 15.000 robot Kiva đã được sử dụng cho tất cả các trụ sở và nhà kho của hãng trên khắp nước Mỹ. Bình quân mỗi công nhân Kiva sẽ vận chuyển khoảng 700.000 món hàng mỗi ngày và lúc cao điểm, con số này có thể lên tới 1,5 triệu món.

Theo Amazon, việc sử dụng robot thay cho con người để di chuyển hàng trong kho sẽ giúp tiết kiệm được diện tích do không cần phải làm lối đi cho người, từ đó sẽ có thể chứa được lượng kệ hàng nhiều hơn. Đồng thời, việc dùng robot còn giúp giảm thiểu thời gian chết và thực hiện quá trình luân chuyển hàng hóa hiệu quả hơn. Dù đã sử dụng rất nhiều hệ thống robot hiện đại nhưng Amazon cho biết họ vẫn phải dùng nguồn nhân lực rất lớn. Trong năm 2014 này, hãng đã lên kế hoạch tuyển thêm 80.000 nhân viên để trang trải cho các mùa cao điểm mua sắm vốn được dự đoán là còn cao hơn 14% so với năm ngoái.

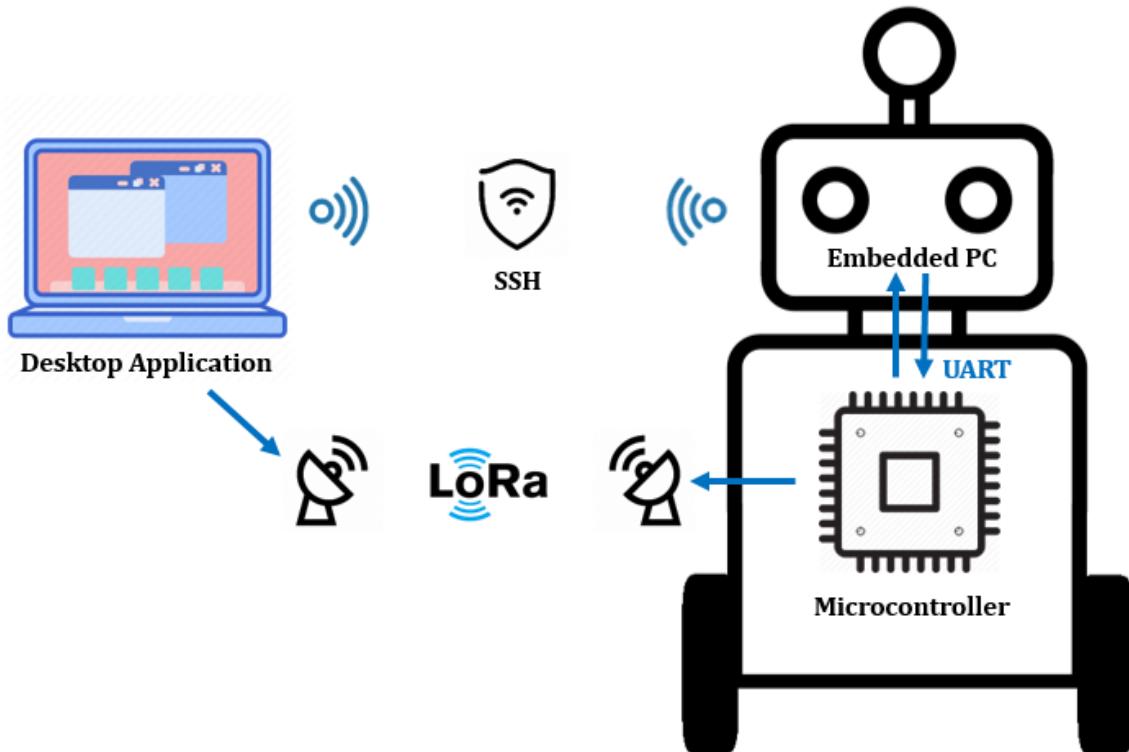
Một đội quân 10.000 robot mạnh mẽ có thể giúp Amazon tiết kiệm 450 triệu USD đến 900 triệu USD một năm trong chi phí lao động, Shawn Milne của Janney Capital Markets ước tính trong báo cáo mùa hè năm ngoái.

1.3. Mục tiêu của đề tài

- Xây dựng mô hình xe tự hành chạy ngoài trời, sử dụng GPS để bám các quỹ đạo cho trước.
- Phát hiện vật cản và ra quyết định ra khỏi quỹ đạo để tránh khi cần thiết với thông tin từ stereo camera.

- Tối ưu tốc độ xử lý, giảm thiểu sai số bám quỹ đạo đồng thời đảm bảo độ an toàn cho mô hình khi tránh các vật cản.
- Xây dựng giao diện người dùng trực quan, dễ sử dụng.

1.4. Cấu trúc tổng quát của hệ thống



Hình 1.4 Cấu trúc tổng quát hệ thống

Cấu trúc hệ thống gồm 3 phần:

- **Desktop Application:** là một ứng dụng người dùng Windows Forms App được xây dựng bằng C# với framework dotNET và framework Bunifu (hỗ trợ thiết kế UI). Giao diện dùng để hỗ trợ quá trình điều khiển và giám sát robot trong lúc vận hành bằng hệ thống các lệnh nhóm tự định nghĩa, dữ liệu và lệnh được trao đổi với Microcontroller thông qua một module Lora RF được kết nối với USB UART để giao tiếp với máy tính qua Serial Port. Ngoài ra, để truy cập máy tính nhúng từ xa trong quá trình vận hành,

Laptop sử dụng giao thức SSH để kết nối với máy tính nhúng để chạy và giám sát các Node ROS.

- **Microcontroller:** sử dụng vi điều khiển STM32F407VGTX với ngôn ngữ lập trình chính là C, là trung tâm điều khiển chính của robot, nơi thu thập xử lý dữ liệu từ cảm biến (IMU, GPS) cũng như thi hành các thuật toán điều khiển chính như Stanley, Pure Pursuit. Truyền thông giao tiếp với Desktop App thông qua Lora RF, và giao tiếp với Serial Port của máy tính nhúng thông qua USB UART. Phần giao tiếp với máy tính nhúng bao gồm: dữ liệu của IMU được trả về từ máy tính nhúng, ngoài ra vi xử lý còn cập nhật vị trí và đường đi cho máy tính nhúng để thực hiện các thuật toán tránh vật cản và nhận về các lệnh điều khiển né kèm <vận tốc, góc lái> nếu camera phát hiện được vật cản.
- **Embedded PC:** Máy tính nhúng với mainboard thuộc họ PC/104 được kết nối với IMU ADIS16488 và Stereo Camera Bumblebee2, nơi thực hiện các giải thuật xử lý ảnh và tránh vật cản, truyền thông nối tiếp với vi điều khiển thông qua Serial Port. Các chương trình được viết trên nền tảng ROS sử dụng các ngôn ngữ C++ và Python. Về phần truyền thông, máy tính nhúng chuyển dữ liệu từ IMU xuống cho vi điều khiển và gửi lệnh cấu hình từ vi điều khiển lên IMU, ngoài ra trong lúc chạy bám quỹ đạo, robot liên tục cập nhật <vị trí, góc heading, điểm tham chiếu trên map> cho máy tính nhúng input cho các thuật toán tránh vật cản sử dụng để ra quyết định né vật cản, nếu phát hiện được vật cản, thuật toán né output ra <vận tốc, góc lái> cho vi điều khiển.

1.5. Sơ lược nội dung của luận văn

Luận văn bao gồm 6 chương:

Chương 1: Tổng quan về đề tài, cung cấp những thông tin cơ bản về xe tự hành và các ứng dụng thực tiễn của chúng. Giới thiệu mục tiêu, nhiệm vụ và cấu trúc tổng quan của đề tài.

Chương 2: Trình bày cơ sở lý thuyết của các bộ điều khiển, các thuật toán sử dụng trong đề tài.

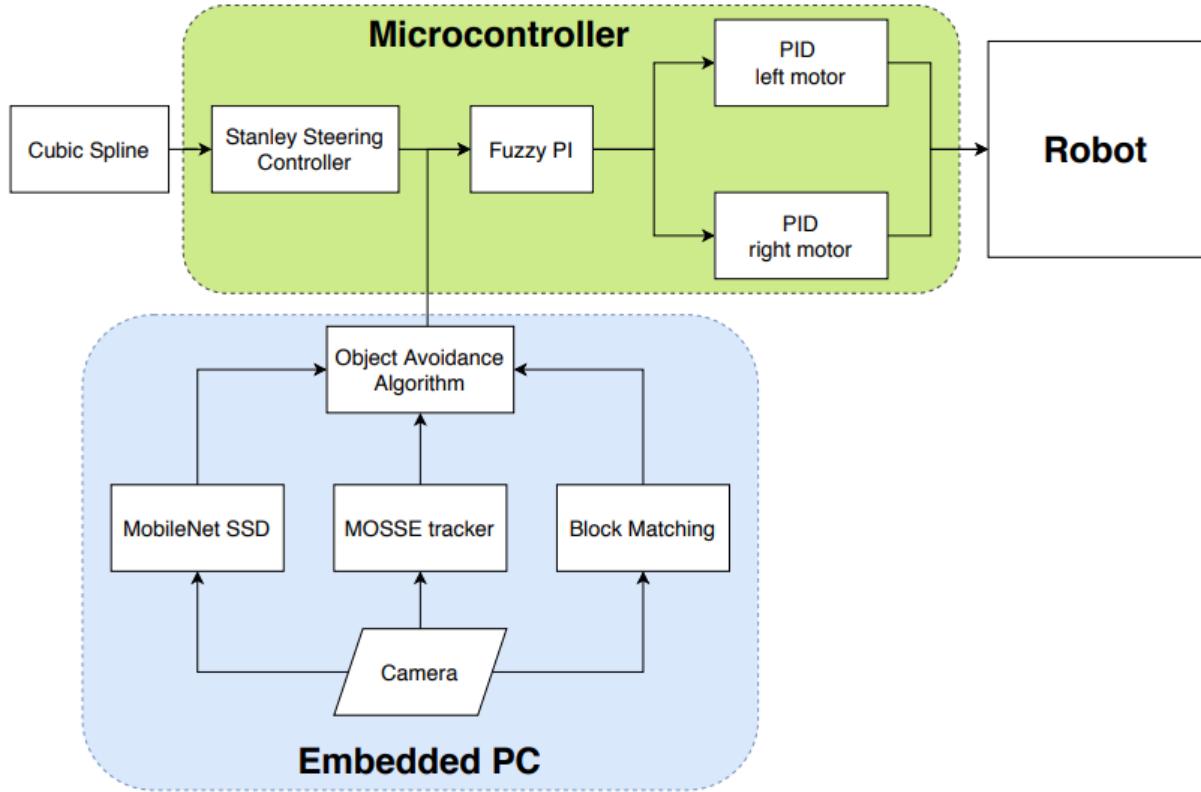
Chương 3: Giới thiệu các thiết bị sử dụng trong đề tài. Trình bày chi tiết về cấu trúc, chức năng và nhiệm vụ của từng khối trong phần cứng.

Chương 4: Trình bày các giải thuật điều khiển và phát hiện vật cản. Tóm tắt cấu trúc và ứng dụng của giao diện người dùng.

Chương 5: Trình bày kết quả thực nghiệm và ảnh hưởng của các thông số đến kết quả.

Chương 6: Đánh giá kết quả, kết luận và phương hướng phát triển cho đề tài.

Chương 2. CƠ SỞ LÝ THUYẾT



Hình 2.1 Sơ đồ các thuật toán sử dụng trong mô hình

Hoạt động chính của robot gồm 2 phần:

- **Điều khiển bám quỹ đạo:** Ứng dụng thuật toán Cubic Spline để tối ưu quỹ đạo trước khi vận hành và thuật toán bám quỹ đạo Stanley Steering Controller. Sử dụng các bộ điều khiển PI mờ điều khiển góc, PID điều khiển vận tốc động cơ.
- **Điều khiển tránh vật cản:** Ứng dụng mô hình MobileNet SSD, bộ theo vết MOSSE của thư viện OpenCV để phát hiện, theo dõi và xác định vị trí của vật cản. Điều khiển mô hình tránh vật cản khi cần thiết bằng thuật toán Curvature – Velocity Method.

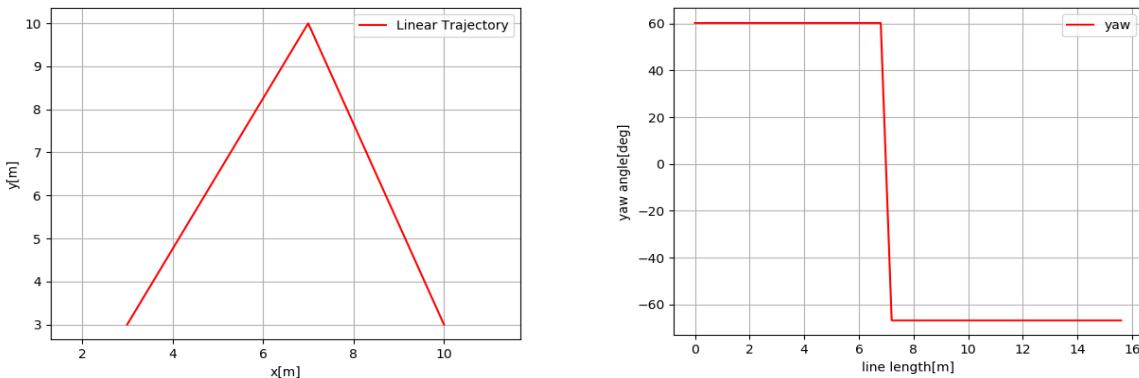
2.1. Điều khiển bám quỹ đạo

2.1.1. Thuật toán Cubic Spline Trajectories

Giả sử ta có 3 điểm trên quỹ đạo, tương ứng với các điểm $M_1(3, 3)$, $M_2(7, 10)$ và $M_3(10, 3)$, hệ phương trình bậc 1 của quỹ đạo có dạng:

$$\begin{cases} y = 1.052x - 1.166, & 3 \leq x < 7 \\ y = -0.155x + 18.161, & 7 \leq x \leq 10 \end{cases}$$

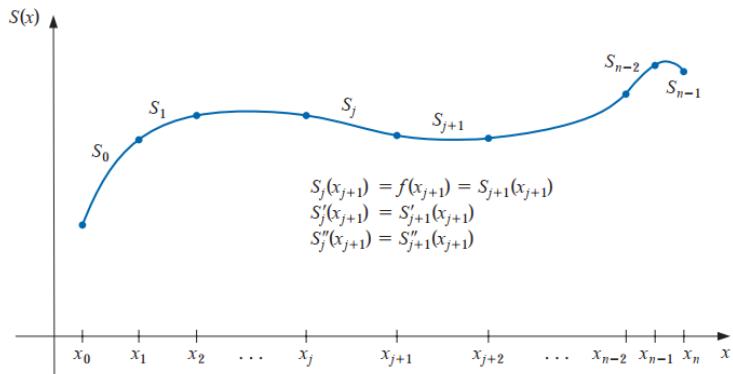
Từ phương trình, ta sẽ có quỹ đạo và góc tiếp tuyến như hình:



Hình 2.2 Quỹ đạo tuyến tính và đáp ứng góc

Góc của tiếp tuyến quỹ đạo có giá trị thay đổi đột ngột khi chuyển làn, dẫn tới đáp ứng của robot có thể không kịp thời, gây ra một số hiện tượng vọt lố và làm tăng sai số so với quỹ đạo. Do đó cần một quỹ đạo có góc thay đổi một cách liên tục.

Thuật toán Cubic Spline Trajectories cho ra quỹ đạo bao gồm các phân đoạn là đa thức bậc ba dựa trên các điểm tọa độ cố định. Mỗi đoạn kết nối 2 điểm cố định của quỹ đạo, 2 bậc tự do còn lại được tính dựa trên vector tiếp tuyến của quỹ đạo. Do đó quỹ đạo sẽ có độ thay đổi của vector tiếp tuyến và độ cong một cách liên tục giúp robot di chuyển mượt hơn và ít bị vọt lố hơn so với quỹ đạo thông thường.



Hình 2.3 Path planning sử dụng thuật toán Cubic Spline Trajectories

Cho một phương trình f được định nghĩa trong khoảng $[a, b]$ sao cho các điểm $a = x_0 < x_1 < \dots < x_n = b$. Phương trình f là *Cubic spline* khi thỏa các điều kiện sau:

- $S(x)$ là đa thức bậc 3, ký hiệu là $S_j(x)$, trong phân đoạn $[x_j, x_{j+1}]$, với mỗi $j = 0, 1, \dots, n - 1$;
- $S_j(x_j) = f(x_j)$ và $S_j(x_{j+1}) = f(x_{j+1})$ với mỗi $j = 0, 1, \dots, n - 1$;
- $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ với mỗi $j = 0, 1, \dots, n - 2$;
- $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$ với mỗi $j = 0, 1, \dots, n - 2$;
- $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$ với mỗi $j = 0, 1, \dots, n - 2$;
- Một trong hai điều kiện sau được thỏa mãn:

$$S''(x_0) = S''(x_n) = 0$$

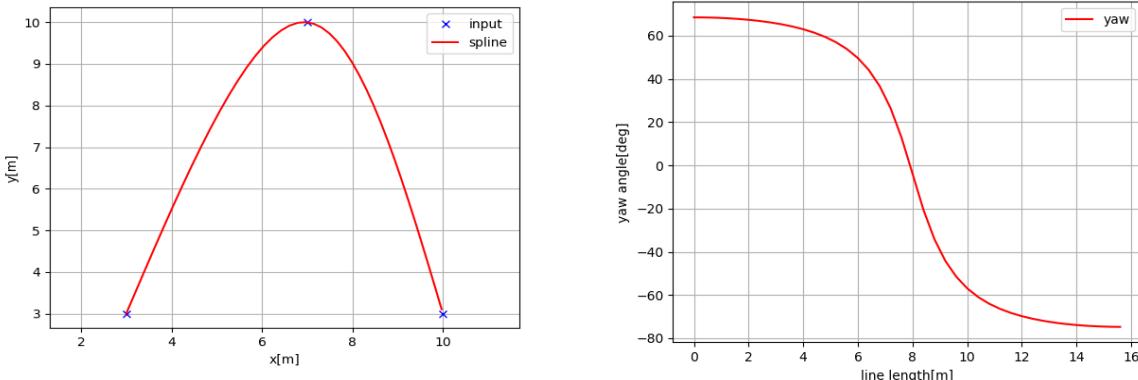
$$S'(x_0) = f'(x_0) \text{ và } S'(x_n) = f'(x_n)$$

Phương trình cơ bản của *Cubic Spline*:

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

Với: $j = 0, 1, \dots, n - 1$

Áp dụng cho ví dụ trên, ta được kết quả quỹ đạo và góc điều khiển cho Robot như hình:



Hình 2.4 Quỹ đạo và đáp ứng góc sau khi sử dụng Cubic Spline

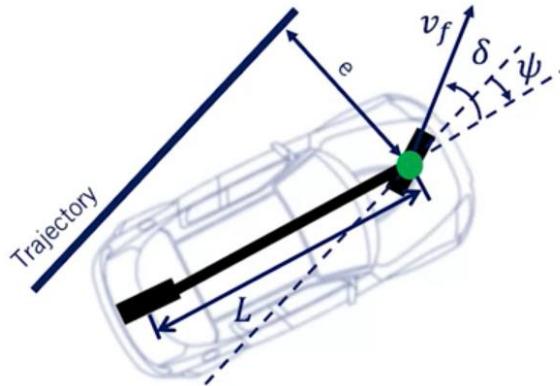
Dựa trên kết quả, có thể thấy góc tiếp tuyến của quỹ đạo thay đổi một cách liên tục, tránh thay đổi một cách đột ngột, dẫn đến góc điều khiển của robot không bị thay đổi quá nhanh, giúp tránh được các trường hợp vọt lố hoặc điều khiển không tốt.

2.1.2. Thuật toán bám quỹ đạo Stanley Steering Controller

Bộ điều khiển Stanley là bộ điều khiển bám quỹ đạo đơn giản nhưng hữu ích cho người máy tự động và xe tự hành. Bộ điều khiển này đã được đội đua Stanford sử dụng để giành chiến thắng trong sự kiện Darpa Grand Challenge lần thứ hai. Nội dung chính của việc tạo ra bộ điều khiển Stanley chính là sự thay đổi vị trí tham chiếu có thể dẫn đến các thuộc tính khác nhau của bộ điều khiển.

- Sử dụng tâm của trục trước làm điểm tham chiếu trên xe.
- Sử dụng cả lỗi trong hướng chuyển động và lỗi ở vị trí liên quan đến điểm gần nhất trên quỹ đạo.
- Bộ điều khiển Stanley giới hạn đầu ra của nó nằm trong giới hạn của góc lái.

Tất cả ba điều kiện này hình thành cơ sở cho việc hình thành luật điều khiển. Crosstrack error được đo so với trục trước của xe và điểm tham chiếu trên quỹ đạo không có khoảng cách look – ahead.



Hình 2.5 Mô hình sử dụng trong thuật toán Stanley Controller

Với $\psi(t)$ là góc giữa quỹ đạo và hướng của xe, $\delta(t)$ là góc lái. Đầu tiên, để loại bỏ lỗi của hướng chuyển động so với quỹ đạo, góc lái được đặt bằng với hướng chuyển động:

$$\delta(t) = \psi(t)$$

Sau đó, để loại bỏ crosstrack error, một bộ điều khiển tỷ lệ được thêm vào, mức tăng của nó tỷ lệ nghịch với vận tốc. Bộ điều khiển sau đó được đưa qua một hàm \tan^{-1} để tín hiệu điều khiển nằm trong khoảng từ $-\pi$ đến π :

$$\delta(t) = \tan^{-1} \left(\frac{ke(t)}{v_f(t)} \right)$$

Cuối cùng, lệnh điều khiển góc lái được giữ trong khoảng góc lái tối thiểu và tối đa, thường đổi xứng qua 0:

$$\delta(t) \in [\delta_{min}, \delta_{max}]$$

Luật điều khiển cuối cùng kết hợp ba yếu tố này để thiết lập góc lái của chiếc xe:

$$\delta(t) = \psi(t) + \tan^{-1} \left(\frac{ke(t)}{v_f(t)} \right)$$

$$\delta(t) \in [\delta_{min}, \delta_{max}]$$

Tuy nhiên, trên thực tế, bộ điều khiển Stanley vẫn là bộ điều khiển bám quỹ đạo hình học (geometric path tracking controller) và do đó không xem xét nhiều

khía cạnh khác nhau của xe tự hành thực sự. Ví dụ, nó không xem xét các phép đo, động lực học cơ cấu hoặc hiệu ứng lực căng của lốp xe, tất cả đều có thể gây ra các đặc điểm không mong muốn trong quá trình điều khiển. Tuy nhiên, có thể thực hiện một vài điều chỉnh cho bộ điều khiển giúp giảm thiểu một số hiệu ứng không mong muốn này.

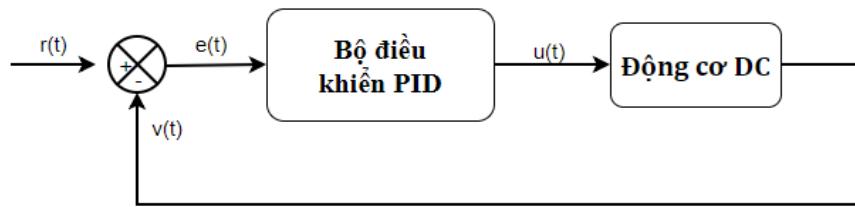
Trong quá trình vận hành ở tốc độ thấp, vì thuật ngữ vận tốc nằm trong mẫu số của phân số bên trong \tan^{-1} , những giá trị sai số nhỏ trong vận tốc có xu hướng được khuếch đại trong tín hiệu điều khiển góc lái. Điều này dẫn đến sự thay đổi lớn trong tay lái, điều không mong muốn cho người lái. Vì vậy, để thoát khỏi vấn đề này và để tăng tính ổn định ở vận tốc thấp, chúng ta thêm một hằng số k_s (softening constant) dương để đảm bảo mẫu số luôn có một giá trị nhỏ nhất. Hằng số k_s này có thể được điều chỉnh trong thực nghiệm.

$$\delta(t) = \psi(t) + \tan^{-1} \left(\frac{ke(t)}{k_s + v_f(t)} \right)$$

2.1.3. Bộ điều khiển PI mờ

2.1.3.1. Bộ điều khiển PID

Vận dụng thuật toán bộ điều khiển PID trong việc điều khiển động cơ DC, sử dụng hàm truyền của bộ điều khiển PID rời rạc từ đó tìm được phương trình điều khiển.



Hình 2.6 Sơ đồ khối bộ điều khiển PID

Trong đó:

- r : Giá trị đặt
- v : Giá trị hiện tại

- e : Sai số giá trị đặt và giá trị hiện tại $e = r - v$

Ta có hàm truyền của bộ điều khiển PID rời rạc có dạng như sau:

$$G(z) = \frac{U(z)}{E(z)} = K_P + \frac{K_I T}{2} \frac{z+1}{z-1} + \frac{K_D}{T} \frac{z-1}{z}$$

Viết lại phương trình trên ta được:

$$G(z) = \frac{\left(K_P + \frac{K_I T}{2} + \frac{K_D}{T}\right) + \left(-K_P + \frac{K_I T}{2} - \frac{2K_D}{T}\right)z^{-1} + \left(\frac{K_D}{T}\right)z^{-2}}{1 - z^{-1}}$$

Đặt:

$$\begin{aligned} a_0 &= K_P + \frac{K_I T}{2} + \frac{K_D}{T} \\ a_1 &= -K_P + \frac{K_I T}{2} - \frac{2K_D}{T} \\ a_2 &= \frac{K_D}{T} \end{aligned}$$

Ta có:

$$G(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 - z^{-1}}$$

Từ đó, ta tính được giá trị ngõ ra của bộ điều khiển PID $u(k)$ khi giá trị ngõ vào là $e(k)$ như sau:

$$u(k) = G(z)e(k) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 - z^{-1}} e(k)$$

Thay các giá trị a_0, a_1 và a_2 vào phương trình trên và áp dụng tính chất dời thời gian theo biến đổi Z ta có:

$$\begin{aligned} u(k) &= u(k-1) + \left(K_P + \frac{K_I T}{2} + \frac{K_D}{T}\right)e(k) + \left(-K_P + \frac{K_I T}{2} - \frac{2K_D}{T}\right)e(k-1) \\ &\quad + \left(\frac{K_D}{T}\right)e(k-2) \end{aligned}$$

Viết lại phương trình trên ta tìm được giá trị tại các khâu P, I và D như sau:

$$P_{Part} = K_P(e(k) - e(k-1))$$

$$I_{Part} = \frac{K_I T}{2} (e(k) + e(k-1))$$

$$D_{Part} = \frac{K_D}{T} (e(k) - 2e(k-1) + e(k-2))$$

Từ đó ta có được phương trình bộ điều khiển PID rời rạc dùng cho điều khiển động cơ DC như sau:

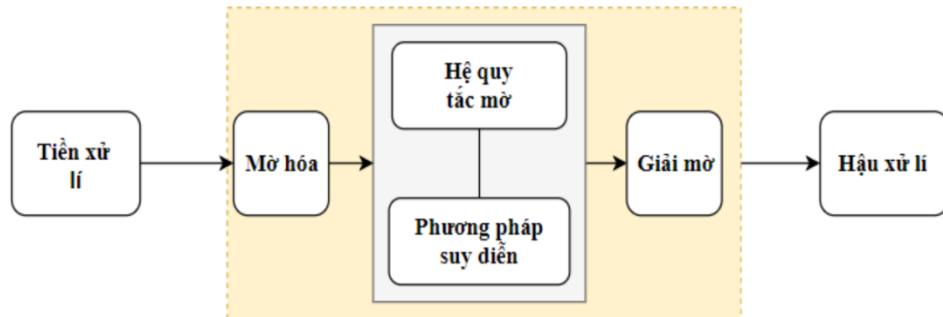
$$u(k) = u(k-1) + P_{Part} + I_{Part} + D_{Part}$$

Điều khiển tỉ lệ (K_P) có ảnh hưởng làm giảm thời gian lên và sẽ làm giảm nhưng không loại bỏ sai số xác lập. Điều khiển tích phân (K_I) sẽ loại bỏ sai số xác lập nhưng có thể là đáp ứng quá độ xấu đi. Điều khiển vi phân (K_D) có tác dụng là tăng sự ổn định của hệ thống, giảm vọt lố và cải thiện đáp ứng quá độ. Ảnh hưởng của mỗi bộ điều khiển K_P , K_I và K_D lên hệ thống vòng kín được tóm tắt ở bảng dưới đây.

Bảng 2.1 Ảnh hưởng của các hệ số lên bộ điều khiển PID

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

2.1.3.2. Bộ điều khiển mờ



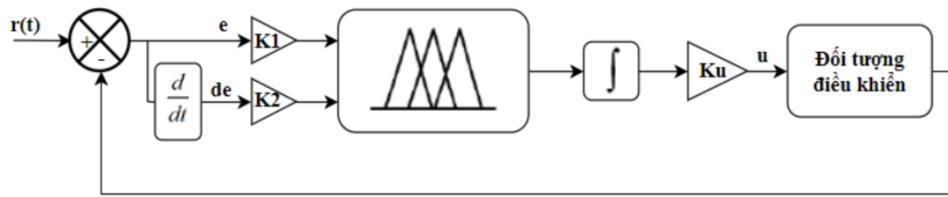
Hình 2.7 Sơ đồ khái niệm về sơ đồ khối cấu trúc bộ điều khiển Fuzzy

Sơ đồ khái niệm của một bộ điều khiển mờ cơ bản bao gồm các khía:

- **Khối tiền xử lí:** Tín hiệu vào bộ điều khiển thường là các giá trị rõ từ các mạch đo, bộ tiền xử lí có chức năng xử lí các giá trị đo này trước khi đưa vào bộ điều khiển mờ. Khối tiền xử lí có thể:
 - Vi phân, tích phân tín hiệu.
 - Chuẩn hóa, lượng tử hóa.
 - Lọc nhiễu.
- **Khối mờ hóa:** Chuyển giá trị rõ phản hồi từ ngõ ra của đối tượng thành giá trị mờ để hệ quy tắc có thể suy luận được.
- **Khối hệ quy tắc mờ:** Có thể được xem như mô hình toán học biểu diễn tri thức, kinh nghiệm của con người trong việc giải quyết bài toán dưới dạng các phát biểu ngôn ngữ. Các hệ quy tắc mờ thường dùng: Quy tắc mờ mamdani và quy tắc mờ sugeno.
 - 2 phương pháp suy diễn Max – Min và Max – Prob là 2 phương pháp dùng trong suy luận từ hệ quy tắc mờ.
- **Khối giải mờ:** Chuyển giá trị giải mờ suy luận được ở ngõ ra của khối hệ quy tắc mờ thành giá trị rõ để điều khiển đối tượng.
 - Giải mờ (defuzzification) là chuyển đổi giá trị mờ ở ngõ ra của hệ mờ thành giá trị rõ.

- Các phương pháp giải mờ có thể qui vào 2 nhóm:
 - Giải mờ dựa theo độ cao: thường dùng trong các bài toán phân nhóm.
 - Giải mờ dựa vào điểm trọng tâm: thường dùng trong các bài toán điều khiển.
- **Khối hậu xử lí:** Khuếch đại tín hiệu giải mờ chuẩn hóa thành giá trị vật lí.

2.1.3.3. Bộ điều khiển PI mờ



Hình 2.8 Sơ đồ khối thiết kế bộ điều khiển PI mờ

Bộ điều khiển PI mờ nếu thiết kế tốt có thể điều khiển đối tượng trong miền làm việc rộng với sai số xác lập bằng 0. Tuy nhiên bộ điều khiển PI làm chậm đáp ứng của hệ thống và trong nhiều trường hợp làm cho quá trình quá độ có dao động.

Trình tự thiết kế bộ điều khiển PI mờ:

Bước 1: Dựa trên sơ đồ khối thiết kế bộ điều khiển PI mờ, xác định tầm giá trị của:

- Biến vào: sai số (e) và vi phân sai số (\dot{e})
- Biến ra: vi phân của tín hiệu điều khiển (u)

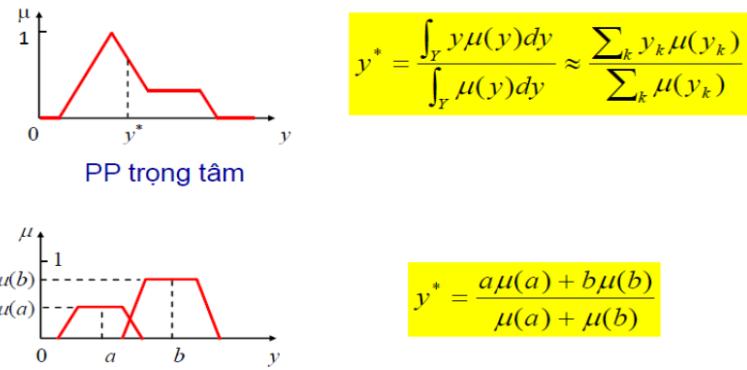
Bước 2: Xác định các hệ số chuẩn hóa giá trị biến vào, biến ra về miền giá trị $[-1,1]$.

Bước 3: Định nghĩa các giá trị ngôn ngữ cho biến vào và biến ra, định lượng các giá trị ngôn ngữ bằng tập mờ

Bước 4: Xây dựng hệ qui tắc mờ bằng cách vẽ hình minh họa, dựa trên đặc tính của đối tượng đang điều khiển mà đưa ra những một số quy tắc điển hình, sau đó có thể áp dụng tính đối xứng và tính liên tục của hệ mờ để đưa ra các quy tắc còn lại.

Bước 5: Chọn phương pháp suy diễn (Max – Min hay Max – Prod).

Bước 6: Chọn phương pháp giải mờ (trọng tâm hay trung bình có trọng số).



Hình 2.9 Phương pháp giải mờ trọng tâm

Bước 7: Mô phỏng hoặc thực nghiệm để đánh giá kết quả, tinh chỉnh thông số của bộ điều khiển để đạt được chất lượng mong muốn.

2.2. Điều khiển tránh vật cản

2.2.1. Mô hình MobileNet SSD

2.2.1.1. MobileNet

Mô hình MobileNet được đề xuất bởi nhóm tác giả Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam đến từ Google trong bài báo “MobileNets Efficient Convolutional Neural Networks for Mobile Vision Applications” được đăng tải năm 2017. Mục tiêu của mô hình này là xây dựng một mô hình mạnh mẽ nhưng nhỏ gọn, có thể chạy trên các thiết bị di động như điện thoại, máy tính bảng hoặc các thiết bị nhúng.

Cấu trúc mạng MobileNet

MobileNet có 30 lớp với các đặc điểm sau:

- Lớp 1: Convolution layer với stride = 2.
- Lớp 2: Depthwise layer.
- Lớp 3: Pointwise layer.
- Lớp 4: Depthwise layer với stride = 2.
- Lớp 5: Pointwise layer.
- Lớp 30: Softmax, dùng để phân lớp.

Sau mỗi lớp tích chập, MobileNet sẽ sử dụng Batch Normalization (BN) và ReLU.

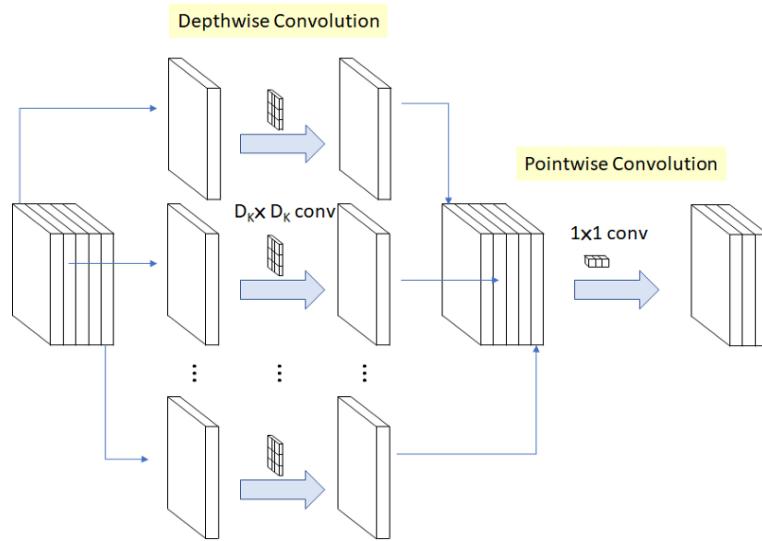
Table 1. MobileNet Body Architecture

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	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
5× 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 2.10 Cấu trúc mạng MobileNet

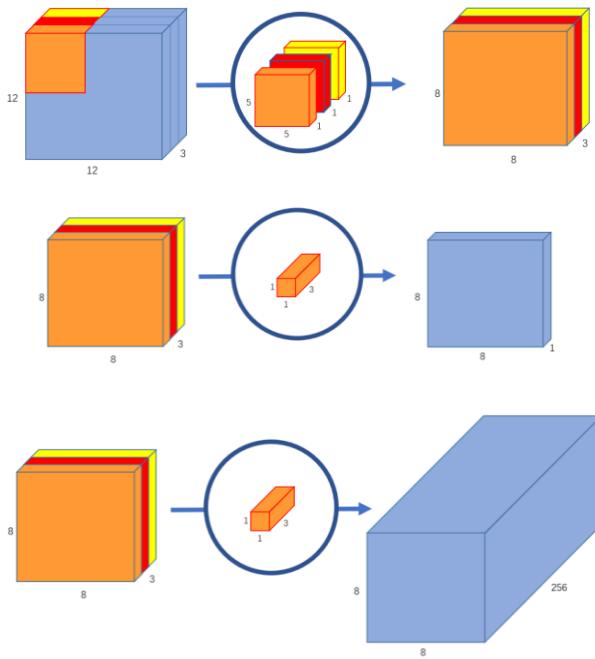
Điểm cải tiến của mô hình là sử dụng một cách tính tích chập có tên là Depthwise Separable Convolution để giảm kích thước mô hình và giảm độ phức

tập tính toán. Do đó, mô hình sẽ hữu ích khi chạy các ứng dụng trên di động và các thiết bị nhúng. Depthwise separable convolution là một depthwise convolution theo sau bởi một pointwise convolution.



Hình 2.11 Depthwise Separable Convolution

Giả sử rằng chúng ta có hình ảnh đầu vào là $12 \times 12 \times 3$ pixel. Để thực hiện một phép tích chập 5×5 trên ảnh mà $\text{padding} = 0$ và $\text{stride} = 1$, depthwise convolution sử dụng 3 kernels có kích thước $5 \times 5 \times 1$.



Hình 2.12 Depthwise convolution và Pointwise convolution

Mỗi kernel $5 \times 5 \times 1$ trượt qua trên 1 kênh của ảnh, tính tích chập cho mỗi nhóm 25 pixels, cho ra ảnh có kích thước $8 \times 8 \times 1$, sau đó xếp chồng các ảnh này lại với nhau tạo ra ảnh $8 \times 8 \times 3$.

Pointwise convolution sử dụng kernel 1×1 để trượt qua từng điểm. Kernel này có độ sâu phụ thuộc số kênh của ảnh đầu vào, trong trường hợp này là 3. Do đó, chúng ta trượt kernel $1 \times 1 \times 3$ qua ảnh $8 \times 8 \times 3$ ở trên để có được ảnh $8 \times 8 \times 1$.

Chúng ta có thể dùng 256 kernels $1 \times 1 \times 3$ tạo ra ảnh $8 \times 8 \times 1$ cho mỗi kernel để có được ảnh cuối cùng có kích thước $8 \times 8 \times 256$.

Kết quả huấn luyện mạng

Với M là số lượng input channel, N là số lượng output channel, D_k là kernel size, D_f là feature map size, chúng ta có thể tính được:

- Chi phí tính toán của Depthwise convolution:

$$D_k \cdot D_k \cdot M \cdot D_f \cdot D_f$$

- Chi phí tính toán của Pointwise convolution:

$$M \cdot N \cdot D_f \cdot D_f$$

- Tổng chi phí tính toán của Depthwise Separable Convolution:

$$D_k \cdot D_k \cdot M \cdot D_f \cdot D_f + M \cdot N \cdot D_f \cdot D_f$$

- Chi phí tính toán của phép tích chập thuần là:

$$D_k \cdot D_k \cdot M \cdot N \cdot D_f \cdot D_f$$

Do đó, chi phí tính toán sẽ giảm:

$$\frac{(D_k \cdot D_k \cdot M \cdot D_f \cdot D_f + M \cdot N \cdot D_f \cdot D_f)}{(D_k \cdot D_k \cdot M \cdot N \cdot D_f \cdot D_f)} = \frac{1}{N} + \frac{1}{D_k^2}$$

Với mong muốn làm mô hình gọn nhẹ hơn nữa, nhóm tác giả đã thêm vào hai tham số α và ρ .

- Tham số α : Điều khiển số lượng channel (M và N).
- Tham số ρ : Điều khiển độ phân giải của ảnh đầu vào.

Giá trị của α và ρ nằm trong đoạn [0,1]. Chi phí tính toán của MobileNet khi sử dụng thêm 2 tham số α và ρ :

$$D_k \cdot D_k \cdot \alpha M \cdot \rho D_f \cdot \rho D_f + \alpha M \cdot \alpha N \cdot \rho D_f \cdot \rho D_f$$

MobileNet cho kết quả tốt ngang với các mô hình hiện đại nhưng có số lượng tham số nhỏ hơn và số phép tính toán ít hơn. Điều này đạt được là nhờ vào việc sử dụng *Depthwise Separable Convolution*.

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

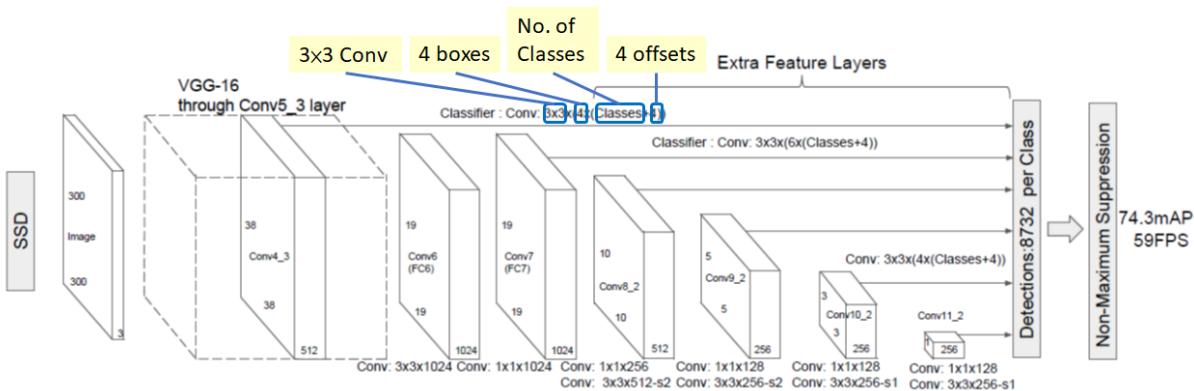
Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.50 MobileNet-160	60.2%	76	1.32
SqueezeNet	57.5%	1700	1.25
AlexNet	57.2%	720	60

Model	Top-1 Accuracy	Million Mult-Adds	Million Parameters
Inception V3 [18]	84%	5000	23.2
1.0 MobileNet-224	83.3%	569	3.3
0.75 MobileNet-224	81.9%	325	1.9
1.0 MobileNet-192	81.9%	418	3.3
0.75 MobileNet-192	80.5%	239	1.9

Hình 2.13 So sánh MobileNet với các mô hình khác

2.2.1.2. Single Shot Multibox Detector (SSD)

Cấu trúc mạng SSD



Hình 2.14 Cấu trúc mạng SSD

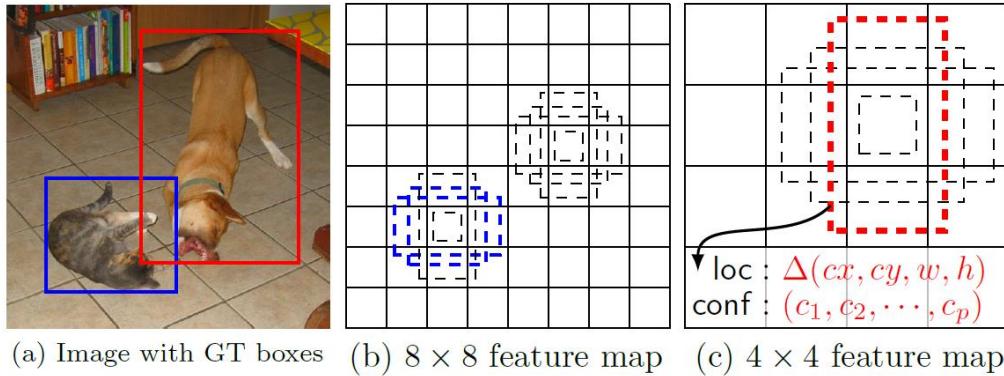
Bộ phát hiện đối tượng SSD gồm có 2 phần:

- Trích xuất bản đồ đặc trưng
- Áp dụng bộ lọc tích chập để phát hiện đối tượng.

Đầu tiên, SSD sử dụng VGG16 để trích xuất bản đồ đặc trưng. Nhưng ở đây chúng ta sẽ thay thế VGG16 bằng MobileNet vì cấu trúc đơn giản của nó. Sau khi trải qua một số lớp tích chập nhất định, chúng ta có được lớp đặc trưng có kích thước $m \times n \times p$, và một lớp tích chập 3×3 được áp dụng trên lớp đặc trưng $m \times n \times p$ này. Đối với mỗi vị trí, chúng ta có (k) bounding box. Các bounding box này có kích thước và tỷ lệ khung hình khác nhau. Đối với mỗi bounding box, chúng ta sẽ tính c điểm cho c lớp cần phân loại và 4 điểm offsets ($\Delta cx, \Delta cy, w, h$) tương ứng với hình

dạng bounding box mặc định ban đầu. Do đó, chúng ta có $(c + 4)*k$ bộ lọc được áp dụng tại mỗi ô của bản đồ đặc trưng và $(c + 4)*k*m*n$ đầu ra.

Trên thực tế, SSD sử dụng nhiều lớp bản đồ đặc trưng đa tỷ lệ (multi – scale feature map). Các lớp bản đồ đặc trưng khác nhau này cũng trải qua một lớp tích chập 3×3 nhỏ để phát hiện đối tượng một cách độc lập.



Hình 2.15 SSD sử dụng nhiều bounding box

Khi CNN giảm dần kích thước không gian, độ phân giải của các bản đồ đặc trưng cũng giảm. Bản đồ đặc trưng với độ phân giải cao hơn có trách nhiệm phát hiện các vật thể nhỏ hơn. Lớp đầu tiên để phát hiện đối tượng Conv4_3 có kích thước 38×38 , giảm khá nhiều so với kích thước ảnh đầu vào. Do đó, SSD thường hoạt động kém đối với các đối tượng nhỏ so với các phương pháp phát hiện đối tượng khác.

Mỗi lớp bản đồ đặc trưng sử dụng một bộ bounding box mặc định có tâm tại các ô tương ứng. Các lớp khác nhau sử dụng các bộ bounding box mặc định khác nhau để tùy chỉnh phát hiện đối tượng ở các độ phân giải khác nhau.

Giả sử chúng ta có (m) bản đồ đặc trưng để dự đoán, chúng ta có thể tính S_k cho bản đồ đặc trưng thứ (k) . S_{\min} là $0,2$, S_{\max} là $0,9$. Điều đó có nghĩa là tỷ lệ ở lớp thấp nhất là $0,2$ và tỷ lệ ở lớp cao nhất là $0,9$. Tỷ lệ ở các lớp ở giữa cách đều nhau.

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

Mỗi tỷ lệ, S_k , chúng ta có 5 tỷ lệ khung hình:

$$a_r \in \left\{1, 2, 3, \frac{1}{2}, \frac{1}{3}\right\}$$

$$w_k^a = s_k \sqrt{a_r}$$

$$h_k^a = \frac{s_k}{\sqrt{a_r}}$$

Đối với $a_r = 1$, chúng ta thêm 1 bounding box mặc định có tỷ lệ s'_k :

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

Do đó, chúng ta có thể có tối đa 6 bounding box với các tỷ lệ khung hình khác nhau. Đối với các lớp chỉ có 4 bounding box, $a_r = \frac{1}{3}$ và $a_r = 3$ được bỏ qua.

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 $IoU > 0,45$ 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.

Hàm mất mát

Kết quả của SSD được phân loại thành kết quả tích cực (positive matches) hoặc kết quả tiêu cực (negative matches). SSD chỉ sử dụng kết quả tích cực trong việc tính toán L_{loc} . Nếu bounding box mặc định tương ứng với bounding box dự đoán có $IoU > 0,5$ với bounding box thực tế, thì kết quả là tích cực. Nếu không, nó là tiêu cực.

Ta có:

$$x_{ij}^p = \begin{cases} 1 & \text{if } IoU > 0.5 \text{ between default box } i \text{ and ground truth box } j \text{ on class } p \\ 0 & \text{otherwise} \end{cases}$$

Hàm mất bao gồm hai phần: L_{conf} và L_{loc} , trong đó N là số kết quả tích cực và α là trọng số cho hàm L_{loc} (thường được chọn bằng 1). Nếu $N = 0$, chúng ta đặt $L(x, c, l, g) = 0$.

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

L_{loc} (localization loss) là sự không phù hợp giữa bounding box thực tế (g) và bounding box dự đoán (l), có công thức là hàm smooth giữa các tham số của (l) và (g). Các tham số này bao gồm các độ lệch cho điểm trung tâm (cx, cy), chiều rộng (w) và chiều cao (h) của bounding box dự đoán.

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^N \sum_{m \in \{cx, cy, w, h\}} x_{ij}^k \cdot smooth_{L1}(l_i^m - \hat{g}_j^m)$$

Trong đó:

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & if |x| < 1 \\ |x| - 0.5 & otherwise \end{cases}$$

$$\hat{g}_j^{cx} = \frac{g_j^{cx} - d_i^{cx}}{d_i^w}$$

$$\hat{g}_j^{cy} = \frac{g_j^{cy} - d_i^{cy}}{d_i^h}$$

$$\hat{g}_j^w = \log \left(\frac{g_j^w}{d_i^w} \right)$$

$$\hat{g}_j^h = \log \left(\frac{g_j^h}{d_i^h} \right)$$

L_{conf} (confidence loss) là độ mất mát khi thực hiện dự đoán lớp của đối tượng. Đối với các kết quả tích cực, chúng ta tính độ mất mát theo điểm tin cậy của lớp tương ứng. Đối với các kết quả tiêu cực, chúng ta tính độ mất mát theo điểm tin cậy của lớp 0, lớp dành cho các bounding box không chứa đối tượng nào.

$$L_{conf}(x, c) = - \sum_{i \in Pos}^N x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in Neg} \log(\hat{c}_i^0) \text{ where } \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

Kết quả huấn luyện mạng

SSD có hai mô hình: SSD300 và SSD512.

- SSD300: hình ảnh đầu vào 300×300 , độ phân giải thấp hơn, nhanh hơn.
- SSD512: hình ảnh đầu vào 512×512 , độ phân giải cao hơn, chính xác hơn.

SSD được thiết kế để phát hiện đối tượng trong thời gian thực. Theo so sánh sau đây, nó đạt được tốc độ xử lý thời gian thực và thậm chí đánh bại độ chính xác của Faster R – CNN.

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 2.16 So sánh giữa các mạng phát hiện đối tượng

SSD300 đạt 77.2% mAP ở 46 FPS trong khi SSD500 đạt 79.8% mAP ở 19 FPS, vượt trội so với Faster R-CNN (73,2% mAP ở 7 FPS) và YOLOv1 (63,4% mAP ở 45 FPS).

Đề tài này sử dụng mô hình MobileNet SSD với trọng số được huấn luyện trên tập dữ liệu VOC0712 và mAP = 72.7%. Tập dữ liệu VOC0712 là tập dữ liệu có độ phân giải cao bao gồm khoảng 20.000 hình ảnh với 20 lớp:

Bảng 2.2 Các lớp của tập dữ liệu VOC0712

ID	Đối tượng						
1	Máy bay	6	Xe buýt	11	Bàn ăn	16	Chậu cây
2	Xe đạp	7	Xe ô tô	12	Chó	17	Cừu
3	Chim	8	Mèo	13	Ngựa	18	Ghế sofa
4	Tàu thủy	9	Ghế	14	Xe máy	19	Tàu hỏa
5	Chai, lọ	10	Bò	15	Người	20	Tivi/màn hình

2.2.2. Thuật toán theo vết đối tượng MOSSE

2.2.2.1. Theo dõi dựa trên bộ lọc tương quan

Bộ theo dõi dựa trên bộ lọc mô hình hóa sự xuất hiện của các đối tượng bằng các bộ lọc được đào tạo trên các ảnh của chúng. Ban đầu, mục tiêu được chọn dựa trên một cửa sổ nhỏ có tâm là tâm của đối tượng trong khung đầu tiên. Từ thời điểm này, việc theo dõi và cập nhật bộ lọc được thực hiện cùng lúc. Trong khung tiếp theo, mục tiêu được theo dõi bằng tính các giá trị tương quan với bộ lọc của các cửa sổ trong một vùng tìm kiếm. Vị trí tương ứng với giá trị tương quan tối đa cho biết vị trí mới của mục tiêu. Và việc cập nhật bộ lọc trực tuyến được thực hiện dựa trên vị trí mới đó.

Định lý Convolution nói rằng phép tích chập trở thành phép nhân theo từng phần tử trong miền Fourier. Do đó, để tăng tốc độ tính toán, độ tương quan được tính trong miền Fourier sử dụng phép biến đổi Fourier nhanh FFT (Fast Fourier Transform). Đầu tiên, biến đổi Fourier 2D của hình ảnh đầu vào $F = F(f)$ và của bộ lọc $H = F(h)$ được tính toán. Khi đó:

$$G = F \odot H^*$$

Trong đó, ký hiệu \odot biểu thị cho phép nhân theo từng phần tử và $*$ biểu thị cho liên hợp phức. Cuối cùng, giá trị tương quan đầu ra được chuyển trở lại vào miền không gian bằng cách sử dụng phép nghịch đảo FFT.

2.2.2.2. Bộ theo dõi MOSSE

Bộ theo dõi MOSSE học một bộ lọc tương quan phân biệt để xác định vị trí của mỗi mục tiêu trong khung hình mới. Phương pháp này sử dụng một số ảnh xám f_1, f_2, \dots, f_t của mục tiêu trong các khung hình trước đó làm dữ liệu đào tạo. Chúng được gắn nhãn với các đầu ra tương quan mong muốn g_1, g_2, \dots, g_t . Bộ lọc tương quan tối ưu h_t tại thời điểm t có được bằng cách giảm thiểu tổng bình phương:

$$\varepsilon = \sum_{j=1}^t \|h_t * f_j - g_j\|^2 = \frac{1}{MN} \sum_{j=1}^t \|\bar{H}_t F_j - G_j\|^2$$

Trong đó, f_j , g_j và h_t có kích thước $M \times N$. Ký hiệu $*$ biểu thị phép tích chập. F_i , G_i và H tương ứng là biến đổi Fourier của f_i , g_i và h . \bar{H}_t biểu thị cho liên hợp phức và $\bar{H}_t F_j$ được thực hiện theo từng phần tử. Biểu thức đạt giá trị nhỏ nhất tại:

$$H_t = \frac{\sum_i \bar{G}_i \ F_i}{\sum_i \bar{F}_i \ F_i}$$

Đầu ra mong muốn g_j được xây dựng như một hàm Gaussian với đỉnh nằm ở tâm của đối tượng trong f_j . Trong thực tế, tử số A_t và mẫu số B_t của H_t được cập nhật riêng với f_t mới của mục tiêu bằng cách lấy trung bình có trọng số:

$$H_t = \frac{A_t}{B_t}$$

$$A_t = (1 - \eta)A_{t-1} + \eta \bar{G}_i \ F_i$$

$$B_t = (1 - \eta)B_{t-1} + \eta \bar{F}_i \ F_i$$

Trong đó η là learning rate.

Cho một ảnh z có kích thước $M \times N$ trong một khung mới, độ tương quan y được tính $y = F^{-1}(\bar{H}_t z)$. Ở đây F^{-1} biểu thị cho phép nghịch đảo DFT. Vị trí mới

của đối tượng được ước tính là ở điểm có độ tương quan y tối đa. Các bước cập nhật và phát hiện được thực hiện hiệu quả bằng cách sử dụng biến đổi Fourier nhanh FFT (Fast Fourier Transform).

2.2.3. Thuật toán Block Matching

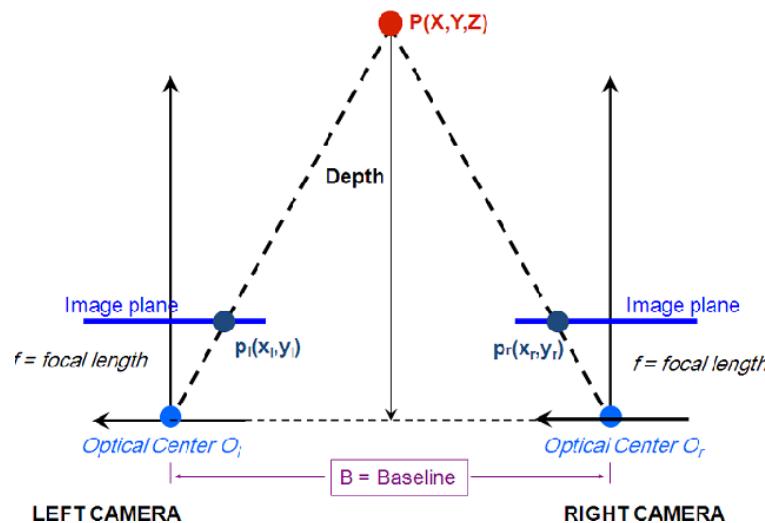
2.2.3.1. Nguyên lý tính khoảng cách từ stereo camera

Stereo camera là một cặp gồm hai camera giống nhau về các thông số vật lý cấu hình nên camera. Tương tự như với mắt người, stereo camera có thể cho được ảnh với không gian ba chiều từ môi trường xung quanh.

Giả sử, ta có một nguồn sáng được đặt ngang với camera cách một đoạn là b , thấu kính camera có tiêu cự là f và nguồn sáng nằm trên ảnh có khoảng cách là d . Từ đó, ta có thể dễ dàng tính được khoảng cách từ camera đến mặt phẳng cần đo là:

$$Z = \frac{bf}{d}$$

Nếu ta thay nguồn sáng đó thành một camera khác thì ta có sơ đồ như sau:



Hình 2.17 Nguyên lý hoạt động stereo camera

Như giả sử ban đầu, ta có hai camera có cùng các thông số vật lý như tiêu cự, góc nhìn,... Dựa vào tam giác đồng dạng, ta có:

$$\frac{b_1}{Z} = -\frac{x_1}{f}, \frac{b_2}{Z} = \frac{x_2}{f}$$

Do $b = b_1 + b_2$ nên:

$$b = \frac{Z}{f} * (x_2 - x_1) \Rightarrow Z = \frac{bf}{x_2 - x_1}$$

Đặt $d = x_2 - x_1$, ta gọi d là độ chênh lệch (disparity).

Khi đó, tọa độ X, Y của điểm P là:

$$X = x_l * \frac{Z}{f} = b + x_r * \frac{Z}{f}$$

$$Y = y_l * \frac{Z}{f} = y_r * \frac{Z}{f}$$

Trong đó (x_l, y_l) và (x_r, y_r) lần lượt là tọa độ của điểm P trên ảnh của camera trái và phải (đơn vị pixel).

2.2.3.2. Thuật toán Block Matching

Thuật toán Block Matching sử dụng những cửa sổ SAD (Sum of Absolute Difference) nhỏ để tìm các điểm tương ứng giữa ảnh trái và ảnh phải đã được hiệu chỉnh của stereo camera. Thuật toán này dùng để kết hợp hai hình ảnh, một trái và một phải, thành một bản đồ chênh lệch (disparity map) duy nhất. Giá trị của mỗi pixel trong bản đồ chênh lệch có thể dùng để tính khoảng cách từ máy ảnh đến đối tượng mà pixel này đại diện.

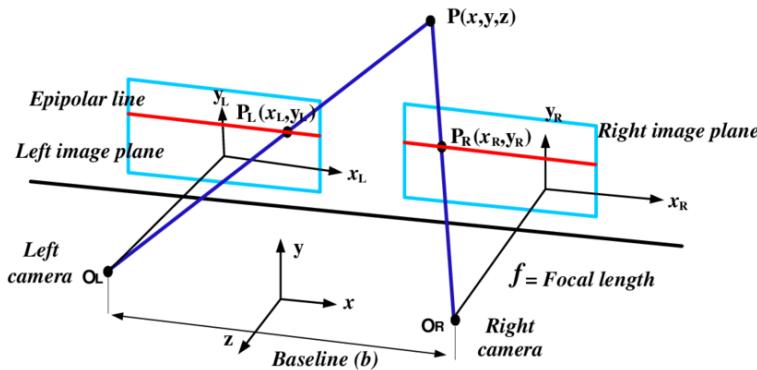
Thuật toán Block Matching có ba giai đoạn, hoạt động trên các cặp hình ảnh stereo đã được hiệu chỉnh:

- Lọc trước để chuẩn hóa độ sáng của hình ảnh và tăng cường kết cấu.
- Tìm kiếm các điểm tương ứng dọc theo các đường epipolar ngang bằng cửa sổ SAD.
- Lọc sau để loại bỏ các kết quả xấu.

Trong bước lọc trước, chúng ta chuẩn hóa hình ảnh đầu vào để giảm sự khác biệt về ánh sáng và tăng cường kết cấu hình ảnh. Chúng ta thực hiện điều này bằng

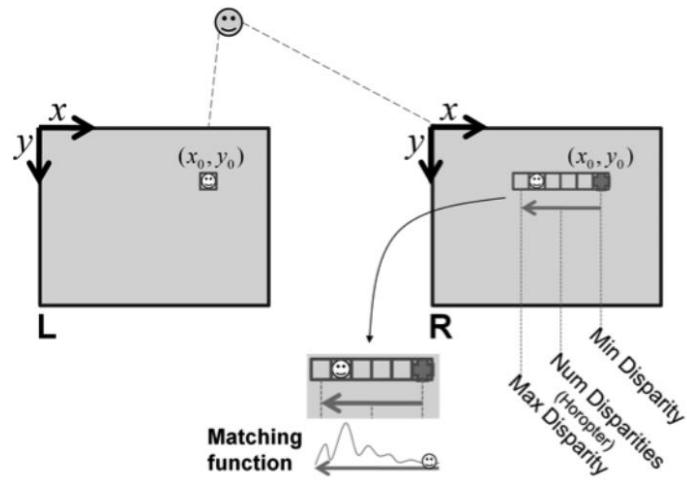
cách trượt một cửa sổ có kích thước 5×5 , 7×7 (mặc định), ..., 21×21 (tối đa) qua ảnh. Pixel trung tâm I_c của cửa sổ được thay thế bằng $\min(\max(I_c - \bar{I}, -I_{cap}), I_{cap})$, trong đó \bar{I} là giá trị trung bình trong cửa sổ và $I_{cap} > 0$ là giới hạn có giá trị mặc định là 30.

Tiếp theo, chúng ta tính toán các điểm tương ứng bằng cách trượt cửa sổ SAD. Đối với mỗi pixel trong hình ảnh bên trái, chúng tôi tìm kiếm pixel trong hàng tương ứng trong hình ảnh bên phải để có kết quả phù hợp nhất. Sau khi hiệu chỉnh, mỗi hàng là một đường epipolar, vì vậy vị trí tương ứng trong hình ảnh bên phải phải nằm cùng hàng (cùng một tọa độ y) như trong hình ảnh bên trái.



Hình 2.18 Đường epipolar

Vị trí phù hợp này có thể được tìm thấy nếu đối tượng có đủ kết cấu để có thể phát hiện được và nếu nó không bị che khuất trong ảnh bên phải. Nếu tọa độ pixel của đối tượng bên trái là (x_0, y_0) , thì đối với hai máy ảnh được bố trí song song, pixel tương ứng (nếu có) phải được tìm thấy trên cùng một hàng và tại hoặc ở bên trái x_0 (độ chênh lệch lớn hơn hoặc bằng 0). Đối với các camera đặt nghiêng về phía nhau, pixel tương ứng có thể nằm bên phải x_0 (độ chênh lệch âm). Thuật toán cần được thông báo về chênh lệch tối thiểu mà nó sẽ gặp phải. Sau đó, việc tìm kiếm điểm tương ứng được thực hiện trên một số lượng độ chênh lệch đã chọn trước, được tính bằng pixel (mặc định là 64 pixel).



Hình 2.19 Nguyên lý cơ bản của thuật toán Block Matching

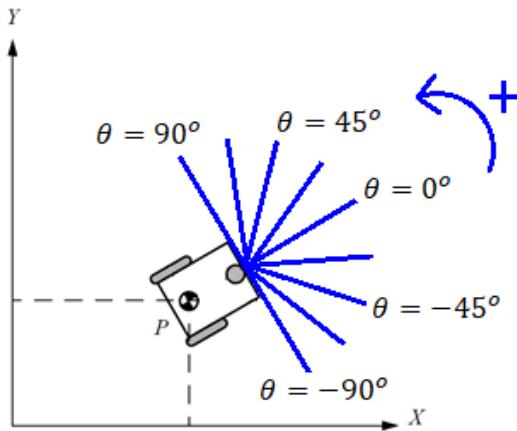
Sau khi tìm được các điểm tương ứng, chúng ta chuyển sang lọc sau. Bộ lọc sau sử dụng hàm phù hợp thông qua khái niệm về tỷ lệ duy nhất (uniqueness ratio) để ngăn chặn các kết quả tương ứng sai. Bộ lọc này về cơ bản yêu cầu rằng giá trị cửa sổ SAD của pixel tương ứng được chọn phải lớn hơn giá trị cửa sổ tại các pixel khác theo một tỷ lệ.

Cuối cùng, thuật toán Block Matching có thể gặp vấn đề ở gần ranh giới của các đối tượng vì cửa sổ bắt gặp đối tượng ở một bên và nền ở phía bên kia. Điều này dẫn đến một khu vực có cả các giá trị độ chênh lệch lớn và nhỏ mà chúng ta gọi là đốm (speckle). Để ngăn chặn những pixel tương ứng trên đường biên này, chúng ta có thể sử dụng một thuật toán phát hiện đốm trên một cửa sổ đốm (speckle window). Mỗi pixel sẽ được sử dụng làm cơ sở cho việc xây dựng một vùng. Vùng này chỉ bao gồm các pixel lân cận nếu nó nằm trong một phạm vi đốm (speckle range) của pixel hiện tại. Sau khi vùng này được tính toán, nếu nó có kích thước nhỏ hơn cửa sổ đốm, thì nó được coi là đốm.

2.2.4. Thuật toán tránh vật cản

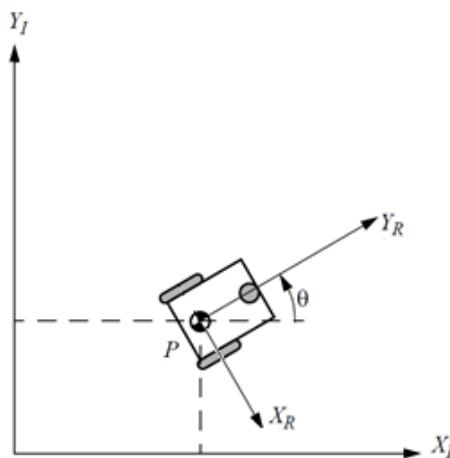
Thuật toán tránh chướng ngại vật được xây dựng dựa trên phương pháp The Curvature – Velocity Method, có thể được phát biểu như sau: Chọn cặp (v, θ) để

tối đa một số hàm mục tiêu, trong khi đáp ứng tất cả các ràng buộc cho phép. Trong đó, v và θ là vận tốc tịnh tiến và góc lái của robot. Giả định rằng robot luôn di chuyển dọc theo các đoạn thẳng, trong đó góc lái dương biểu thị chuyển động sang trái và ngược lại. Khi đó, mỗi điểm (v, θ) tương ứng với một chuyển động thẳng của robot trong không gian Cartesian.



Hình 2.20 Chuyển động của mô hình với các góc lái khác nhau

Trong thuật toán này, tất cả các giá trị được tính trong hệ tọa độ của mô hình, tức là hệ tọa độ với gốc tọa độ tại tâm của mô hình, trục y theo hướng hiện tại của mô hình, chiều dương hướng ra xa, và trục x vuông góc, chiều dương hướng từ trái sang phải theo hướng của mô hình.



Hình 2.21 Hệ tọa độ của mô hình

2.2.4.1. Các ràng buộc

Các giới hạn vật lý của robot đặt ra ràng buộc về vận tốc tịnh tiến và góc lái tối đa:

$$0 \leq v \leq v_{MAX} \text{ và } -\theta_{MAX} \leq \theta \leq \theta_{MAX}$$

Các chướng ngại vật trong môi trường cũng đặt ra các ràng buộc quan trọng. Chúng ta có thể biến các chướng ngại vật trong không gian Cartesian thành các ràng buộc như sau: Cho một tập các chướng ngại vật OBS, hàm khoảng cách được định nghĩa là:

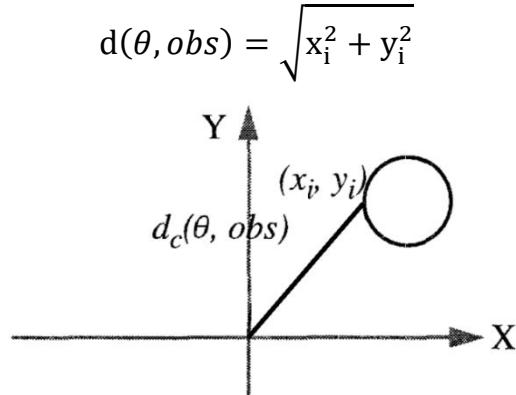
$$D(\theta, OBS) = \min_{obs \in OBS} d(\theta, obs)$$

Trong đó, $d(\theta, obs)$ là độ dài đoạn đường robot có thể đi được theo hướng θ trước khi va chạm với chướng ngại vật obs . Do phạm vi camera bị giới hạn, và để tránh tính toán với các giá trị vô hạn, chúng ta cắt hàm D ở một khoảng cách giới hạn L :

$$D_{limit}(\theta, OBS) = \min(L, D(\theta, OBS))$$

Việc tính toán hàm khoảng $d(\theta, obs)$ có thể rất phức tạp đối với các chướng ngại vật có hình dạng tùy ý. Để giải quyết điều này, chúng ta xấp xỉ các chướng ngại vật như các vòng tròn. Vì robot cũng có hình tròn, chúng ta có thể chuyển đổi robot thành một điểm trong tọa độ Cartesian bằng cách tăng bán kính của các chướng ngại vật một khoảng bằng bán kính của robot. Và để bù nhiêu cho cảm biến, chúng ta tăng bán kính của các chướng ngại vật bằng một giới hạn an toàn. Chúng ta sử dụng một mức an toàn tương đối nhỏ (0.2m), vì kích thước của giới hạn an toàn tỷ lệ nghịch với mức độ hẹp mà robot có thể đi qua. Giới hạn an toàn quá lớn sẽ khiến cho robot gặp khó khăn khi điều hướng qua các đoạn đường đồng đúc.

Bây giờ, với một robot tại tâm trục tọa độ đang chuyển động theo chiều dương trục Oy, với điều kiện chuyển động của robot theo hướng θ giao với chướng ngại vật obs tại (x_i, y_i) , chúng ta được:



Hình 2.22 Cách xác định $d(\theta, obs)$

Sau khi tìm được $D_{limit}(\theta, OBS)$, vận tốc v được giới hạn sao cho robot có thể di chuyển ít nhất T_{imp} giây trước khi gặp chướng ngại vật:

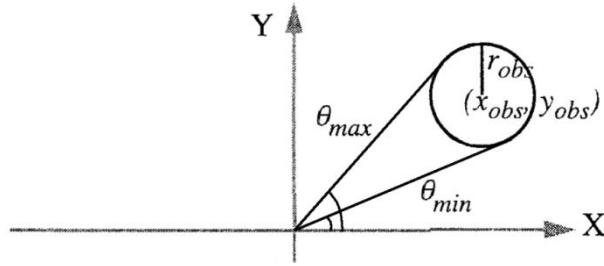
$$v \leq \frac{D_{limit}(\theta, OBS)}{T_{imp}}$$

Tuy nhiên, vẫn rất khó để tính toán D_{limit} . Thậm chí khi đã đơn giản hóa các chướng ngại vật thành hình tròn và có một công thức chung cho D_{limit} , sẽ rất tốn thời gian để tối ưu hóa $f(v, \theta)$. Để giải quyết vấn đề này, chúng ta xấp xỉ D_{limit} bằng một tập hữu hạn các khoảng, mỗi khoảng có một D_{limit} bằng hằng số. Tập hợp các khoảng này được xác định bằng các đường tiếp tuyến với chướng ngại vật để chia không gian thành các vùng có D_{limit} không đổi, và D_{limit} tại vị trí mà các khoảng chồng lấp lên nhau sẽ bằng D_{limit} nhỏ nhất của tất cả các khoảng đó.

Chúng ta tính toán D_{limit} bằng cách sử dụng một xấp xỉ của hàm $d(\theta, obs)$. Do đối với một chướng ngại vật cho trước obs , $d(\theta, obs) = +\infty$ bên ngoài các đường tiếp tuyến với chướng ngại vật, chúng ta chỉ cần xem xét các đường nằm giữa θ_{min} và θ_{max} :

$$\theta_{min} = \theta_c - \theta_t$$

$$\theta_{max} = \theta_c + \theta_t$$



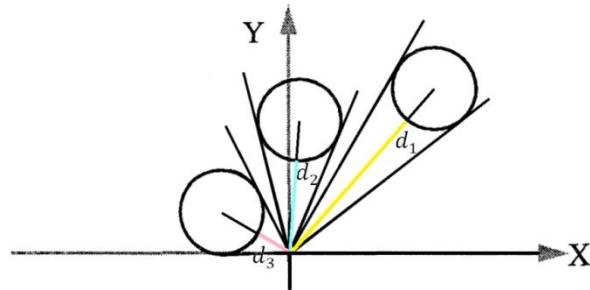
Hình 2.23 Cách xác định θ_{min} và θ_{max}

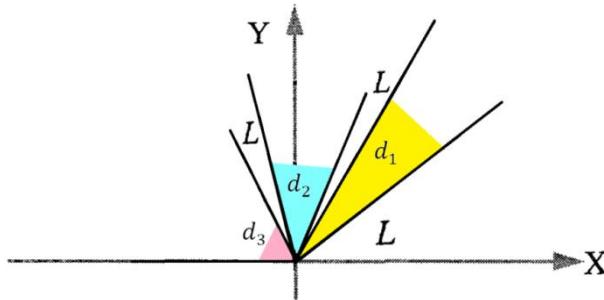
Trong đó:

$$\theta_c = \text{atan2}\left(\frac{y_{obs}}{x_{obs}}\right)$$

$$\theta_t = \arcsin\left(\frac{r_{obs}}{\sqrt{x_{obs}^2 + y_{obs}^2}}\right)$$

Phương trình trên không được xác định nếu vòng tròn chướng ngại vật trùng với gốc tọa độ ($x_{obs}^2 + y_{obs}^2 \leq r_{obs}^2$). Vì trong thực tế, chướng ngại vật và robot không thể chiếm cùng một không gian, điều này chỉ có thể xảy ra khi có nhiều cảm biến hoặc do các xấp xỉ làm tròn. Chúng ta đổi phó với các chướng ngại vật như vậy bằng cách đặt bán kính vòng tròn của chúng bằng $x_{obs}^2 + y_{obs}^2 - \xi$, trong đó $\xi = 0.01m$. Sau đó chúng ta có thể sử dụng phương trình trên để tính θ_{min} và θ_{max} cho tất cả các chướng ngại vật.





Hình 2.24 Cách chọn $d(\theta, obs)$ xấp xỉ cho từng khoảng

Ta lấy một giá trị $d(\theta, obs)$ xấp xỉ, không đổi giữa θ_{min} và θ_{max} , bằng khoảng cách từ gốc tọa độ đến điểm gần nhất trên đường tròn của chướng ngại vật obs . Điều này tạo ra một tập hợp các khoảng, mỗi khoảng có $d(\theta, obs)$ không đổi. Về mặt hình học, mỗi khoảng xác định một cặp đường thẳng, với giá trị $d(\theta, obs)$ giữa hai đường là không đổi.

Để tính toán D_{limit} , chúng ta liên kết các khoảng này bằng cách tách các khoảng chồng lên nhau và loại bỏ phần có $d(\theta, obs)$ lớn hơn. Min – union là một thuật toán hiệu quả sử dụng cấu trúc dữ liệu $((\theta_1, \theta_2), d_{1,2})$, trong đó θ_1 và θ_2 là θ_{min} và θ_{max} ($\theta_1 \leq \theta_2$) và $d_{1,2}$ là giá trị $d(\theta, obs)$ không đổi trong khoảng đó.

Thuật toán min – union bắt đầu với khoảng $((-\theta_{MAX}, \theta_{MAX}), L)$. Đối với mỗi chướng ngại vật, cặp đường tiếp tuyến và khoảng cách tương ứng được tính để tạo thành một khoảng mới $((\theta_{min}, \theta_{max}), d)$. Một khoảng hiện có $((\theta_1, \theta_2), d_{1,2})$ được sửa đổi tùy thuộc vào mối quan hệ của nó với khoảng mới như sau:

- Nếu $\theta_{min} \leq \theta_1 \leq \theta_2 \leq \theta_{max}$: $d_{1,2} = \min(d, d_{1,2})$.
- Nếu $\theta_1 \leq \theta_{min} \leq \theta_{max} \leq \theta_2$ và $d < d_{1,2}$: chia khoảng hiện có thành 3 khoảng $((\theta_1, \theta_{min}), d_{1,2}), ((\theta_{min}, \theta_{max}), d)$ và $((\theta_{max}, \theta_2), d_{1,2})$.
- Nếu $\theta_1 \leq \theta_{min} < \theta_2 \leq \theta_{max}$ và $d < d_{1,2}$: chia khoảng hiện có thành 2 khoảng $((\theta_1, \theta_{min}), d_{1,2})$ và $((\theta_{min}, \theta_2), d)$.
- Nếu $\theta_{min} \leq \theta_1 < \theta_{max} \leq \theta_2$ và $d < d_{1,2}$: chia khoảng hiện có thành 2 khoảng $((\theta_1, \theta_{max}), d)$ và $((\theta_{max}, \theta_2), d_{1,2})$.

- Các trường hợp còn lại: Không thay đổi.

2.2.4.2. Hàm mục tiêu

Với các ràng buộc trên, các lệnh điều khiển cho robot (v, θ) được chọn bằng cách tối ưu hóa một hàm mục tiêu. Hàm mục tiêu này ưu tiên tốc độ cao hơn, độ dài đoạn đường di chuyển lâu hơn trước khi chạm chướng ngại vật và cố gắng điều hướng robot di chuyển đến mục tiêu mong muốn. Chúng ta biểu diễn các tiêu chí này bằng một hàm mục tiêu tuyến tính đơn giản, trong đó mỗi thành phần đều được chuẩn hóa giữa 0 và 1:

$$f(v, \theta) = \alpha_1 \text{speed}(v) + \alpha_2 \text{dist}(\theta) + \alpha_3 \text{head}(\theta)$$

$$\text{speed}(v) = \frac{v}{v_{MAX}}$$

$$\text{dist}(\theta) = \frac{D_{limit}(\theta, OBS)}{L}$$

$$\text{head}(\theta) = 1 - \frac{|\theta_g - \theta|}{\pi}$$

Thành phần $\text{speed}(v)$ ưu tiên robot di chuyển nhanh hơn, $\text{dist}(\theta)$ ưu tiên cho việc di chuyển quãng đường dài hơn mà không gặp trở ngại, và $\text{head}(\theta)$ là sai số giữa góc lái hướng đến mục tiêu mong muốn θ_g (trong hệ tọa độ của robot) và góc lái θ . Các giá trị α là trọng số của mỗi thành phần trong hàm mục tiêu.

Tuy nhiên, các trọng số được chọn hoạt động tốt khi robot gần đối mặt với mục tiêu (trong phạm vi $\pm 60^\circ$), nhưng hoạt động kém khi nó quay mặt về hướng ngược lại. Trong những trường hợp như vậy, chúng ta khuyến khích robot quay lại để đối mặt với mục tiêu bằng cách tăng trọng số α_3 của thành phần $\text{head}(\theta)$, tức là thay α_3 bằng:

$$\alpha_3 \left(1 + \alpha_4 \left(\frac{\theta_g}{\pi}\right)^2\right)$$

Bằng việc chia không gian thành các khoảng với khoảng cách D_{limit} là hằng số, mỗi khoảng cong cung cấp một cặp bất đẳng thức tuyến tính trong không gian. Các

ràng buộc được mô tả trong phần trước cũng là các bất đẳng thức tuyến tính. Do đó, với một tập hợp các bất đẳng thức tuyến tính và hàm mục tiêu tuyến tính, rất dễ để tìm giá trị lớn nhất của hàm mục tiêu.

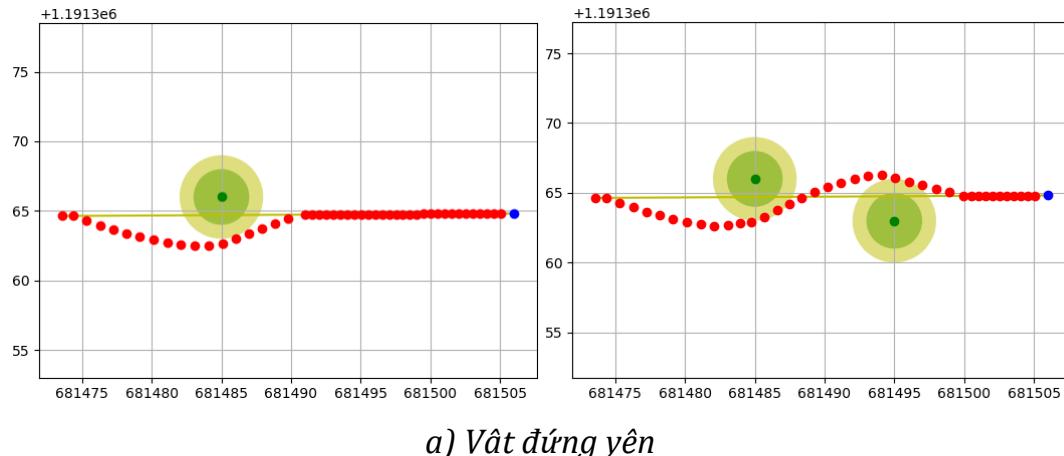
2.2.4.3. Ràng buộc về làn đường

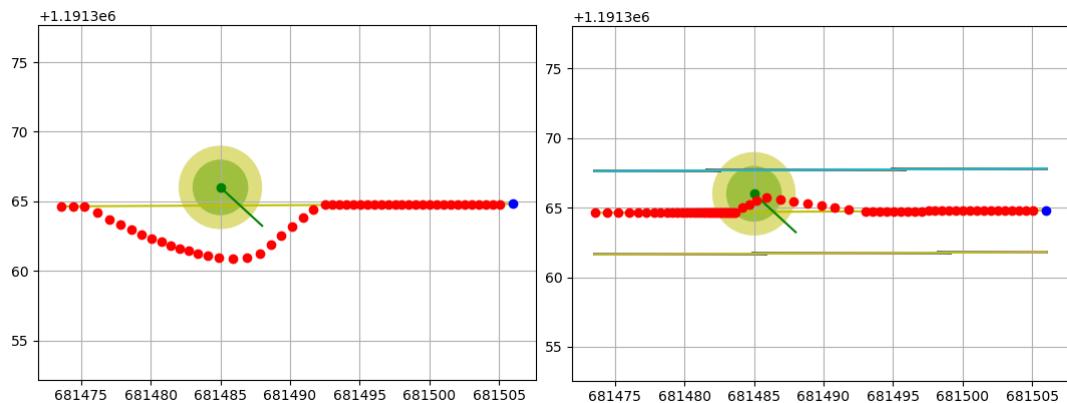
Trong quá trình tránh vật cản, ta cần đảm bảo rằng mô hình không đi ra khỏi phần đường cho phép. Gọi $D_{lane}(\theta)$ là độ dài đoạn đường robot có thể đi được theo hướng θ trước khi ra khỏi phần đường này. Khi đó:

$$D_{limit}(\theta, OBS) = \min (L, D(\theta, OBS), D_{lane}(\theta))$$

Để đơn giản, tại mỗi thời điểm, ta xem như robot đang di chuyển trên một đường thẳng Δ nối điểm gần nhất trên bản đồ và điểm tiếp theo của nó. Phần đường cho phép được định nghĩa là vùng nằm giữa hai đường thẳng biên, song song và cách đường thẳng Δ một khoảng cho trước.

Một số kết quả mô phỏng:





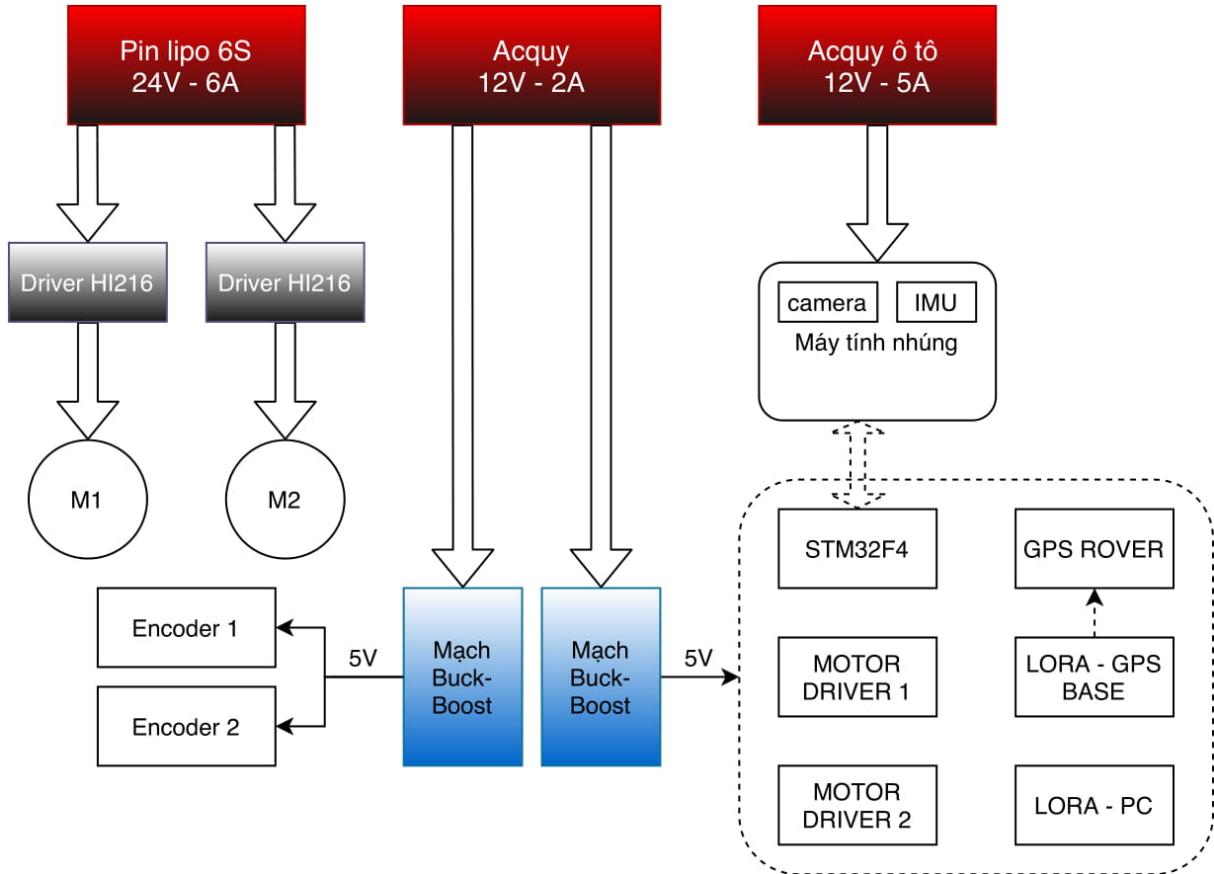
c) *Ánh trái: Không có đường biên, robot lách sang phải để vượt qua vật cản có vận tốc nhỏ. Ánh phải: Có đường biên, robot chờ vật cản đi qua rồi lách sang trái để đảm bảo vẫn đi trong phần đường cho phép.*

Hình 2.25 Một số kết quả mô phỏng thuật toán tránh vật cản

Chương 3. THIẾT KẾ VÀ THI CÔNG PHẦN CỨNG

3.1. Sơ đồ kết nối phần cứng

3.1.1. Sơ đồ khái niệm công suất



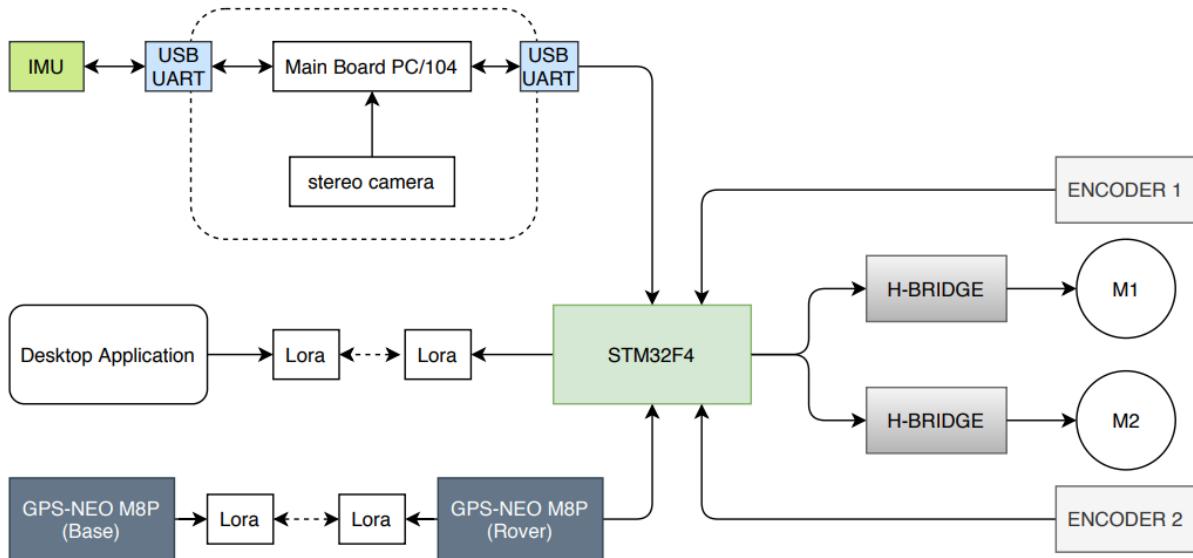
Hình 3.1 Sơ đồ kết nối các khái niệm nguồn, cung cấp công suất cho hệ robot

Toàn bộ năng lượng của robot được cấp từ ba nguồn chính:

- Pin LiPo 6S: là nguồn công suất độc lập được dành riêng cho driver cầu H điều khiển động cơ với dòng ra liên tục lớn, có thể chịu được đến hơn 5A phù hợp cho điều khiển 2 động cơ. Vì mỗi động cơ có thể kéo dòng tối đa 3A ở tốc độ và moment cao nhất, ngoài ra điện áp LiPo Battery ở mức 24V ~ 25V tối đa, phù hợp với điện áp điều khiển động cơ là 24V.
- Acquy 12V – 7.5Ah: với đặc tính dung lượng lớn, chịu tải bền bỉ và dòng ra liên tục từ 1 đến 2A, acquy 12V – 7.5Ah phù hợp để làm nguồn cấp cho toàn

- bộ mạch điều khiển, acquy được qua các mạch buck boost để giảm áp còn 5V ổn định. Một mạch được dùng riêng cho 2 encoder của 2 động cơ để đảm bảo tính ổn định, xung encoder đọc về chính xác. Mạch buck boost còn lại được cấp cho toàn bộ mạch điều khiển gồm: STM32F407, module GPS TINY RTK, các module lora RF, và tín hiệu điều khiển động cơ cho mạch driver,
- Acquy 12V – 20Ah: với mainboard máy tính nhúng và các module bên trong nó, yêu cầu dòng liên tục để hoạt động vào khoảng 5A, đồng thời phải có dung lượng đủ lớn để hoạt động được vài giờ, nên acquy 12V – 20Ah là lựa chọn phù hợp. Acquy cấp nguồn cho cả hệ máy tính nhúng gồm Mainboard, stereo camera BumbleBee2, IMU, ...

3.1.2. Sơ đồ kết nối các thành phần



Hình 3.2 Components Connection Diagram

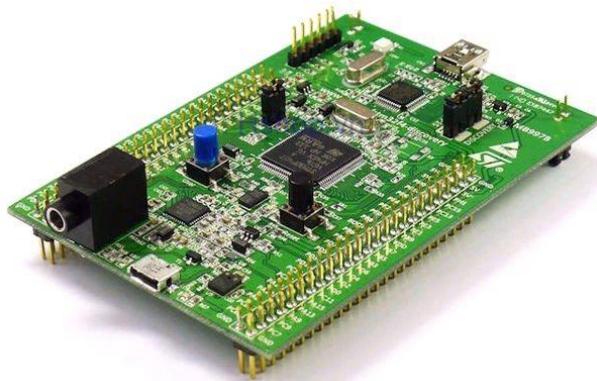
Sơ đồ kết nối phần cứng với trung tâm là vi điều khiển STM32F407, cũng có thể xem sơ đồ này đại diện cho sự truyền thông của cả hệ thống, bao gồm:

- Kết nối với máy tính nhúng thông qua một USB UART để có thể truyền/nhận dữ liệu và lệnh điều khiển tránh vật cản.
- Giao tiếp với giao diện thông qua một cặp module LORA.

- Cặp LORA giữa GPS base và GPS rover để trạm base truyền cho trạm rover các bản tin RTCM3, từ đó rover sẽ giải ra Code Phases Bias để có thể vào fix mode, đạt độ chính xác tối đa cho robot. Rover sẽ gửi bản tin NMEA cho vi điều khiển để nó biết được tọa độ của mình và các thông tin cần thiết khác.
- Bước cuối cùng của điều khiển là điều khiển vận tốc 2 động cơ, vì vậy cần thiết có dữ liệu feedback từ encoder cho vi điều khiển.

3.2. Tổng quan các thiết bị

3.2.1. Vi xử lý STM32F407



Hình 3.3 Board STM32F407 Discovery

Kit STM32F4DISCOVERY thúc đẩy khả năng của vi điều khiển hiệu suất cao STM32F407, cho phép người dùng dễ dàng phát triển các ứng dụng. Kit bao gồm một bộ công cụ gỡ lỗi nhúng ST-LINK, một giao tốc số ST-MEMS, một micro số, một audio DAC với trình điều khiển loa tích hợp lớp D, đèn LED, nút bấm và cổng micro-AB USB OTG.

Các đặc điểm tính năng chính

- Bộ vi xử lý STM32F407VGT6: 32-bit ARM Cortex – M4 với lõi FPU, bộ nhớ Flash 1MB, RAM 192KB với kiểu chân LQFP100.
- Trình gỡ lỗi ST-LINK/V2 trên STM32F4DISCOVERY.
- USB ST-LINK với khả năng kiểm tra lại và ba giao diện khác nhau:

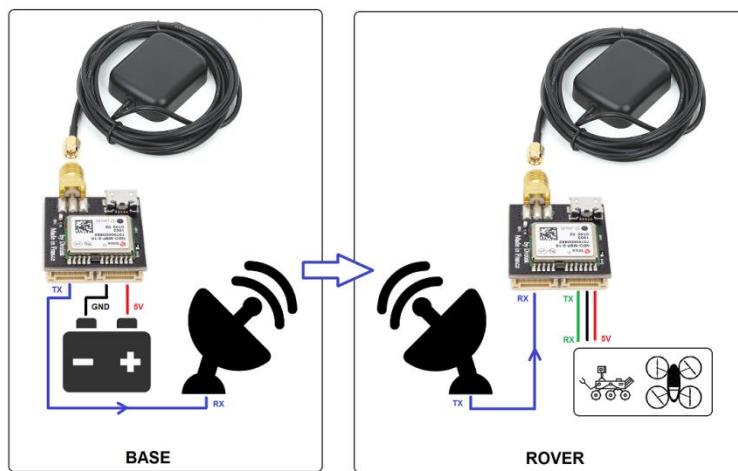
- Cổng gõ lõi.
- Cổng COM ảo.
- Lưu trữ khối lượng.
- Nguồn cung cấp: thông qua USB hoặc nguồn điện bên ngoài 3V – 5V.
- Gia tốc kế 3 trục LIS302DL hoặc LIS3DSH ST MEMS.
- Micro số đa hướng dùng cảm biến âm thanh MP45DT02 ST-MEMS.
- CS43L22 bộ chuyển đổi âm thanh/kỹ thuật số âm thanh công suất thấp (DAC) với trình điều khiển loa tích hợp lớp D.
- Tám đèn LED:
 - LD1 (đỏ/xanh) cho giao tiếp USB.
 - LD2 (đỏ) led báo nguồn 3.3V.
 - Bốn led của người dùng, LD3 (màu cam), LD4 (xanh lục), LD5 (màu đỏ) và LD6 (màu xanh lam).
 - 2 đèn led OTG USB LD7 (xanh lá cây) VBUS và LD8 (màu đỏ) quá dòng điện.
- Hai nút nhấn (người dùng và reset).
- USB OTG FS có cổng kết nối micro-AB.
- Header mở rộng I/O cho LQPF64 để kết nối nhanh với bo mạch gốc và dễ dàng sử dụng.

Bảng 3.1 Thông số kỹ thuật vi xử lý STM32F407

Thông số kỹ thuật	Mô tả
Nguồn cấp	3V – 5V DC
Bộ nhớ	1MB Flash, 192 KB RAM
Hỗ trợ mạch nạp và debug	ST – LINK/V2
Tần số xung Clock	168MHz

3.2.2. Module định vị GPS NEO – M8P

3.2.2.1. Module GPS NEO – M8P



Hình 3.4 Mô hình RTK GPS NEO – M8P

Module GPS NEO-M8P gồm hai bộ: trạm cơ sở (Base) và máy di động (Rover), với kích thước nhỏ gọn và độ chính xác cao (2.5cm), module phù hợp với các ứng dụng định vị thời gian thực như thiết bị tự hành.

Trạm cơ sở có công dụng tính toán ra một số nguyên giá trị N, hay còn được gọi là số gia cải chính thông qua việc thu định vị vệ tinh nhân tạo.

Số gia cải chính này sẽ được gửi tới các máy di động để hiệu chỉnh vị trí nhằm đạt được độ chính xác cao. Trong đó, thiết bị dùng để gửi số cải chính này cho các máy di động thường là tín hiệu dạng sóng vô tuyến UHF (Radio).

Bảng 3.2 Thông số kỹ thuật của Module GPS

Thông số kỹ thuật	Mô tả
Nguồn cấp	3.3V tới 5V
Bộ nhận	72 kênh GPS L1, GLONASS L1, BeiDou B1, GALILEO E1
Tần số cập nhật tối đa	8 Hz

Độ chính xác tối đa	2.5 cm
Công suất tiêu thụ	75 mW
Nhiệt độ hoạt động ổn định	-20°C tới 60°C
Data link	Bất kì thiết bị có chuẩn giao tiếp tương thích UART: Bluetooth, 433-868-915 MHz radio link, Xbee...
Chuẩn kết nối	SMA, USB, JST-GH, I2C
Kích thước	23.5x24.5 mm

3.2.2.2. GPS Antenna



Hình 3.5 GPS Antenna

Antenna TW2410 xuất xứ từ Tallyman, được dùng để thu tín hiệu định vị từ các vệ tinh nhân tạo như GPS L1, GLONASS L1, SBAS với dải tần số từ 1574 MHz tới 1606 MHz. Được thiết kế đặc biệt cho các ứng dụng công nghiệp, nông nghiệp hay quân sự (OEM).

Bảng 3.3 Thông số kỹ thuật của GPS Antenna

Thông số kỹ thuật	Mô tả
Nguồn cấp	2.5 V tới 16 V
Dòng sử dụng	15 mA
Độ lợi	4.25 dBi
Dải tần số	1.574 GHz tới 1.606 GHz
Chuẩn bảo vệ	IP67

Kết nối	SMA Male
Độ dày	15 mm
Độ dài cáp	5 m

3.2.3. Module cảm biến gia tốc và góc quay IMU



Hình 3.6 IMU sử dụng cảm biến MEMS ADIS16488

Module sử dụng cảm biến MEMS ADIS16488 là một hệ thống cảm biến quán tính hoàn chỉnh trong đó bao gồm 3 trục gyroscope (con quay hồi chuyển), 3 trục accelerometer (gia tốc kế), 3 trục đo magnetometer (từ trường kế) và một cảm biến đo áp lực (pressure sensor).

- Gyroscope: Tầm đo $450^\circ/\text{s}$, Bias $6.250/\text{h}$.
- Accelerometer: Tầm đo $\pm 18\text{g}$, Bias 3.5mg .
- Magnetometer: Tầm đo ± 2.5 gauss.

Bảng 3.4 Thông số kỹ thuật của Module IMU

Thông số kỹ thuật	Mô tả
Nguồn cấp	5VDC, 0.4A
Gyroscope	Tầm đo $450^\circ/\text{s}$, Bias $6.25^\circ/\text{h}$
Accelerometer	Tầm đo 8g , Bias 3.5mg
Sai số góc nghiêng tĩnh	$0.1^\circ(\text{RMS})$

Sai số góc nghiêng động	0.2° (RMS)
Sai số hướng tĩnh	0.1° (RMS)
Sai số hướng động	0.3° (RMS)
Độ phân giải đo tốc độ	0.001°/s
Độ phân giải góc	0.001°/s
Dải đo vị trí 3 trục:	(Liên tục)
+ Roll	-180° tới 180°
+ Pitch	-90° tới 90°
+ Yaw	-180° tới 180°
Hoạt động đảm bảo ở dải tần số	0 tới 5Hz
Hoạt động đảm bảo ở dải nhiệt độ	20°C tới 85°C
Tần số cập nhật ngoã ra tối đa	500 Hz
Chuẩn truyền thông RS232	115.2 tới 921.6 Kbps
Thời gian khởi động	30s
Độ trễ tín hiệu	< 1ms
Kích thước	64x46x3mm

Dữ liệu từ Module IMU được xử lí và truyền thông qua chuẩn truyền thông USART tới vi xử lí.

Bảng 3.5 Frame truyền dữ liệu của IMU

0x0A	roll	pitch	yaw	ω_x	ω_y	ω_z	ax	ax	az	0x0D
1byte	7bytes	7bytes	7bytes	7bytes	7bytes	7bytes	6bytes	6bytes	6bytes	1byte

- Mỗi giá trị cách nhau bởi ký tự khoảng trắng (0x20)
- Tổng số byte truyền: 71 (không từ trường), 89 (có từ trường)
- Tổng số byte truyền có thể thay đổi theo yêu cầu lựa chọn các giá trị gửi lên.

3.2.4. Module thu phát sóng RF Lora 433MHz



Hình 3.7 Module truyền nhận không dây RF LoRa 433MHz

Module Lora SX1278 433MHz sử dụng chip SX1278 giao tiếp chuẩn Lora, với hai yếu tố quan trọng đó là tiết kiệm năng lượng và khoảng cách phát siêu xa (Ultimate long range wireless solution), ngoài ra nó còn có khả năng cấu hình để tạo thành mạng nên hiện tại được phát triển và sử dụng rất nhiều trong các nghiên cứu về IoT.

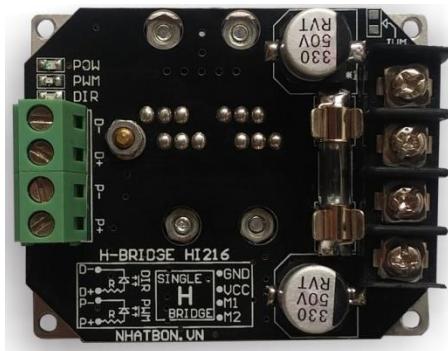
Module Lora SX1278 có khả năng thu phát với khoảng cách 3000m. Được tích hợp chuyển đổi giao tiếp chuẩn SPI của SX1278 sang UART giúp cho việc giao tiếp với vi xử lí dễ dàng hơn.

Bảng 3.6 Thông số kỹ thuật Module thu phát RF LoRa

Thông số kỹ thuật	Mô tả
Điện áp hoạt động	2.3V tới 5.5V
Điện áp giao tiếp	Chuẩn TTL 3.3V
Giao tiếp UART	Baudrate tối đa 115200, data 8 bits, 1 stop bit, none parity
Tần số	410 – 441 MHz
Công suất tối đa	20 dbm (100mW)
Khoảng cách truyền tối đa trong điều kiện lý tưởng	3000m

Tốc độ truyền	0.3Kbps tới 19.2 Kbps
Buffer	512 Bytes
Số lượng địa chỉ	65536
Kích thước	21x36 mm

3.2.5. Mạch cầu H



Hình 3.8 Driver điều khiển động cơ sử dụng mạch cầu H HI216

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, Module này chuyên dùng cho điều khiển tốc độ động cơ công suất lớn, điều khiển vị trí, vận tốc DC Motor, có thể lựa chọn mức kích âm hoặc dương để dễ dàng điều khiển.

Board kết hợp hiệu quả với chân xuất xung PWM trên vi xử lí giúp việc điều khiển tốc độ và vị trí động cơ được tối ưu và phù hợp với giải thuật điều khiển.

Bảng 3.7 Thông số kỹ thuật Mạch cầu H HI216

Thông số kỹ thuật	Mô tả
Điện áp kích	3.3V tới 5.5V
Điện áp công suất	12V tới 48V
Dòng liên tục	15 A
Dòng đỉnh	20 A

Công suất tối đa	600 W
Tần số hoạt động tối đa	100 KHz
Độ rộng xung tối đa	100%
Kích thước	52x64x22 mm

3.2.6. Máy tính nhúng



Hình 3.9 Nano – HR650 – R11

Bảng 3.8 Thông số kỹ thuật của máy tính nhúng

Thông số kỹ thuật	Mô tả
CPU	Socket G2 cho Intel® Core™ i7/i5/i3 thế hệ 2 và vi xử lý Celeron® mobile
BIOS	UEFI BIOS
Bộ nhớ hệ thống	Hỗ trợ 1066/1333MHz DDR3/SO-DIMMs 204 chân (tối đa 8GB hệ thống)
Ethernet	Mạng LAN kép: bộ điều khiển

	Realtek RTL8111E PCIe GbE hỗ trợ ASF 2.0
I/O	1 cổng 6 chân cho KB/MS 1 cổng RS-422/485 2 cổng RS-232 2 cổng SATA 6Gb/s với đầu nối nguồn 5V SATA 8 cổng USB 2.0 I/O số 8 bit (4 bit ngõ vào / 4 bit ngõ ra)
Hiển thị	Hỗ trợ hiển thị kép 1 cổng VGA (lên đến 2048 x 1536 @75Hz) 1 cổng HDMI (lên đến 1920 x 1200 @60Hz) 1 cổng 18/24-bit dual-channel LVDS (lên đến 1920 x 1200 @60Hz)
Mở rộng	1 x PCIe Mini card slot 1 x PCI-104 slot
Nguồn cung cấp	12V, hỗ trợ AT/ATX
Năng lượng tiêu thụ	12V@4.85A (2.6GHz Intel® Core i5-2540M với bộ nhớ 1333MHz 4GB DDR3)

3.2.7. Stereo camera Bumblebee2



Hình 3.10 BumbleBee2 stereo camera

Hệ thống stereo camera Bumblebee2 chứa hai cảm biến quét liên tục CCD (Charged Coupled Device) 1/3 inch, có thể truyền hình ảnh trái và phải đến PC thông qua giao diện IEEE-1394. Hệ thống bao gồm bộ công cụ phát triển phần

mềm Digiclops và Triclops, cho phép người dùng cài đặt camera, điều chỉnh chất lượng hình ảnh và truy cập thông tin về độ sâu trong thời gian thực bằng công nghệ stereo – vision.

Bảng 3.9 Thông số kỹ thuật của BumbleBee2 stereo camera

Thông số kỹ thuật	Mô tả
Cảm biến	2CCDs 1/3 inch, trắng đen
Độ phân giải	640 x 480
Tốc độ tối đa	48 FPS
Tiêu cự ống kính	2.5mm
Tầm nhìn ngang	100° HFOV

3.3. Thi công phần cứng

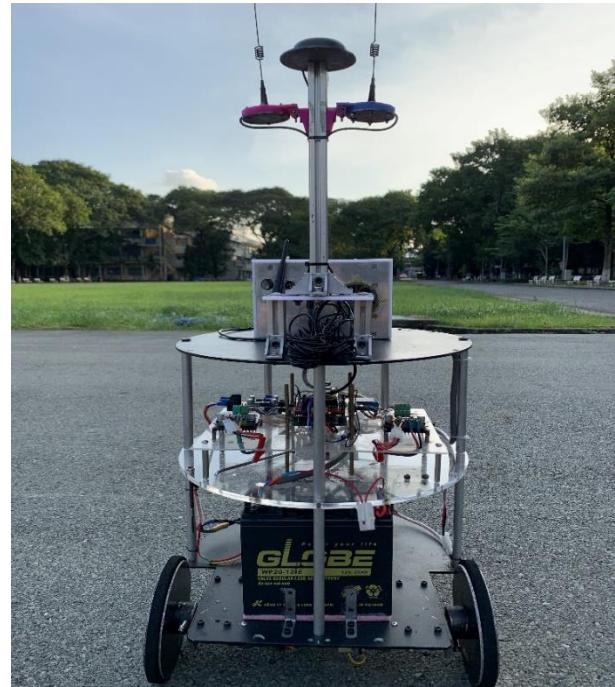
3.3.1. Mô hình robot



a) Mặt trước của mô hình



b) Mặt bên trái của mô hình

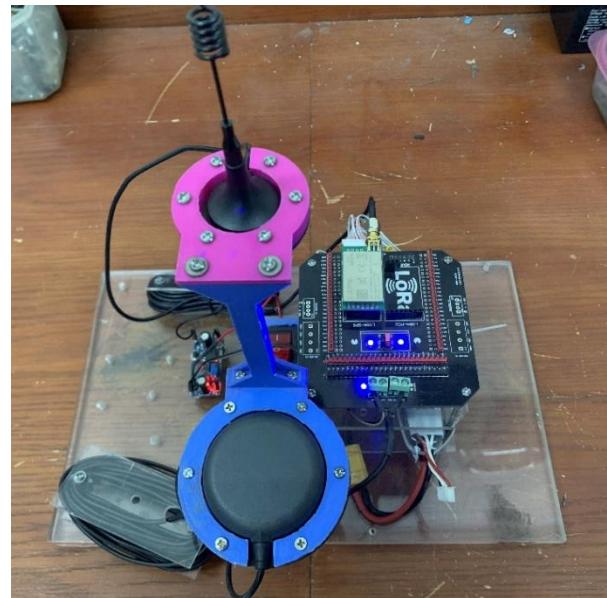


c) Mặt bên phải của mô hình

d) Mặt sau của mô hình

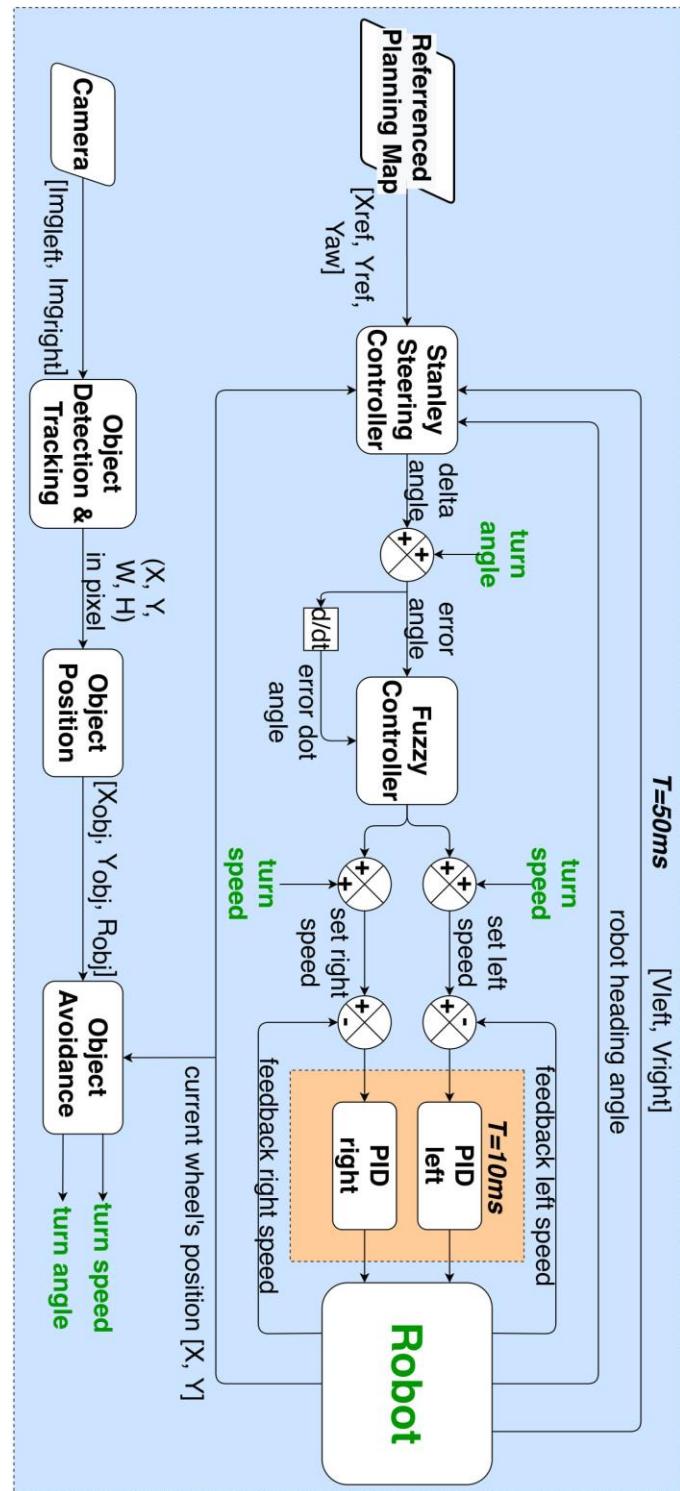
Hình 3.11 Mô hình robot

3.3.1. Trạm Base GPS



Hình 3.12 Mô hình trạm Base GPS

Chương 4. THIẾT KẾ HỆ THỐNG



Hình 4.1 Sơ đồ khái niệm giải thuật điều khiển cho hệ robot

4.1. Giải thuật điều khiển

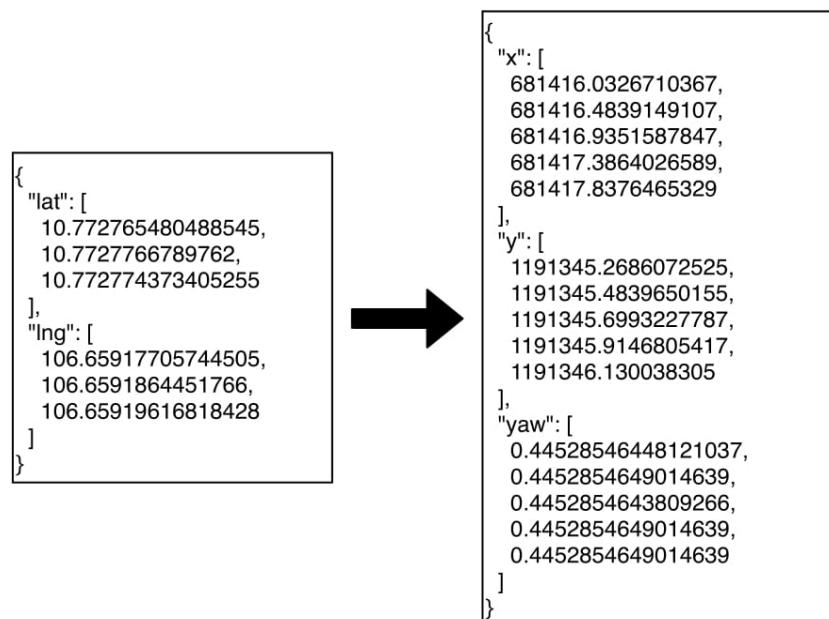
4.1.1. Khối Referenced Planning Map

Là một bộ dữ liệu đặc trưng cho quỹ đạo mà chúng ta muốn tham chiếu đến để robot điều khiển bám. Bộ dữ liệu đường đi được xây dựng như sau:

- Input: $[Latitude, Longitude]$ ở Decimal Degree Format.
- Output: $[X_{ref}, Y_{ref}, \theta_e]$ với X_{ref}, Y_{ref} đo trong hệ tọa độ UTM, θ_e đơn vị là radian.

Trong đó θ_e thực chất là góc tiếp tuyến của quỹ đạo tại mỗi điểm (góc so với trục Ox trong hệ tọa độ UTM), được tính toán trực tiếp từ X_{ref}, Y_{ref} .

Chúng ta cần đầu vào là một tập hợp các tọa độ $[Latitude, Longitude]$ được lấy trực tiếp từ Google Map APIs trên giao diện điều khiển ở định dạng decimal degree. Các điểm này sẽ được thuật toán cubic spline tính toán để “làm trơn” quỹ đạo, biến quỹ đạo trở nên giống đồ thị của một hàm số bậc ba. Đồng thời các tọa độ được “rải” thêm với khoảng cách xấp xỉ 0.5m thì có một điểm. Nhờ vậy, quỹ đạo được chia thành nhiều điểm nhỏ, và góc tiếp tuyến quỹ đạo không thay đổi một cách đột ngột, giúp robot bám quỹ đạo tốt hơn.



Hình 4.2 Dữ liệu mẫu thuật toán Cubic Spline

4.1.2. Khối Stanley Steering Controller

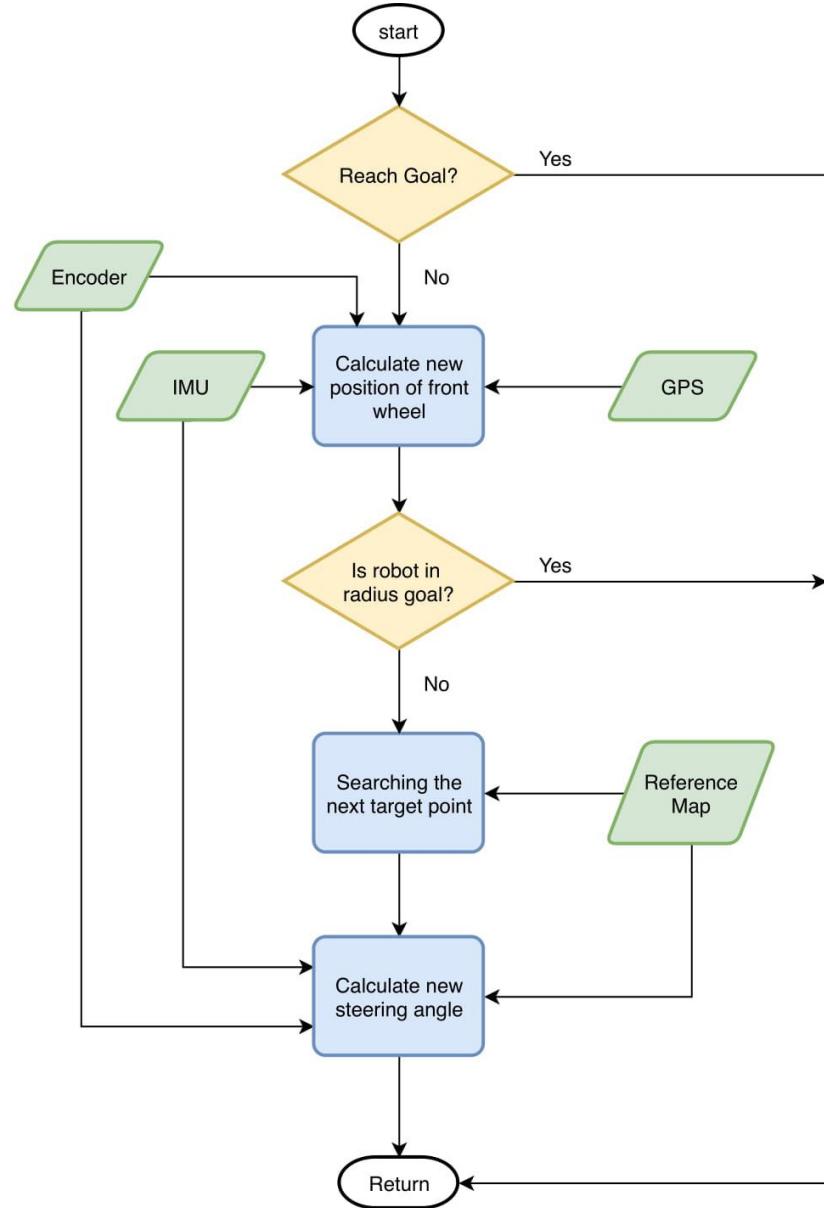
Đại diện cho bộ điều khiển bám quỹ đạo dựa trên giải thuật Stanley. Tần số điều khiển 20Hz.

- Input:

- $\{X_{ref}(j), Y_{ref}(j), \theta_{ref}(j)\}$ là tọa độ điểm đang tham chiếu đến và góc tiếp tuyến tại đó, lấy từ planning map.
- $\{X(t), Y(t), \theta(t)\}$ lần lượt là tọa độ hiện tại của xe lấy từ cảm biến GPS và góc heading của xe tính toán từ cảm biến IMU.
- $\{V_{left}(t), V_{right}(t)\}$ là vận tốc bánh trái và bánh phải, được cập nhật từ cảm biến Encoder. Được sử dụng để tính vận tốc dài $v(t)$ của robot.

- Output: $\delta(t)$ là góc lái của robot cần phải queo.

Sơ đồ giải thuật:



Hình 4.3 Giải thuật tính góc lái Stanley

Bước 1: Kiểm tra xem xe đã tới điểm mục tiêu (điểm cuối cùng của quỹ đạo) hay chưa. Nếu rồi thì trở về, không cần tính toán gì thêm. Nếu chưa thì tính tiếp bước sau.

Bước 2: Tính toán vị trí hiện tại của bánh trước $\{X_{wheel}(t), Y_{wheel}(t)\}$ vì thuật toán Stanley dùng trực bánh trước để làm tham chiếu cho xe. Vị trí này cần được

cập nhật liên tục sau mỗi lần lấy mẫu và yêu cầu độ chính xác cao, tuy nhiên tần số cập nhật của GPS là 1Hz nhưng tần số điều khiển Stanley là 20Hz, vì vậy chúng ta cần tính toán lại vị trí ở những chu kỳ không có dữ liệu mới từ GPS, độ sai lệch vị trí trong mỗi chu kỳ:

$$\Delta x = V_x \times \Delta T = V_{linear} \times \cos(\theta(t)) \times \Delta T$$

$$\Delta y = V_y \times \Delta T = V_{linear} \times \sin(\theta(t)) \times \Delta T$$

Bước 3: Tính bán kính mục tiêu là khoảng cách giữa vị trí bánh trước và điểm cuối cùng của quỹ đạo. Nếu bán kính này $< 0.5m$ thì xác định đã hoàn thành quỹ đạo, dừng Robot. Nếu chưa đạt được điều kiện này thì tiếp tục thuật toán.

Bước 4: Xác định điểm tham chiếu tiếp theo dựa trên vị trí hiện tại của robot. Giả sử i là điểm trên quỹ đạo mà robot đang tham chiếu đến, ta tính các khoảng cách từ vị trí hiện tại của robot với 5 (search offset) điểm tiếp theo (từ $i \rightarrow i + 5$) và chọn điểm có khoảng cách ngắn nhất để làm tham chiếu để tính các bước tiếp theo. Giả sử là j .

Bước 5: Tính e_{fa} là hình chiếu của xe lên tiếp tuyến của quỹ đạo tại điểm tham chiếu.

$$e_{fa}(t) = -((X_{wheel}(t) - X_{ref}(j)) \times \cos(\theta(t)) + (Y_{wheel}(t) - Y_{ref}(j)) \times \sin(\theta(t)))$$

Tính θ_e là góc lệch giữa góc heading của robot và góc tiếp tuyến tại điểm tham chiếu, thành phần này giúp giữ góc lái của xe song song với quỹ đạo

$$\theta_e(t) = \theta(t) - \theta_{ref}(j)$$

Tính θ_d là góc lệch thêm vào so với θ_e để đẩy xe vào quỹ đạo thay vì chạy song song với nó. Trong đó K là hệ số khuếch đại cho e_{fa} , K lớn làm tăng góc lệch đưa robot vào quỹ đạo nhanh nhưng dễ gây vọt lố và kéo theo dao động. Còn k_{soft} là hệ số điều chỉnh được thêm vào mẫu số, cho phép khả năng điều khiển góc ở vận tốc thấp, vì khi vận tốc dần về 0 thì góc thay đổi là rất lớn, tuy nhiên k_{soft} lớn sẽ làm giảm góc lệch cần thiết làm cho robot chậm tiến vào quỹ đạo.

$$\theta_d(t) = -\arctan 2 \left(\frac{K \times e_{fa}(t)}{v(t) + k_{soft}} \right)$$

Tính δ là góc lái cần thiết để đưa mô hình đi vào quỹ đạo, δ nằm trong giới hạn điều khiển $[-\delta_{max}, \delta_{max}]$

$$\delta(t) = \begin{cases} \theta_e(t) + \theta_d(t), & \text{if } |\theta_e(t) + \theta_d(t)| < \delta_{max} \\ \delta_{max}, & \text{if } (\theta_e(t) + \theta_d(t)) \geq \delta_{max} \\ -\delta_{max}, & \text{if } (\theta_e(t) + \theta_d(t)) \leq -\delta_{max} \end{cases}$$

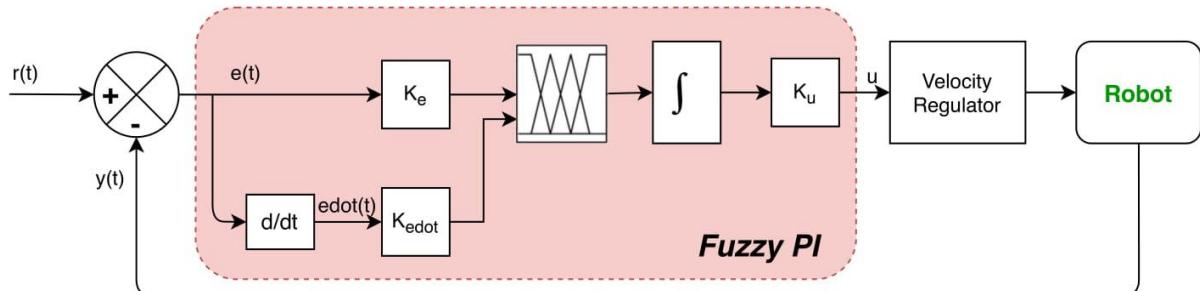
Trên thực tế, chọn $\delta_{max} = \pi$.

4.1.3. Khối Fuzzy Controller

Mục tiêu của khối Fuzzy Controller là tính toán ra giá trị vận tốc đặt cho các động cơ dựa vào góc lái. Từ góc δ được tính từ thuật toán Stanley và góc heading θ được cập nhật từ IMU, ta tính được góc đặt cho robot $r(t)$.

Đầu vào, ra cho cả khối:

- Input:
 - $r(t)$ góc đặt mà robot cần hướng tới
 - $\theta(t)$ góc heading hiện tại của robot
- Output: $\{V_{left}(t), V_{right}(t)\}$ lần lượt là vận tốc đặt cho bánh trái và bánh phải, để robot có thể xoay theo góc mong muốn.



Hình 4.4 Giải thuật cho bộ điều khiển Fuzzy

- Thiết kế bộ điều khiển Fuzzy PI

Đầu vào, ra cho bộ Fuzzy PI:

- Fuzzy Input:
 - $\{e, \dot{e}\}$ lần lượt sai số góc và tốc độ thay đổi sai số góc
- Fuzzy Output:
 - $\{u\}$ giá trị điều khiển u được sử dụng cho bộ thiết lập vận tốc động cơ M1, M2.

Bước 1: Tính toán các giá trị đầu vào

Tính sai số góc $e(t)$ dựa trên góc đặt và góc đọc về từ IMU, thực chất bằng $\delta(t)$

$$e(t) = r(t) - y(t) = \theta_d(t) - \theta(t) = \delta(t)$$

Tính độ thay đổi sai số góc $\dot{e}(t)$ bằng cách vi phân sai số góc

$$\dot{e}(t) = \dot{\theta}_d(t) - \dot{\theta}(t) = \dot{\delta}(t) = \frac{\delta(t) - \delta(t-1)}{\Delta T}$$

Bước 2: Chuẩn hóa các giá trị đầu vào trong miền $[-1, 1]$

Xác định tầm giá trị của $e(t)$: tầm giá trị của $e(t)$ cũng chính là tầm giá trị của $\delta(t)$

$$-\pi \leq \delta(t) \leq \pi$$

Chọn hệ số

$$K_e = \frac{1}{\pi} \rightarrow \bar{e}(t) = K_e \times e(t)$$

Xác định tầm giá trị của $\dot{e}(t)$: độ thay đổi sai số góc $\dot{e}(t)$ phụ thuộc vào tốc độ quay của động cơ, thực nghiệm cho thấy với mỗi chu kỳ lấy mẫu $\Delta T = 50 ms$

$$-\frac{\pi}{6} \leq \dot{e}(t) \leq \frac{\pi}{6}$$

Chọn hệ số

$$K_{\dot{e}} = \frac{6}{\pi} \rightarrow \dot{\bar{e}}(t) = K_{\dot{e}} \times \dot{e}(t)$$

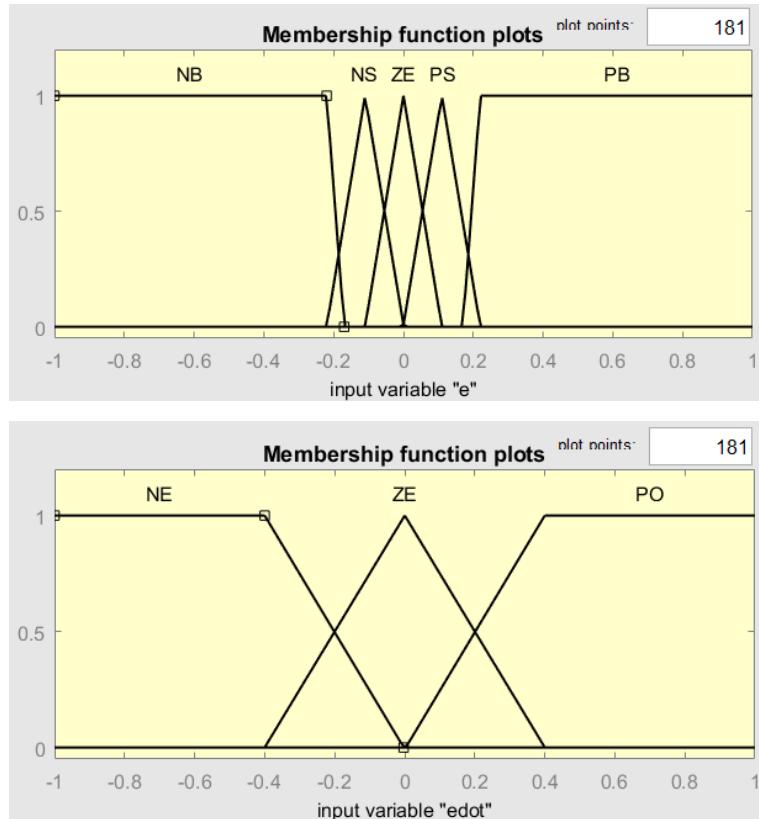
Bước 3: Mờ hóa

Ngõ vào bộ điều khiển PI mờ có 2 ngõ vào là e và \dot{e} .

- Chọn 5 giá trị ngôn ngữ cho biến e : NB, NS, ZE, PS, PB.

- Chọn 3 giá trị ngôn ngữ cho biến \dot{e} : NE, ZE, PO.

Định lượng giá trị 2 biến ngôn ngữ trên bằng tập mờ:



Hình 4.5 Tập mờ cho biến ngôn ngữ e và \dot{e}

Chọn miền giá trị cho các biến ngôn ngữ

- Biến vào e :

$$\begin{cases} \mu_{NB}(e) = \text{trapf}(-2, -1, -0.22, -0.17) \\ \mu_{NS}(e) = \text{trimf}(-0.22, -0.11, 0.001) \\ \mu_{ZE}(e) = \text{trimf}(-0.11, 0, 0.11) \\ \mu_{PS}(e) = \text{trimf}(0.001, 0.11, 0.22) \\ \mu_{PB}(e) = \text{trapf}(0.17, 0.22, 1, 2) \end{cases}$$

- Biến vào \dot{e} :

$$\begin{cases} \mu_{NE}(\dot{e}) = \text{trapf}(-2, -1, -0.22, -0.17) \\ \mu_{ZE}(\dot{e}) = \text{trimf}(-0.4, 0, 0.4) \\ \mu_{PO}(\dot{e}) = \text{trapf}(0.17, 0.22, 1, 2) \end{cases}$$

Các giá trị biến ngôn ngữ được chọn dựa trên kinh nghiệm điều khiển.

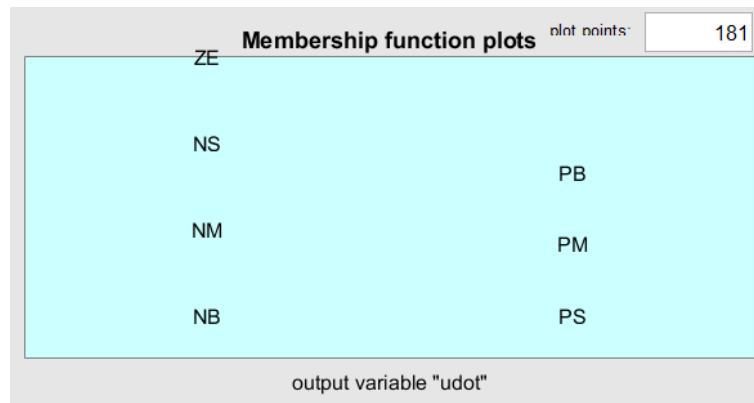
Các hàm thành viên của từng biến ngôn ngữ được chọn và điều chỉnh thông số dựa trên thực tế để có được kết quả theo ý muốn.

Bước 4: Giải mờ

Ngõ ra bộ điều khiển PI mờ Sugeno có 1 ngõ ra là \dot{u}

- Chọn 7 giá trị ngôn ngữ cho biến \dot{u} : NB, NM, NS, ZE, PS, PM, PB.

Định lượng giá trị biến ngôn ngữ trên bằng tập mờ:



Hình 4.6 Tập mờ cho biến ngõ ra \dot{u}

- Chọn các giá trị cho biến ra \dot{u} :

$$\begin{cases} \mu_{NB}(\dot{u}) = -0.95 \\ \mu_{NM}(\dot{u}) = -0.8 \\ \mu_{NS}(\dot{u}) = -0.4 \\ \mu_{ZE}(\dot{u}) = 0 \\ \mu_{PS}(\dot{u}) = 0.4 \\ \mu_{PM}(\dot{u}) = 0.8 \\ \mu_{PB}(\dot{u}) = 0.95 \end{cases}$$

$$r < \theta \Rightarrow \theta \downarrow$$

Bảng 4.1 Các quy tắc mờ

\dot{u}		e				
		NB	NS	ZE	PS	PB
\dot{e}	NE	NB	NM	NS	ZE	PS
	ZE	NM	NS	ZE	PS	PM
	PO	NS	ZE	PS	PM	PB

Sử dụng phương pháp suy diễn **Max-Min** và phương pháp giải mờ trung bình có trọng số để tính toán các thông số ngõ vào và ngõ ra.

$$\dot{u} = \frac{\sum_k y_k \beta_k}{\sum_k \beta_k}$$

Sau khi tính được giá trị ngõ ra bộ Fuzzy, ta có:

$$u(t) = u(t-1) + \dot{u} \times \Delta T$$

Giá trị $u(t)$ được cho qua bộ khởi tạo vận tốc, với $u(t) > 0$ xe quẹo phải và $u(t) < 0$ xe quẹo trái

Bước 5: Tính giá trị vận tốc đặt

Ngõ ra bộ điều khiển PI mờ được sử dụng cho việc thay đổi vận tốc 2 động cơ từ đó thay đổi góc xoay của mô hình tới góc đặt mong muốn.

Vận tốc động cơ thay đổi theo công thức:

huy SG

$$V_{left}(t) = \begin{cases} (1 + u(t)) \times V_{ld}, & \text{if } u(t) \geq 0 \\ (1 - u(t)) \times V_{ld}, & \text{if } u(t) < 0 \end{cases}$$

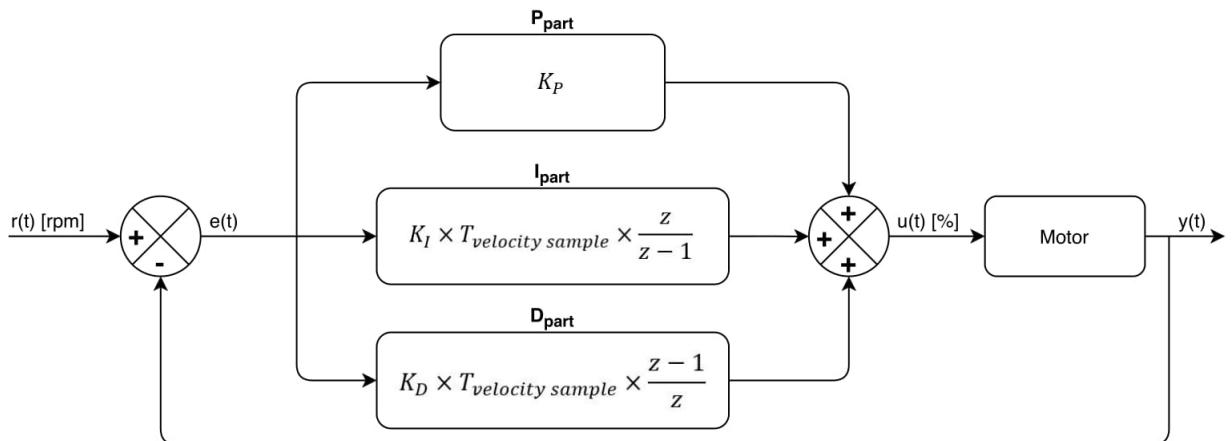
$$V_{right}(t) = \begin{cases} (1 - u(t)) \times V_{rd}, & \text{if } u(t) \geq 0 \\ (1 + u(t)) \times V_{rd}, & \text{if } u(t) < 0 \end{cases}$$

Với: V_{ld} là vận tốc đặt động cơ bên trái, V_{rd} là vận tốc đặt động cơ bên phải. Khi muốn mô hình xoay sang phải ($u(t) \geq 0$) cần tăng tốc độ động cơ bên trái và giảm tốc độ động cơ bên phải một lượng $u(t)$ được tính từ bộ điều khiển PI mờ.

Tương tự cho trường hợp xoay trái, giảm một lượng $|u(t)|$ ở động cơ bên trái và tăng tốc độ động bên phải. Khi góc lệch càng lớn, giá trị ngõ ra bộ PI càng lớn, tốc độ giữa 2 động cơ ngày càng chênh lệch từ đó để mô hình đáp ứng tới góc mong muốn càng nhanh và ổn định. Vì là bộ điều khiển PI nên độ vọt lỗ của hệ thống nhỏ và thời gian đáp ứng đạt yêu cầu khi vận tốc động cơ không quá lớn.

4.1.4. Khối PID Motor

Từ vận tốc đặt cho từng bánh ở bước trước, chúng ta cần điều khiển vận tốc động cơ sử dụng bộ điều khiển PID rời rạc, với thời gian lấy mẫu $T_{velocity sample} = 10 ms$.



Hình 4.7 Sơ đồ khối thiết kế bộ điều khiển PID

Cách điều chỉnh hệ số K_P, K_I, K_D

- Cho K_I và K_D bằng 0, tăng dần K_P cho tới khi hệ thống đạt giá trị xác lập và có thể có vọt lỗ.
- Tăng dần K_I với K_P ở bước trên. Tăng đến khi đạt tới điểm xác lập, có thể có vọt lỗ. Chọn K_D sao cho mất đi độ vọt lỗ.
- Tùy chỉnh K_P để có được thời gian đáp ứng mong muốn.

Bước 1: Xử lý tín hiệu Encoder, tính toán vận tốc từng bánh.

Vận tốc hiện tại của động cơ được tính dựa trên số xung trên vòng của encoder trong một chu kỳ lấy mẫu vận tốc, được tính theo công thức:

$$V(k) = \frac{(Encoder(k) - Encoder(k-1))}{(Encoder Per Resolution)} \times \frac{60}{T_{velocity sample}} [rpm]$$

Trong đó:

- $Encoder(k)$: Giá trị xung hiện tại được đọc từ Encoder của động cơ.
- $Encoder(k-1)$: Giá trị Encoder trước đó 1 chu kỳ.
- $Encoder Per Resolution$: Tổng số xung encoder trên 1 vòng quay động cơ.
- $T_{velocity sample}$: Thời gian lấy mẫu vận tốc.

Mô hình xe sử dụng đơn vị m/s cho việc cấu hình vận tốc, vì thế cần chuyển đổi đơn vị vận tốc m/s sang rpm để đồng bộ với bộ điều khiển PID bằng công thức:

$$V_{rpm} = \frac{V_{m/s} \times 60}{2\pi R}$$

Trong đó R là bán kính bánh xe.

Bước 2: Cập nhật luật điều khiển PID rời rạc, công thức luật điều khiển

$$\begin{aligned} P_{part} &= K_p(e(k) - e(k-1)) \\ I_{part} &= \frac{1}{2} K_I \times T_{velocity sample} \times (e(k) + e(k-1)) \\ D_{part} &= \frac{K_D}{T_{velocity sample}} \times (e(k) - 2e(k-1) + e(k-2)) \\ U(k) &= U(k-1) + P_{part} + I_{part} + D_{part} \end{aligned}$$

Ngõ ra $U(k)$ được cho qua khung bão hòa $(-100; 100)$ để tránh trường hợp vọt lố và vượt ngoài tầm giá trị của PWM.

Bước 3: Xuất xung điều khiển PWM

Vận tốc động cơ điều chỉnh dựa vào giá trị độ rộng xung PWM trong 1 chu kỳ với tần số phù hợp với động cơ đang sử dụng (thường là 20KHz)

$$PWM = \frac{U(k)}{100} \times N$$

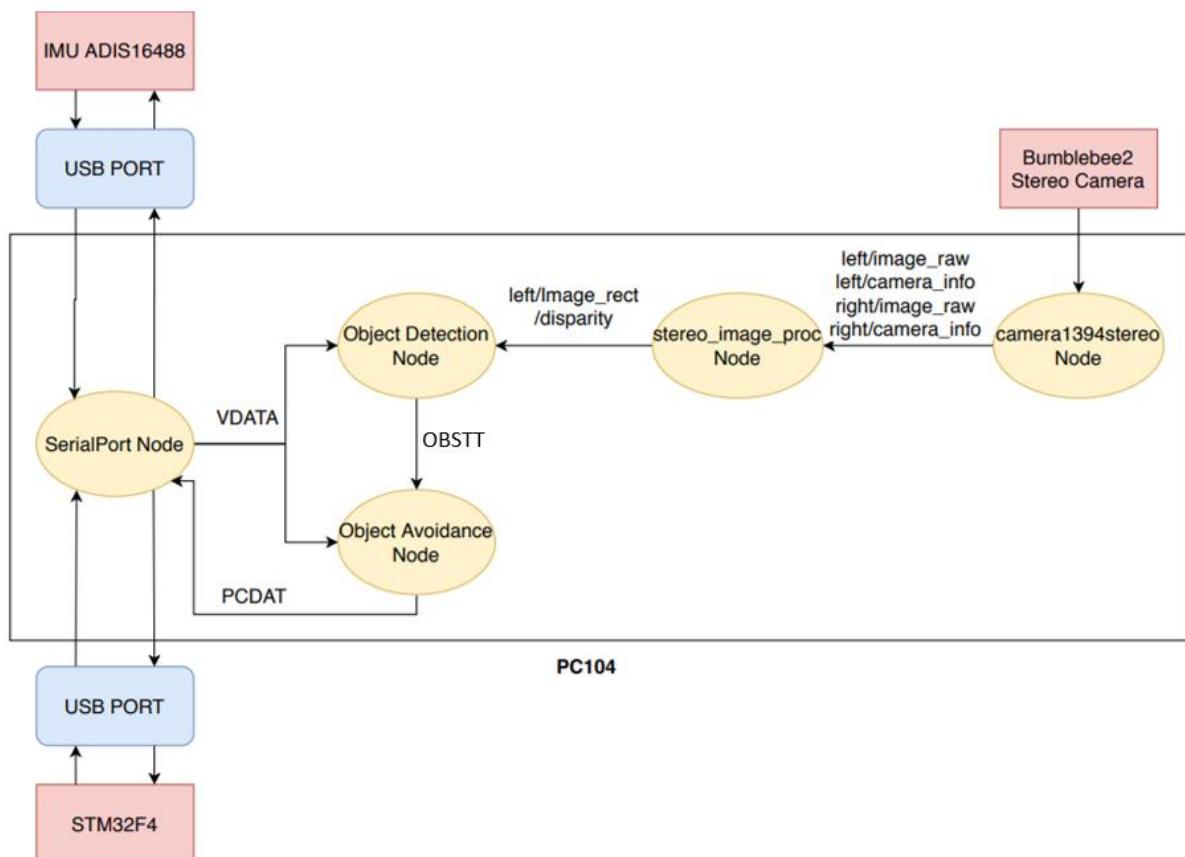
Với N là tổng số xung trong 1 chu kì xuất xung đầy PWM.

4.2. Hệ thống ROS node và giải thuật tránh vật cản

Giải thuật điều khiển mô hình tránh vật cản được thực hiện trên máy tính nhúng PC104 được cài đặt hệ điều hành Ubuntu 16.04. Với ngõ vào bao gồm ảnh trái và ảnh phải từ stereo camera, tọa độ hiện tại của mô hình (VDATA) từ module GPS và hướng của mô hình từ IMU, giải thuật có các ngõ ra (PCDAT) như sau:

- Cờ detected_flag: cờ quyết định khi nào mô hình cần ra khỏi quỹ đạo để tránh vật cản.
- Vận tốc v và góc lái θ : vận tốc v và góc lái θ để tránh vật cản.

Giải thuật được thực hiện trên nền tảng ROS với các node như sau:

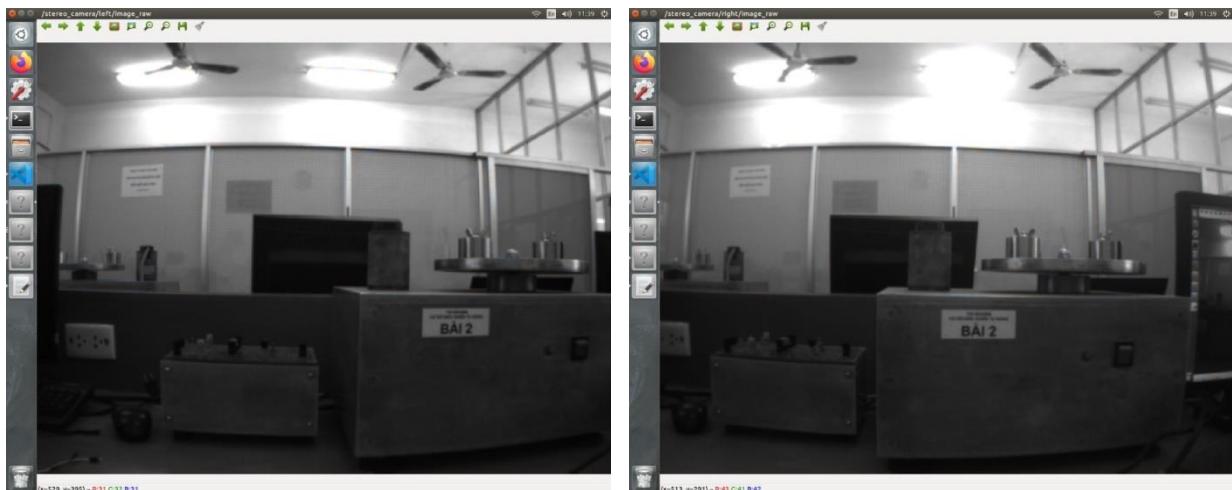


Hình 4.8 Sơ đồ các node

4.2.1. Camera1394stereo Node

Camera1394stereo node được tạo trong một package cùng tên có sẵn của ROS. Đây là phiên bản sửa đổi của ROS cho các thiết bị hỗ trợ giao thức IEEE 1394 Digital Camera IIDC (Instrumentation & Industrial Digital Camera) trong package camera1394. Package này cung cấp giao diện ROS cho các stereo camera dựa trên Firewire như Point Grey Bumblebee2.

Node Camera1394stereo hoạt động như một driver giúp ta đọc ảnh từ Bumblebee2 stereo camera và publish vào các topic stereo_camera/ left/ image_raw và stereo_camera/ right/ image_raw. Đồng thời, node Camera1394stereo cũng publish các topic stereo_camera/ left/ camera_info và stereo_camera/ right/ camera_info cần thiết cho node xử lý ảnh tiếp theo.

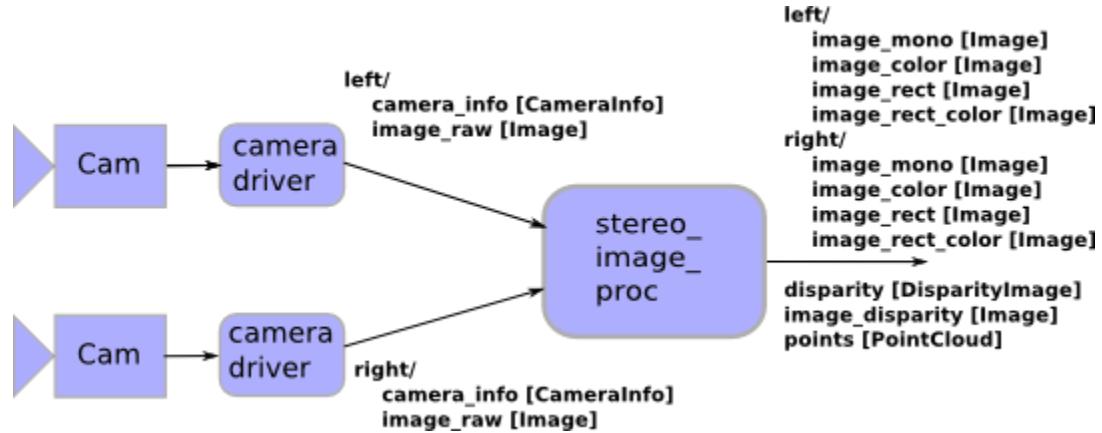


Hình 4.9 Ảnh camera trước khi hiệu chỉnh

4.2.2. Stereo_image_proc Node

Node Stereo_image_proc được tạo trong một package cùng tên có sẵn của ROS. Stereo_image_proc thực hiện các nhiệm vụ xử lý ảnh cho cả hai máy ảnh, bao gồm loại bỏ biến dạng ảnh khỏi hình ảnh thô và nếu cần sẽ chuyển đổi hình ảnh định dạng Bayer hoặc YUV422 thành ảnh màu. Lưu ý rằng đối với các stereo camera được hiệu chỉnh chính xác, loại bỏ biến dạng thực tế được kết hợp với hiệu chỉnh

và biến đổi hình ảnh. Stereo_image_proc cũng sẽ tính toán các ảnh disparity từ các cặp ảnh stereo bằng thuật toán Block Matching của OpenCV.



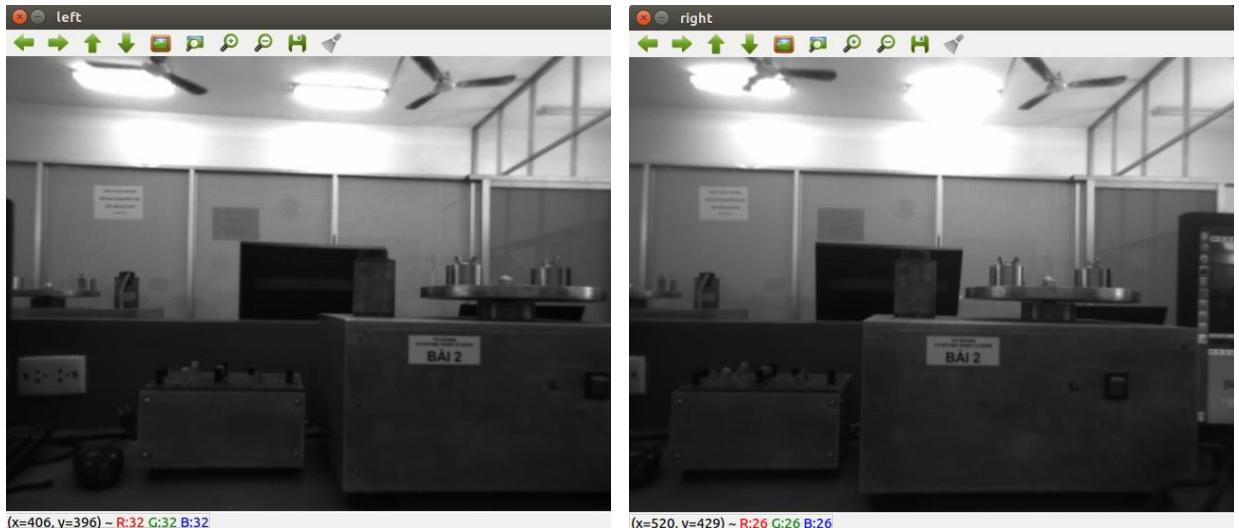
Hình 4.10 Stereo_image_proc node

Stereo_image_proc node được sử dụng để loại bỏ biến dạng, hiệu chỉnh ảnh thông đồng thời tính toán disparity map và publish vào các topic stereo_camera/ left/ image_rect, stereo_camera/ right/ image_rect và stereo_camera/ disparity để dùng cho các node tiếp theo. Trong quá trình tính toán disparity map, ta cần thay đổi một số thông số liên quan đến thuật toán Block Matching cho phù hợp, bao gồm:

- Pre – filtering: các thông số của bộ lọc trước, giúp chuẩn hóa độ sáng của hình ảnh và tăng cường kết cấu.
 - prefilter_size (int, mặc định: 9): kích thước cửa sổ chuẩn hóa, đơn vị pixels.
 - prefilter_cap (int, mặc định: 31): giới hạn các giá trị pixels được chuẩn hóa.
- Correlation: các thông số liên quan đến việc tìm kiếm các pixel tương ứng đọc theo các đường epipole nằm ngang bằng cửa sổ SAD.
 - correlation_window_size (int, mặc định: 15): Kích thước cạnh theo pixel của cửa sổ tương quan. Giá trị phải là số lẻ, trong phạm vi 5 đến

255, nhưng có hiệu suất tốt hơn với các giá trị lớn hơn 21. Các giá trị lớn hơn cho kết quả mượt mà hơn, nhưng làm mờ đi các đặc trưng nhỏ và sự không liên tục về chiều sâu.

- `min_disparity` (int, mặc định: 0): Độ dời tối thiểu cho cửa sổ tìm kiếm. Khi `min_disparity` bằng một giá trị dương, camera sẽ tìm thấy các vật thể gần hơn. Nếu các máy ảnh bị nghiêng về phía nhau, có thể đặt `min_disparity` bằng giá trị âm. Khi `min_disparity` lớn hơn 0, các đối tượng ở khoảng cách lớn sẽ không được tìm thấy.
 - `disparity_range` (int, mặc định: 64): Kích thước của cửa sổ tìm kiếm theo pixel. Cùng với `min_disparity`, `disparity_range` xác định khu vực máy ảnh có thể nhìn thấy.
- Post – filtering: các thông số của bộ lọc sau, loại bỏ các kết quả tương quan không mong muốn.
- `uniqueness_ratio` (double, mặc định: 15.0): Lọc kết quả dựa trên tương quan với giá trị tốt nhất tiếp theo dọc theo đường epipolar. Kết quả với `uniqueness_ratio > (best_match - next_match)/next_match` bị loại bỏ.
 - `texture_threshold` (int, mặc định: 10): Lọc kết quả dựa trên số lượng đặc trưng trong cửa sổ tương quan.
 - `speckle_size` (int, mặc định: 100): Lọc các vùng có ít hơn số pixel này. Ví dụ: `speckle_size = 100` có nghĩa là tất cả các vùng có ít hơn 100 pixels sẽ bị loại bỏ.
 - `speckle_range` (int, mặc định: 4): Nhóm các vùng khác nhau dựa trên sự kết nối của chúng. Disparity được nhóm lại với nhau trong cùng một khu vực nếu chúng nằm trong khoảng cách này tính bằng pixel.



Hình 4.11 Ảnh camera sau khi hiệu chỉnh

4.2.3. Object Detection Node

Object Detection node dùng để phát hiện và xác định vị trí, vận tốc của vật cản, với ngõ vào từ các topic sau:

- stereo_camera/left/image_rect: ảnh đã được hiệu chỉnh từ camera bên trái của Bumblebee2 stereo camera. Mô hình MobileNet SSD và bộ theo dõi MOSSE được thực hiện trên ảnh này để phát hiện, xác định vị trí và tốc độ của vật theo pixel.
- stereo_camera/disparity: bản đồ chênh lệch (disparity map), được tính từ stereo_image_proc node sử dụng thuật toán Block Matching của OpenCV. Bản đồ này giúp chuyển đổi vị trí, vận tốc của vật sang hệ tọa độ thực.
- VDATA: vị trí và hướng hiện tại của mô hình, được publish từ SerialPort node.

Và ngõ ra, được publish vào topic /OBSTT, là một mảng có kích thước $5 \times N$, với N là số vật cản phát hiện được. Mỗi dòng của mảng đại diện cho một vật cản với các giá trị:

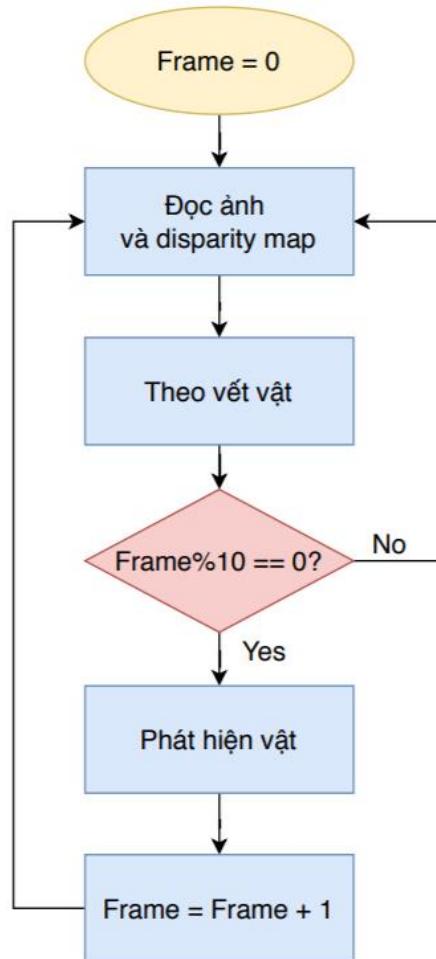
$$[x, y, w, v_x, v_y]$$

Trong đó:

- x, y : tọa độ thực của vật.
- w : kích thước ngang (m) của bounding box chứa vật.
- v_x, v_y : vận tốc (m/s) thực của vật.

Các giá trị được tính trong hệ tọa độ của mô hình.

Giải thuật phát hiện và xác định vị trí, tốc độ của vật cản như sau:



Hình 4.12 Giải thuật phát hiện, xác định vị trí và tốc độ vật cản

Bước 1: Đọc ảnh từ topic `stereo_camera/left/image_rect` và disparity map từ topic `stereo_camera/disparity`.

Bước 2: Cập nhật vị trí bounding box mới của các vật cản đã phát hiện sử dụng bộ theo dõi MOSSE. Do disparity map chứa các giá trị disparity cho cả đối tượng

lẫn nền và các giá trị -1 tại các speckles, nên ta không thể lấy tất cả giá trị disparity trong bounding box để tính độ sâu của vật. Để lấy được chính xác độ sâu từ disparity map, loại bỏ các giá trị nhiễu, với mỗi boundingbox mới được cập nhật:

- Lấy ra các giá trị disparity nằm trong bounding box từ disparity map.
- Sắp xếp tập các giá trị disparity trên theo thứ tự từ thấp đến cao và chia thành n khoảng bằng nhau. Ta sẽ dùng các giá trị trong khoảng chứa nhiều phần tử nhất để tính độ sâu của vật cản.

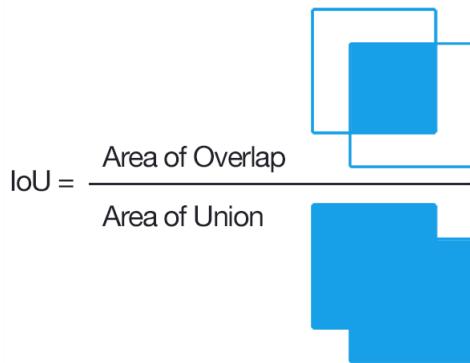
Từ độ sâu, áp dụng các công thức, ta tính được tọa độ thực của vật trong hệ tọa độ mô hình, sau đó chuyển sang hệ tọa độ UTM sử dụng các thông tin về mô hình trong VDATA. Vận tốc của vật được xác định bằng vận tốc trung bình của tâm bounding box chứa vật trong 20 khung hình gần nhất có sự xuất hiện của vật cản đó. Nếu không có vật cản nào được phát hiện, bỏ qua bước 2.

Bước 3: Tại khung hình đầu tiên và các khung hình thứ 10, 20, 30, ... mà Object Detection node đọc được, ta sử dụng mô hình MobileNet SSD để phát hiện vật cản. Ta chỉ giữ lại các bounding box có độ tin cậy lớn hơn 0.5 và thuộc các lớp:

Bảng 4.2 Các lớp đối tượng sử dụng trong đề tài

ID	Lớp
2	Xe đạp
6	Xe buýt
7	Ô tô
14	Xe máy
15	Người

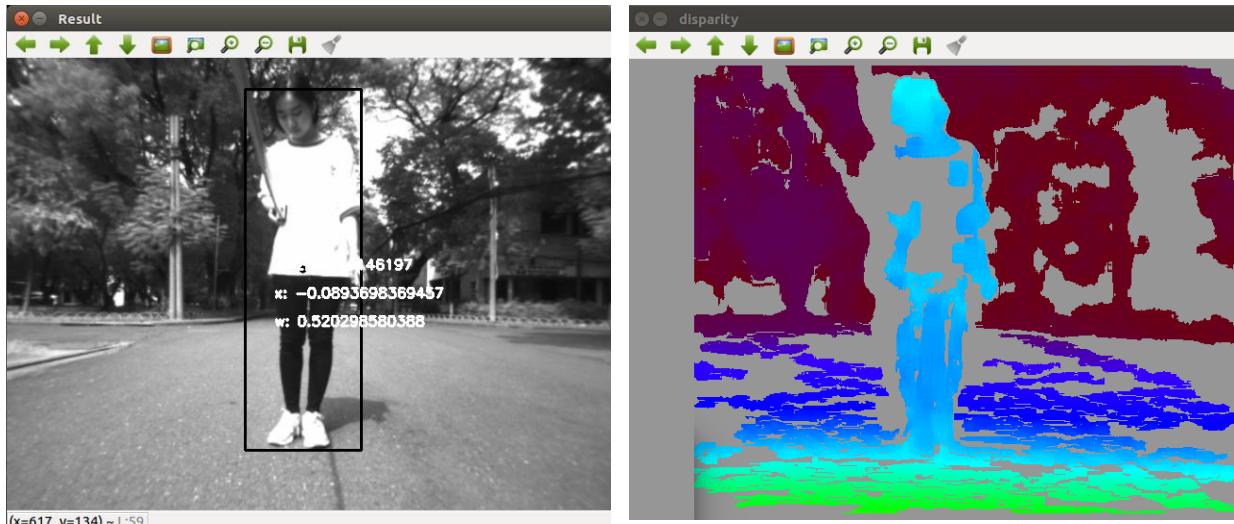
Với mỗi bounding box mới được phát hiện, ta sẽ tính IoU với các bounding box của khung hình trước đó. IoU được tính bằng:



Hình 4.13 Cách tính IoU

Một bounding box được xác định là chứa đối tượng cũ khi có IoU với ít nhất một bounding box cũ lớn hơn 0.5, lúc này bounding box cũ có IoU lớn nhất với bounding box mới đó sẽ được cập nhật vị trí, kích thước và hình ảnh mới. Ngược lại, một bounding box được xác định là chứa đối tượng mới khi không có IoU với bất kì bounding box cũ nào lớn hơn 0.5, và sẽ được thêm vào tập các bounding box của khung hình hiện tại.

Sau cùng, quay lại bước 1. Quy trình này lặp lại trong suốt quá trình vận hành của mô hình.



Hình 4.14 Ví dụ phát hiện vật cản và bản đồ chênh lệch

4.2.4. Object Avoidance Node

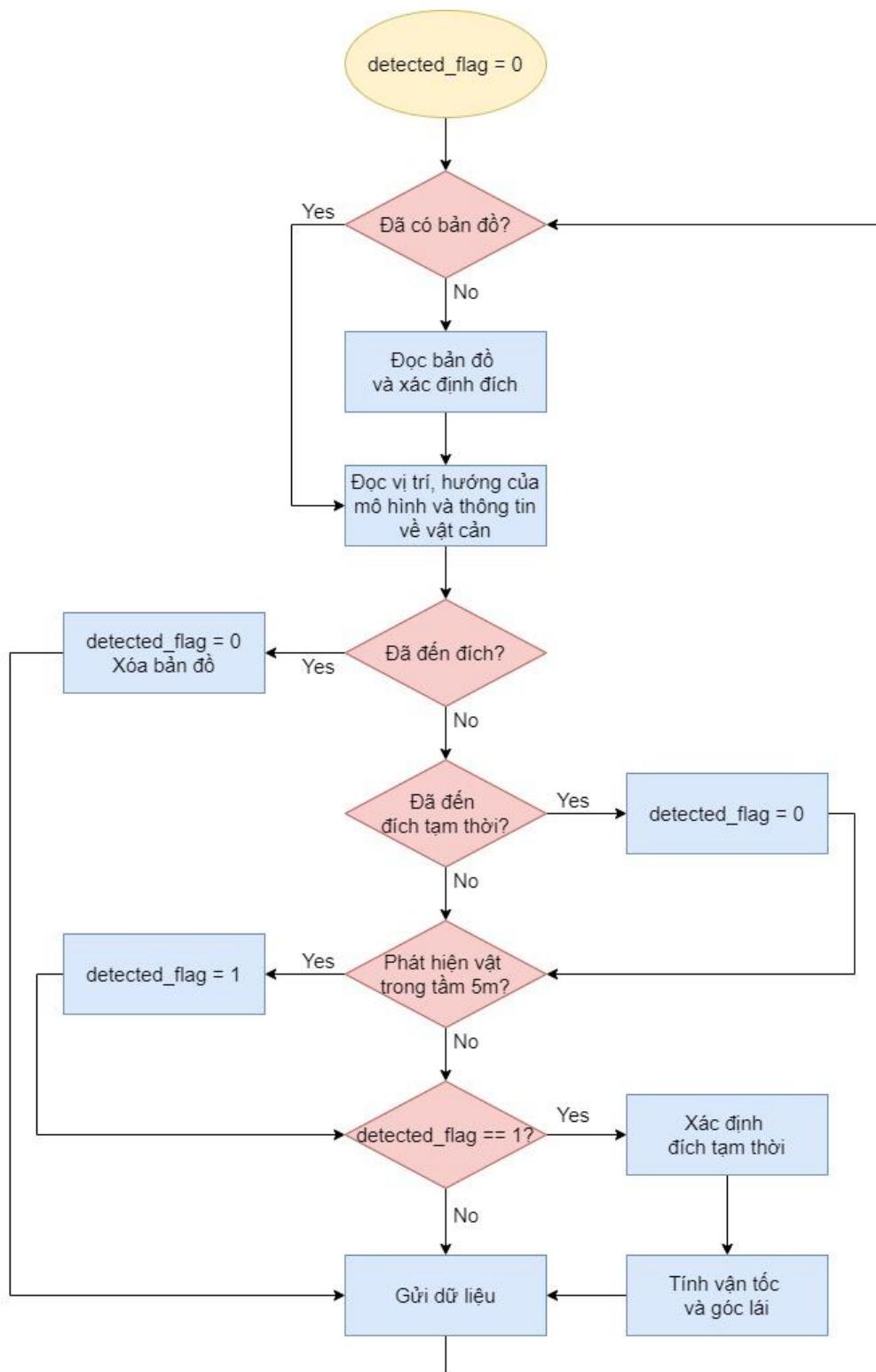
Object Avoidance node dùng để tính toán các giá trị vận tốc và góc lái (v, θ) điều khiển mô hình dựa trên thuật toán tránh vật cản khi cần thiết, với ngõ vào từ các topic:

- OBST: thông tin về các vật cản, bao gồm vị trí, kích thước và tốc độ, được tính toán từ Object Detection node.
- VDATA: vị trí và hướng hiện tại của mô hình, được publish từ SerialPort node.

Ngõ ra của Object Avoidance node được publish vào topic PCDAT, bao gồm:

- Cờ detected_flag: cờ quyết định khi nào mô hình cần ra khỏi quỹ đạo để tránh vật cản. Mô hình tiếp tục bám quỹ đạo khi detected_flag = 0, ra khỏi quỹ đạo khi detected_flag = 1.
- Vận tốc v và góc lái θ : khi detected_flag = 1, mô hình sẽ hoạt động theo vận tốc v và góc lái θ do giải thuật này xuất ra để tránh vật cản. Sau khi đưa mô hình vượt qua khỏi vật cản và trở về gần quỹ đạo, cờ detected_flag được tắt và mô hình tiếp tục bám quỹ đạo theo thuật toán Stanley Steering Control.

Giải thuật điều khiển mô hình tránh vật cản như sau:



Hình 4.15 Giải thuật tránh vật cản

Bước 1: Kiểm tra máy tính nhúng đã nhận được bản đồ được hoạch định trên giao diện hay chưa. Nếu chưa, đọc tọa độ các điểm trên bản đồ từ map_out. Điểm đích được xác định là điểm cuối cùng của bản đồ. Các dữ liệu này được truyền từ STM32F407 thông qua giao tiếp UART và ghi vào file map_out nhờ SerialPort node.

Bước 2: Đọc giá trị vị trí và hướng hiện tại của mô hình từ topic VDATA và thông tin về vật cản phát hiện được từ topic OBST. Đồng thời, ta có thông tin về điểm tham chiếu, điểm trên bản đồ gần nhất với mô hình tại thời điểm hiện tại, được tính toán trên STM32F407 từ topic VDATA.

Bước 3: Kiểm tra mô hình đã đến đích hay chưa. Mô hình được cho là đã đến đích khi khoảng cách giữa mô hình và đích nhỏ hơn 0.5m. Khi đó, tắt cờ detected_flag, xóa bản đồ và đi đến bước 7.

Bước 4: Kiểm tra mô hình đã đến đích tạm thời hay chưa. Mô hình được cho là đã đến đích tạm thời khi điểm tham chiếu hiện tại của mô hình trùng với đích tạm thời. Khi đó, tắt cờ detected_flag. Nếu đích tạm thời chưa được xác định, bỏ qua bước này.

Bước 5: Kiểm tra trong các vật cản phát hiện được, có vật cản nào ở trong phạm vi 5m so với mô hình hay không. Nếu có, bật cờ detected_flag. Nếu không có vật cản nào được phát hiện, bỏ qua bước này.

Bước 6: Kiểm tra cờ detected_flag. Nếu cờ đang được bật, xác định đích tạm thời và tính toán góc lái và vận tốc theo thuật toán tránh vật cản. Từ thông tin về bounding box được gửi từ Detection Node, tâm và bán kính của vật cản được xác định bằng tâm và $\frac{1}{2}$ độ rộng của bounding box chứa vật cản đó.

Chỉ xem xét N điểm tiếp theo trên bản đồ tính từ điểm tham chiếu, vật cản được xác định là sẽ va chạm với mô hình khi có ít nhất một điểm trong N điểm này

có khoảng cách đến tâm của vật cản nhỏ hơn tổng bán kính của vật cản, bán kính của mô hình và khoảng an toàn. Khi đó, đích tạm thời được xác định như sau:

- Nếu không có vật cản nào sẽ va chạm với mô hình, đích tạm thời sẽ là điểm tiếp theo trên bản đồ tính từ điểm tham chiếu.
- Nếu có ít nhất một vật cản sẽ va chạm với mô hình, ta tìm điểm trên bản đồ phía sau vật, gần nhất và cách vật một khoảng an toàn để làm đích tạm thời. Nếu có nhiều đích tạm thời được xác định (hay nhiều vật cản sẽ va chạm với mô hình), ta chọn điểm xa nhất làm đích tạm thời.

Mô hình chỉ tiếp tục bám quỹ đạo sau khi được điều khiển đến đích tạm thời. Việc này đảm bảo cho mô hình không lập tức quay lại quỹ đạo khi vừa mới tránh được vật cản và không nhìn thấy vật cản đó nữa do hạn chế về tầm nhìn của camera.

Sử dụng đích tạm thời, ta tính toán vận tốc và góc lái sử dụng thuật toán tránh vật cản. Trong một số trường hợp, khi vật cản nằm ở giữa quỹ đạo, góc lái tránh sang phải hay trái cho giá trị hàm mục tiêu gần nhau. Điều này làm cho mô hình có xu hướng đi theo đường zigzag do góc lái được chọn thay đổi liên tục trước khi chọn được một bên để tránh vật cản. Để giải quyết vấn đề này, sau khi tính toán hàm mục tiêu, ta sẽ ưu tiên chọn góc lái cùng dấu với góc lái trước đó nếu giá trị hàm mục tiêu của góc lái này không sai lệch nhiều so với giá trị lớn nhất.

Bước 7: Gửi thông tin tính toán được vào topic PCDAT. Nếu cờ detected_flag đang được bật, gửi các giá trị góc lái và vận tốc. Ngược lại, gửi giá trị cờ.

Sau cùng, quay lại bước 1. Quy trình này lặp lại trong suốt quá trình vận hành của mô hình.

4.2.5. SerialPort node

SerialPort node dùng để giao tiếp giữa máy tính nhúng và STM thông qua một USB UART duy nhất kết nối với COM Port của main board máy tính nhúng. Nhiệm vụ chính của SerialPort node là đọc dữ liệu về vị trí và hướng của mô hình từ STM gửi cho Object Avoidance node và đọc dữ liệu về cờ phát hiện vật, vận tốc và góc lái từ Object Avoidance node gửi cho STM. Ngoài ra, SerialPort node còn truyền dữ liệu từ IMU gửi cho STM và nhận lệnh cấu hình từ giao diện, thông qua stm để gửi lên cho máy tính nhúng cấu hình IMU. Có thể xem đây là entry point của việc truyền thông giữa máy tính nhúng và vi điều khiển.

SerialPort node được chia thành 3 luồng:

- **Main thread:**

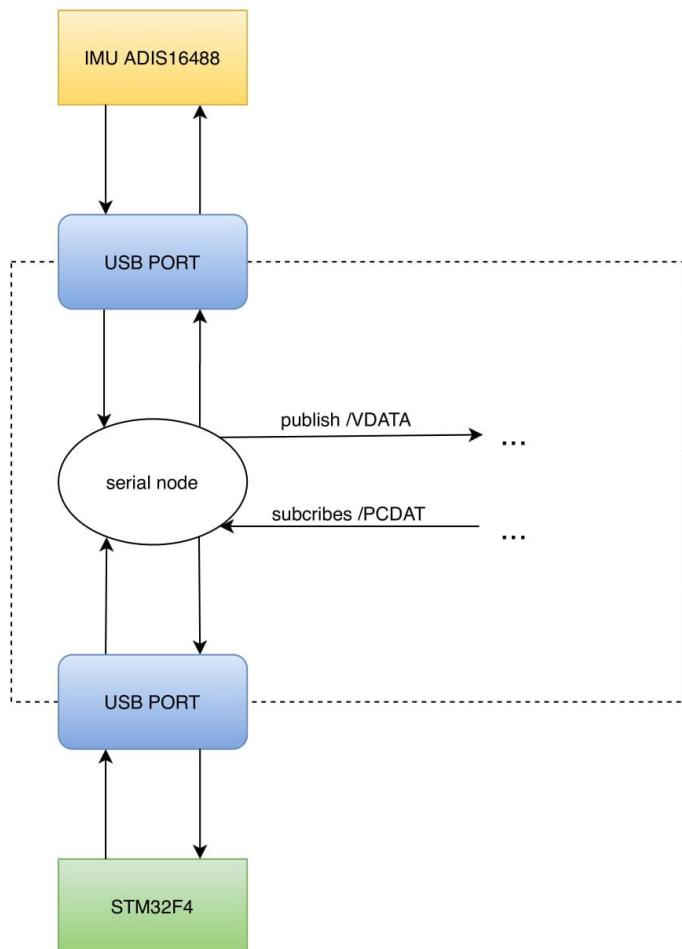
Là luồng chính khởi tạo, quản lý các luồng con và tài nguyên của chúng (như serial files, các mutex lock, thread id, ...). Ngoài ra còn đăng ký node ros và khởi tạo ros node handler, các instance publisher/subscriber để giao tiếp với các node khác. Subscriber sẽ subscribe topic /PCDAT để theo dõi dữ liệu về các đối tượng vật cản, do đó main loop của main thread sẽ kiểm tra callback function đăng ký với subscriber với tần số 10Hz để ra quyết định cho robot né vật cản (nếu có vật cản). Khi nhận được signal kết thúc chương trình (ros::shutdown hoặc nhận được kill signal), main thread sẽ thoát vòng lặp đồng thời terminate các luồng con, thu hồi tài nguyên, giải phóng các vùng nhớ được cấp phát và kết thúc chương trình thành công.

- **USB thread:**

Được tạo ra bởi main thread với nhiệm vụ chính là xử lý dữ liệu nhận được từ vi điều khiển tại cổng usb uart giữa máy tính nhúng và vi điều khiển: các bản tin như các lệnh cấu hình IMU, tọa độ map được nạp lên vi điều khiển, các dữ liệu trong quá trình path tracking để tính toán góc lái và vận tốc né vật cản.

- **IMU thread:**

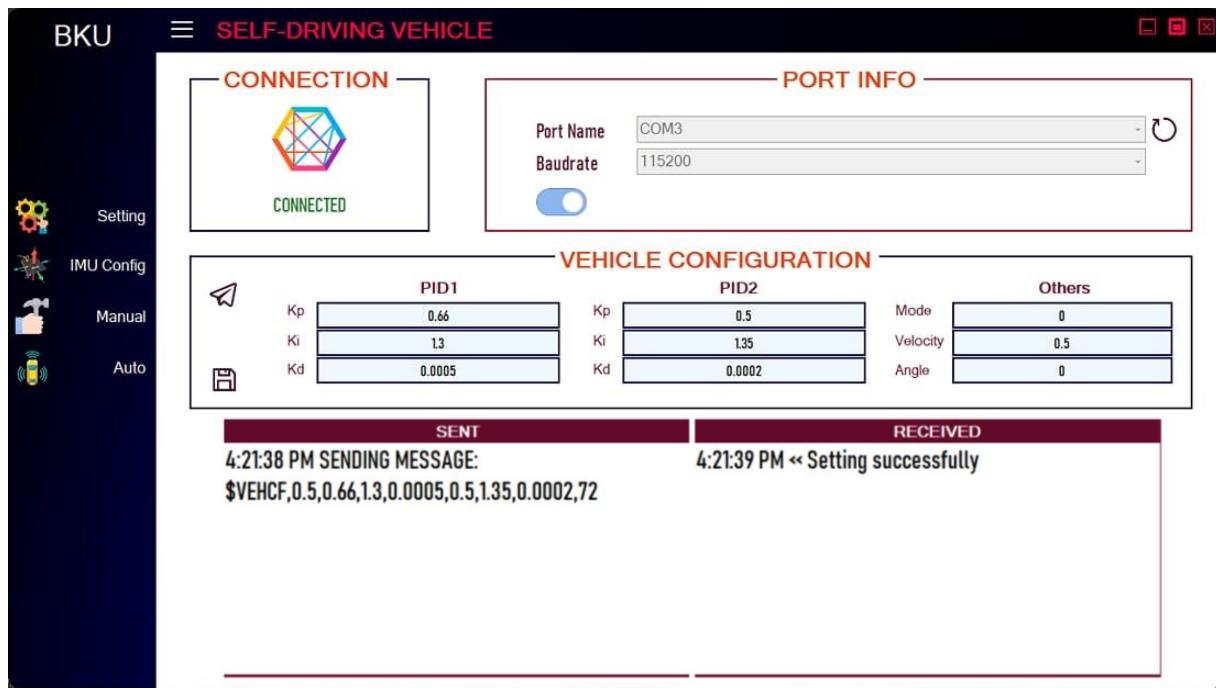
Được tạo bởi USB thread khi máy tính nhúng đã nhận được một map đầy đủ từ vi điều khiển, cũng là điều kiện cần để robot sẵn sàng để chạy path tracking. Luồng này có chức năng chính là đọc dữ liệu từ IMU và gửi cho vi điều khiển với tần số 20Hz, bằng với tần số tính toán góc lái của thuật toán Stanley.



Hình 4.16 Sơ đồ truyền thông Serial Node

4.3. Giao diện điều khiển trên máy tính

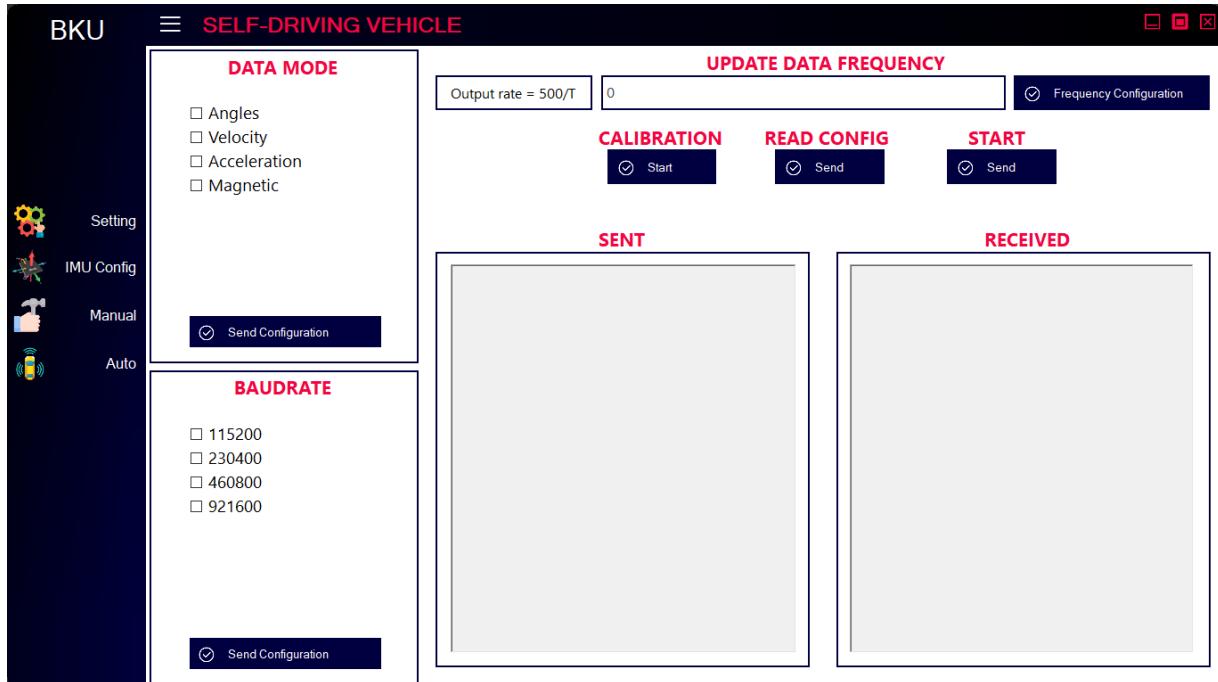
4.3.1. Giao diện cấu hình thông số tổng quát



Hình 4.17 Giao diện cấu hình thông số

- Chọn cổng COM và Baudrate để kết nối tới Serial Port để gửi đi qua Lora.
- Cấu hình thông số PID, vận tốc, và góc cho mô hình.
- Lưu các thông số tốt vào file config để lần sau sử dụng mặc định.
- Chuyển đổi giữa các giao diện để thực hiện các thao tác khác.

4.3.2. Giao diện cấu hình IMU



Hình 4.18 Giao diện cấu hình và calib cảm biến IMU

- Cấu hình thông số cho IMU:

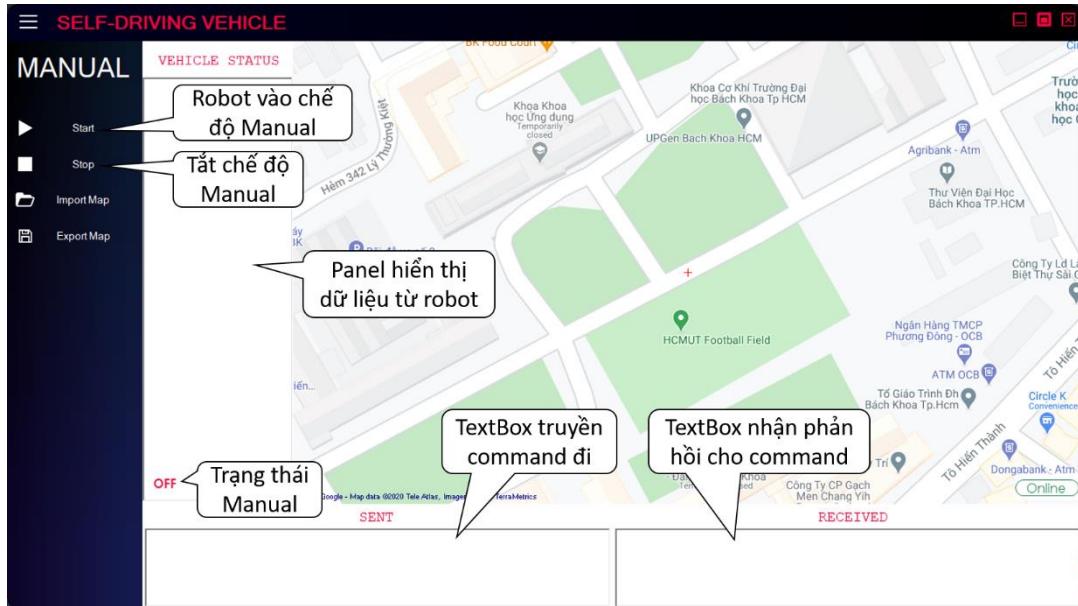
Các chế độ gửi dữ liệu: Góc, vận tốc, gia tốc, từ trường.

Tốc độ truyền Baudrate: 115200 tới 921600.

Tần số cập nhật.

- Đọc thông số từ IMU.
- Calib cảm biến trực tiếp từ giao diện: Sau khi bấm Start, mô hình sẽ gửi lệnh Calib xuống cảm biến và quay 10 vòng để IMU tính toán bias từ trường.
- Sau khi hoàn tất cấu hình và calib, nhấn Send dưới mục Start để gửi lệnh cho phép IMU gửi dữ liệu về.

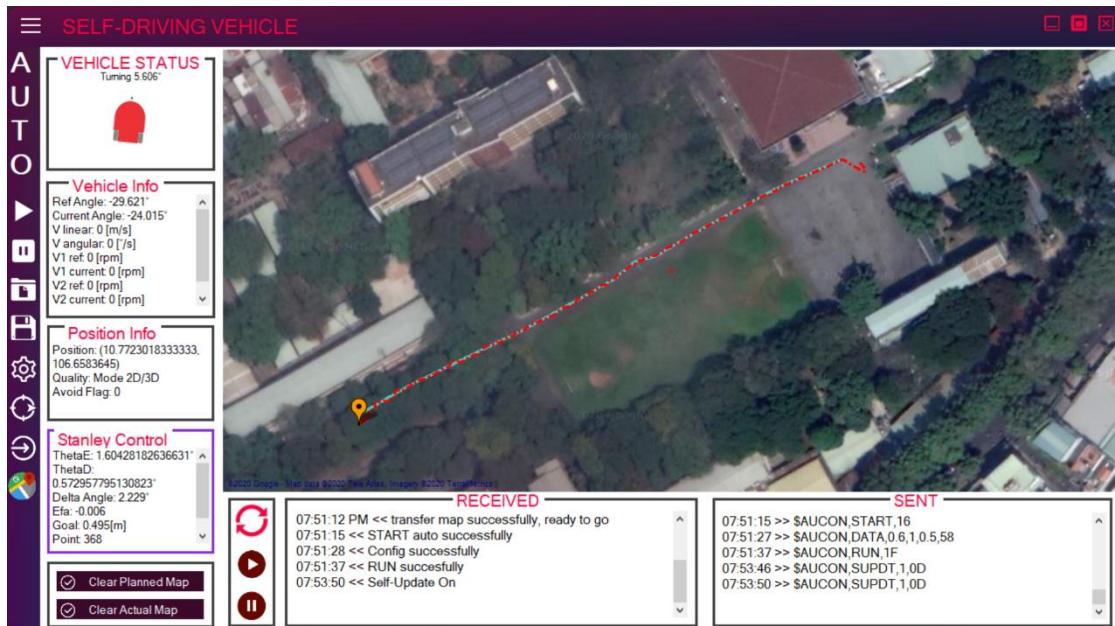
4.3.3. Giao diện điều khiển góc (Fuzzy test)



Hình 4.19 Giao diện điều khiển Manual dựa góc của cảm biến IMU

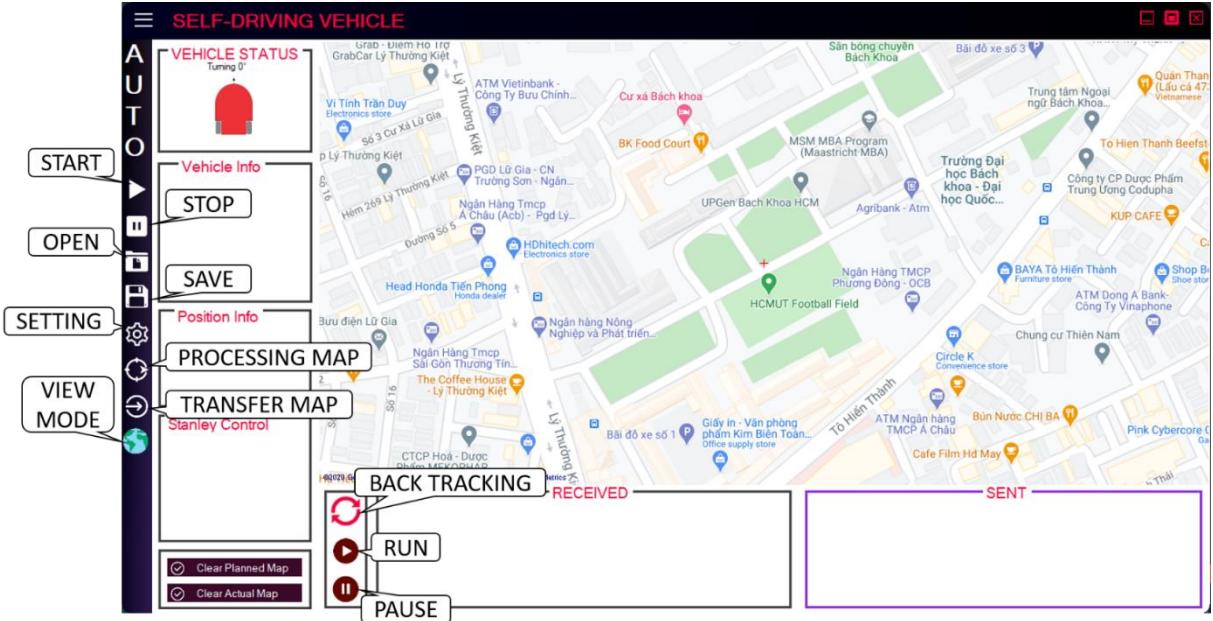
- Tăng/giảm góc bẻ lái và vận tốc lái bằng các phím A/D và W/S.
- Hiển thị dữ liệu từ bộ Fuzzy để giám sát.

4.3.4. Giao diện điều khiển path tracking



Hình 4.20 Giao diện chế độ Auto

Các chức năng chính:

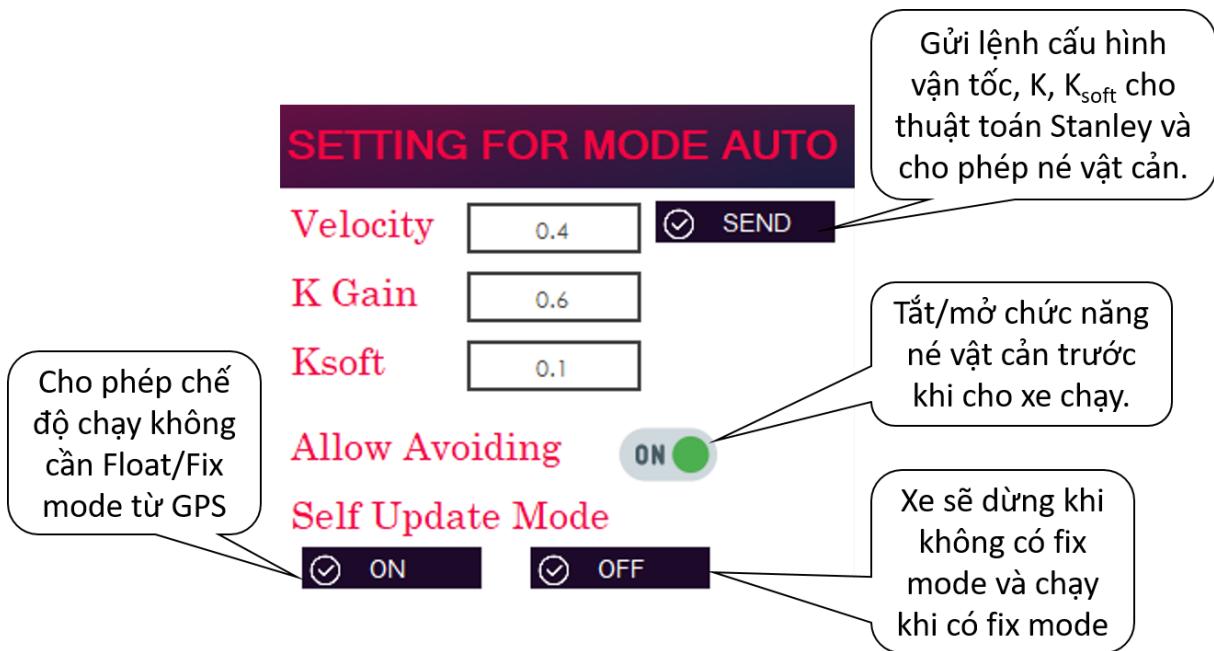


Hình 4.21 Các tính năng chính ở chế độ Auto

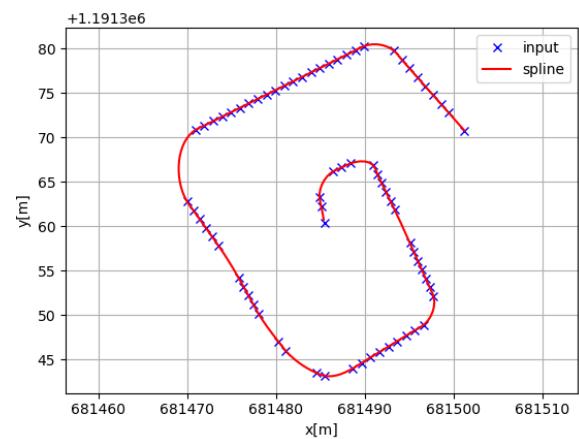
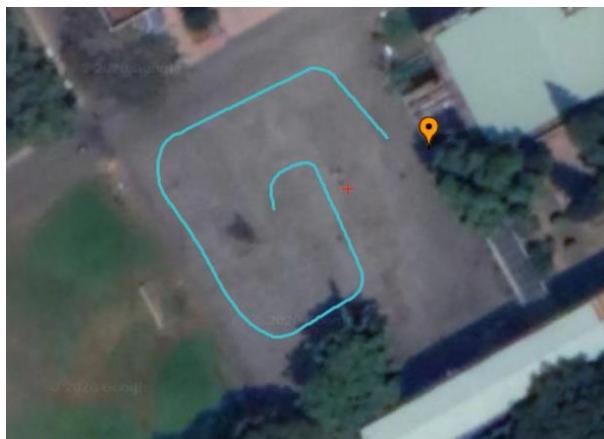
Ngoài giám sát dữ liệu và vẽ hành trình thời gian thực, chế độ auto còn những tính năng sau:

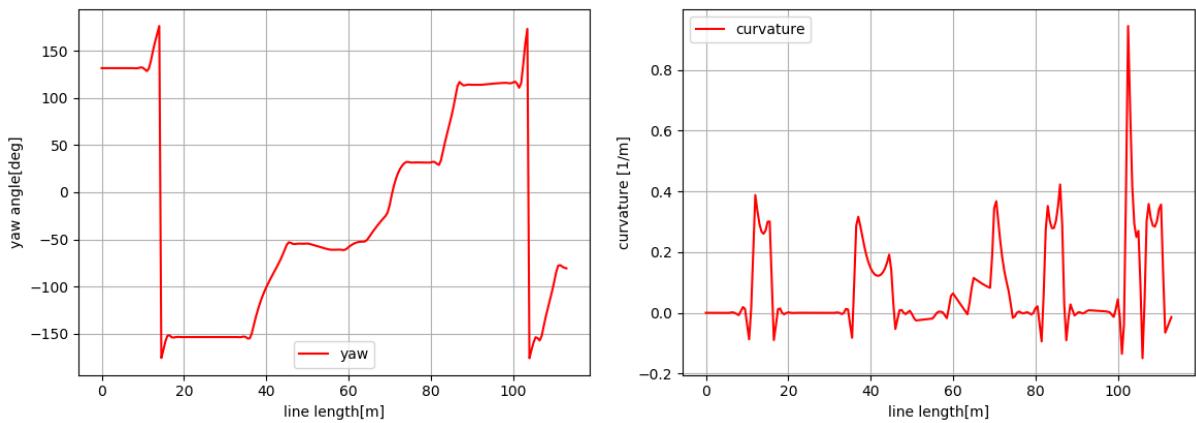
- **START:** Thiết lập chế độ Auto Mode cho robot.
- **STOP:** Tắt chế độ Auto Mode.
- **OPEN:** Mở file dữ liệu đã thu thập.
- **SAVE:** Lưu dữ liệu ra logging file để phân tích lại.
- **SETTING:** Mở panel thiết lập thông số cấu hình cho chế độ trước khi chạy.
- **PROCESSING MAP:** Xử lý file map đầu vào bằng thuật toán Cubic Spline.
- **TRANSFER MAP:** Truyền file map đã xử lý cho robot và hiển thị planning map lên bản đồ.
- **VIEW MODE:** Chuyển chế độ xem sang bản đồ hoặc vệ tinh.
- **BACK TRACKING:** Yêu cầu robot quay ngược lại quỹ đạo ban đầu khi robot đã đến đích.
- **RUN:** Cho phép robot chạy.

- PAUSE: Dừng robot lại khi có vấn đề xảy ra.

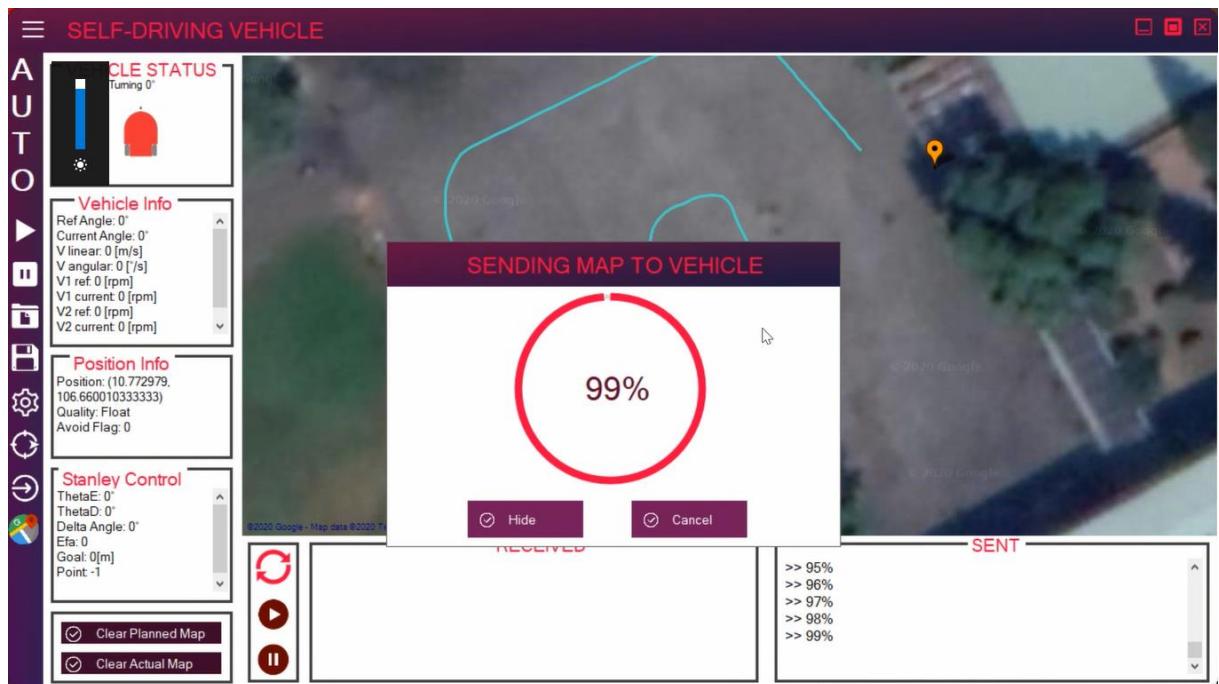


Hình 4.22 Nút SETTING





Hình 4.23 Nút PROCESSING MAP



Hình 4.24 Nút TRANSFER MAP

Chương 5. KẾT QUẢ THỰC NGHIỆM

5.1. Kết quả phát hiện, thep dõi và xác định tọa độ của vật cản

Để khảo sát ảnh hưởng của cường độ sáng, các kết quả được thực hiện trong 3 điều kiện ánh sáng khác nhau: trong nhà, dưới bóng râm và ngoài trời.

5.1.1. Kết quả phát hiện vật

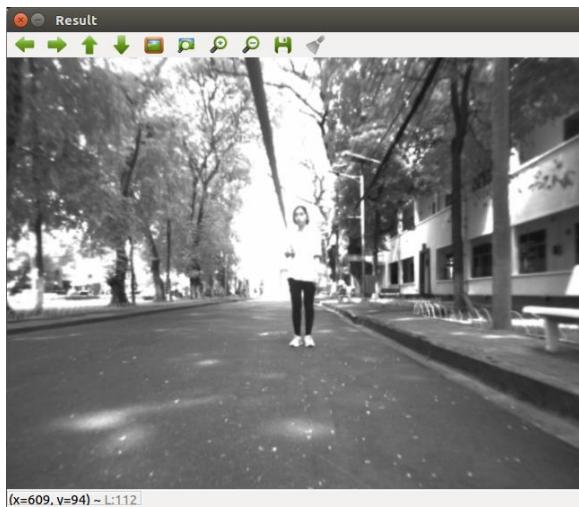
Kết quả dựa trên tỷ lệ khung ảnh có chứa đối tượng mà thuật toán phát hiện được. Đối tượng ở các vị trí cách mô hình từ 1 – 5m.

Bảng 5.1 Kết quả phát hiện đối tượng

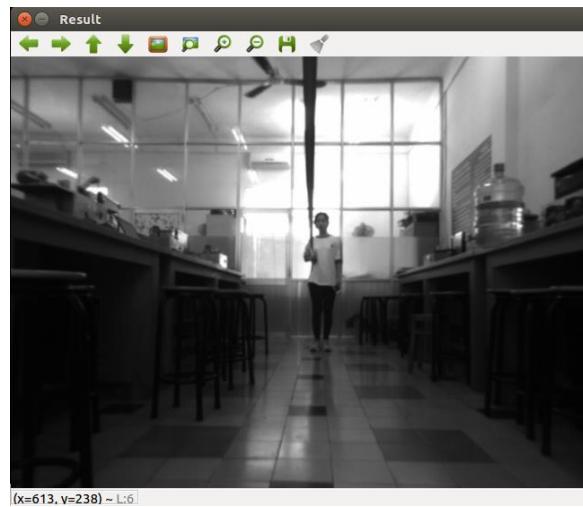
	Tỷ lệ phát hiện người	Tầm hoạt động tốt (m)
Trong nhà	59.8%	0.5 – 5
Dưới bóng râm	42.6%	0.5 – 4
Ngoài trời	73.0%	0.5 – 8

Nhận xét:

- Do camera không có chế độ chống ngược sáng, kết quả phát hiện vật cản bị tác động lớn bởi ánh sáng.
 - Trong nhà: cường độ ánh sáng thấp, khi camera hướng về phía có cường độ ánh sáng cao hơn như ánh nắng từ cửa ra vào, cửa sổ, ban công, ... hình ảnh dễ bị lóa sáng khiến mô hình khó phát hiện vật.
 - Dưới bóng râm: cường độ ánh sáng khá thấp. Tương tự như khi thực hiện trong nhà, camera hướng về phía có cường độ sáng cao hơn khiến hình ảnh bị lóa sáng và mô hình rất khó phát hiện vật.
 - Ngoài trời: cường độ ánh sáng cao, tỷ lệ phát hiện vật tương đối tốt.



a) Dưới bóng râm, khoảng cách 5m



b) Trong nhà, khoảng cách 5m

Hình 5.1 Ví dụ hình ảnh bị lóa sáng

- Khi vật ở quá gần (khoảng cách nhỏ hơn 0.5m), camera chỉ nhìn thấy được một bộ phận của vật, ví dụ như chân hoặc tay của con người, khiến mô hình không thể xác định được đối tượng.
- Khi vật ở quá xa, do ảnh có độ phân giải thấp, kích thước của vật trong ảnh nhỏ khiến mô hình khó phát hiện vật.
- Bounding box trong một số trường hợp không bao được hết đối tượng. Vấn đề này có thể được giải quyết bằng cách cộng thêm một khoảng an toàn vào bán kính của vật, để đảm bảo độ an toàn trong quá trình di chuyển của mô hình.



Hình 5.2 Ví dụ bounding box không bao hết vật

5.1.2. Kết quả theo dõi vật



Hình 5.3 Kết quả theo dõi vật đứng yên



Hình 5.4 Một số kết quả theo dõi vật chuyển động

Nhận xét:

- Đối với vật đứng yên, kết quả theo dõi vật khá tốt, gần như không bị mất dấu của vật cần.
- Đối với vật chuyển động, thuật toán theo dõi MOSSE thực hiện tốt khi vật có chuyển động không quá nhanh. Khi vật chuyển động quá nhanh (hoặc vật chuyển động với tốc độ trung bình nhưng ở quá gần mô hình), trong một số trường hợp, bộ theo dõi MOSSE không bắt kịp vật mà chuyển sang theo dõi nền.

- Để loại bỏ các bounding box không chứa vật trên, tại mỗi lần phát hiện vật cản, trong các bounding box trước đó, bounding box nào có IoU với ít nhất một bounding box mới lớn hơn 0.5 thì được xác định là có chứa đối tượng, và ngược lại. Dựa vào đó, ta xóa đi các bounding box được xác định là không chứa đối tượng từ 2 lần trở lên. Do độ chính xác của bộ phát hiện đối tượng MobileNet SSD, ta sẽ có một vài khung hình không phát hiện được đối tượng. Điều này dẫn đến trường hợp bounding box có chứa đối tượng nhưng vẫn bị xóa và phải chờ đến các lần phát hiện sau để tiếp tục theo dõi.

5.1.3. Kết quả xác định tọa độ của vật

Tọa độ của vật được xác định trong hệ tọa độ của mô hình, với gốc tọa độ tại điểm chính giữa của stereo camera, chiều dương trục X hướng sang phải và chiều dương trục Y hướng ra xa mô hình. Mỗi giá trị được thực hiện 5 lần đo, giá trị tính toán trung bình được ghi nhận.

Sai số trung bình được tính bằng công thức:

$$\overline{\Delta p} = \frac{1}{N} \sum_{i=1}^N \Delta p_i$$

Với N là số lần thực hiện, và:

$$\Delta p_i = \sqrt{(\Delta x_i)^2 + (\Delta y_i)^2}$$

Sai số RMS được tính bằng công thức:

$$\Delta p_{RMS} = \sqrt{\frac{\sum_{i=1}^N (\overline{\Delta p} - \Delta p_i)^2}{N}}$$

Bảng 5.2 Kết quả xác định vị trí vật cản

Trong nhà					
Giá trị thực		Giá trị tính toán		Sai số trung bình	Sai số RMS
X	Y	X	Y		
0	1	0.068	1.010	0.081	0.008
0	2	0.111	2.035	0.120	0.038
0	3	0.154	3.082	0.268	0.018
0	4	0.092	4.060	0.206	0.021

Dưới bóng râm					
Giá trị thực		Giá trị tính toán		Sai số trung bình	Sai số RMS
X	Y	X	Y		
0	1	-0.018	0.986	0.026	0.023
0	2	0.002	2.028	0.035	0.014
0	3	-0.041	3.097	0.106	0.004

Ngoài trời					
Giá trị thực		Giá trị tính toán		Sai số trung bình	Sai số RMS
X	Y	X	Y		
0	1	-0.116	1.081	0.158	0.039
0	2	-0.081	2.025	0.102	0.018
0	3	-0.058	2.934	0.097	0.022
0	4	0.040	4.072	0.101	0.045
0	5	0.067	5.023	0.080	0.022

Nhận xét:

- Với thuật toán đơn giản Block Matching, mặc dù không tạo ra một bản đồ chênh lệch hoàn hảo (không phải pixel nào cũng tìm được pixel tương ứng và tính được độ sâu, được thể hiện bằng các mảng màu xám trên bản đồ chênh lệch), nhưng với việc chọn ra khoảng giá trị độ chênh lệch tập trung nhiều pixel nhất trong bounding box, kết quả xác định vị trí và kích thước của vật tương đối tốt.
- Theo công thức tính độ sâu, khoảng cách tỉ lệ nghịch với độ chênh lệch, do đó để xác định vị trí các vật ở càng gần, ta cần giá trị disparity_range càng cao. Ví dụ với disparity_range = 64, ta có thể xác định vị trí của các vật cách mô hình từ 1m, và disparity_range = 128 với khoảng cách từ 0.5m. Khi vật ở khoảng cách nhỏ hơn các giá trị này, mô hình có thể không xác định được hoặc thậm chí xác định sai vị trí của vật. Ngược lại, giá trị disparity_range quá lớn sẽ làm tăng khối lượng tính toán và giảm tốc độ xử lý của mô hình.

5.1.4. Tốc độ xử lý

Với tần số của stereo camera là 20 FPS, mô hình có thể phát hiện, theo dõi vật cản và tính toán các giá trị tọa độ với tốc độ được ghi trong bảng sau.

Bảng 5.3 Tốc độ xử lý của thuật toán phát hiện, theo dõi vật cản

	Tốc độ xử lý (FPS)
Trong nhà	16.818
Dưới bóng râm	18.143
Ngoài trời	17.633

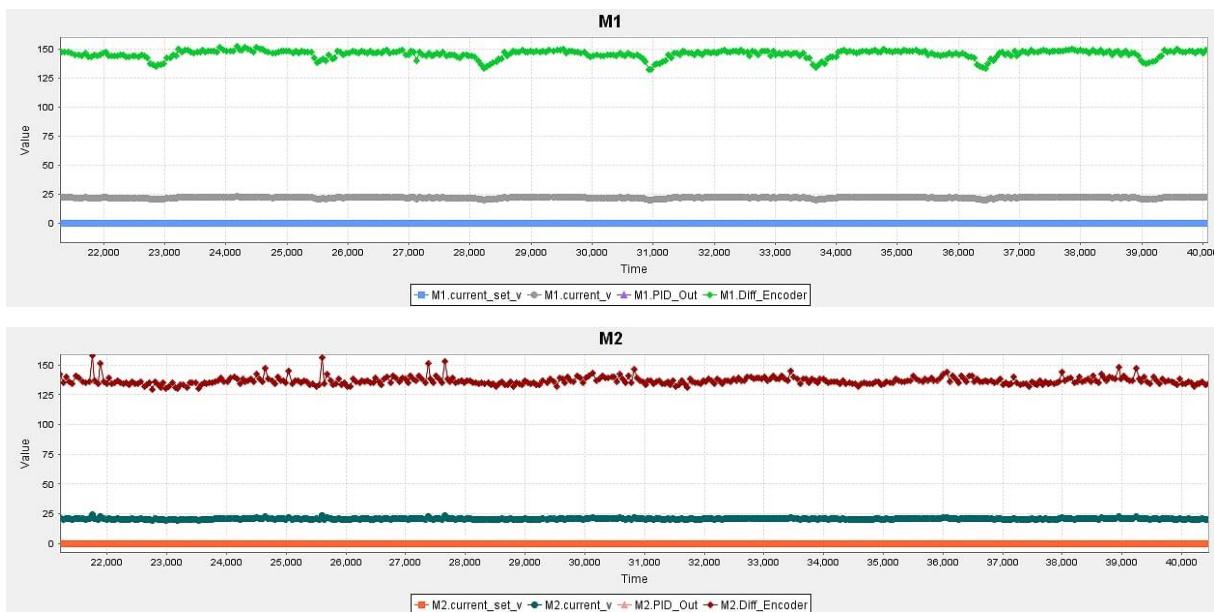
Nhận xét:

- Tốc độ của thuật toán phát hiện, theo dõi và xác định vị trí của vật cản không bị ảnh hưởng bởi cường độ ánh sáng, trong 3 môi trường khác nhau đều đạt được 16 – 18 FPS.

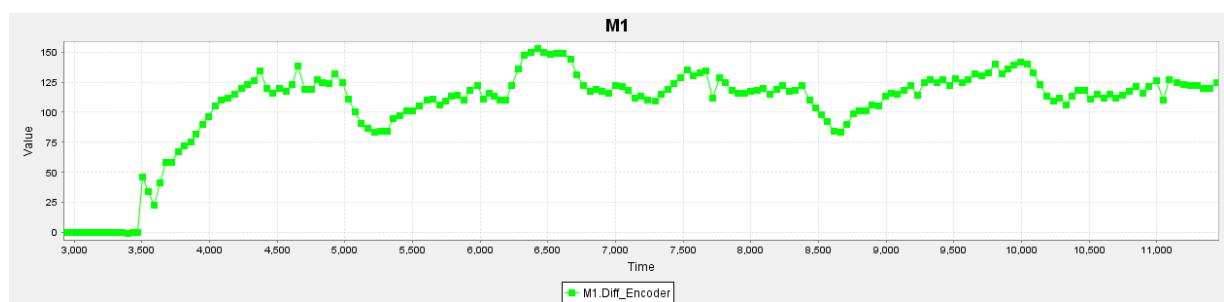
5.2. Các kết quả điều khiển

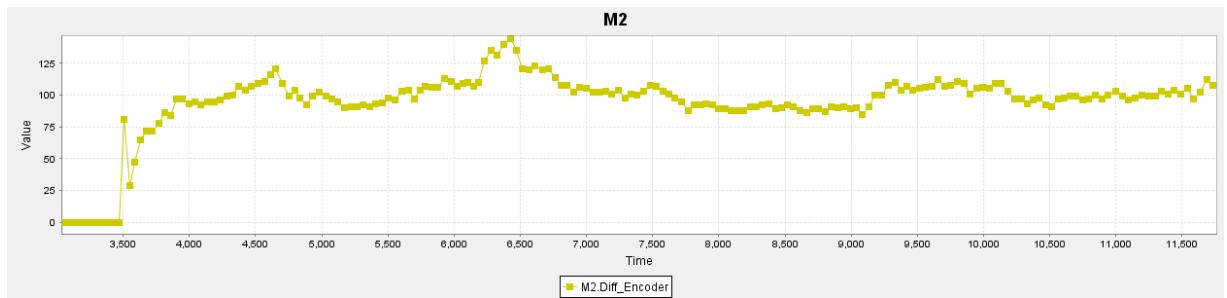
5.2.1. Kết quả lọc nhiễu và xử lý Encoder

Dưới đây là độ thị thu thập dữ liệu encoder theo thời gian thực khi cấp một mức PWM là 20% duty cycle.



Hình 5.5 Tín hiệu đọc về từ Encoder không tải, PWM=20%





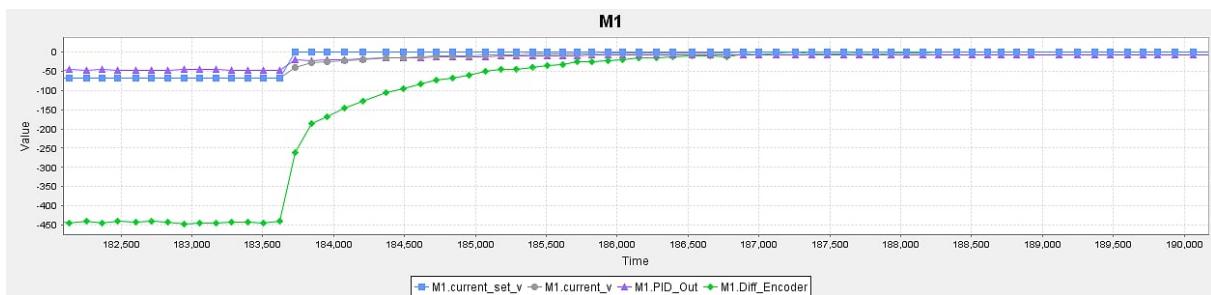
Hình 5.6 Tín hiệu đọc về từ Encoder có tải, PWM=20%

Nhận xét:

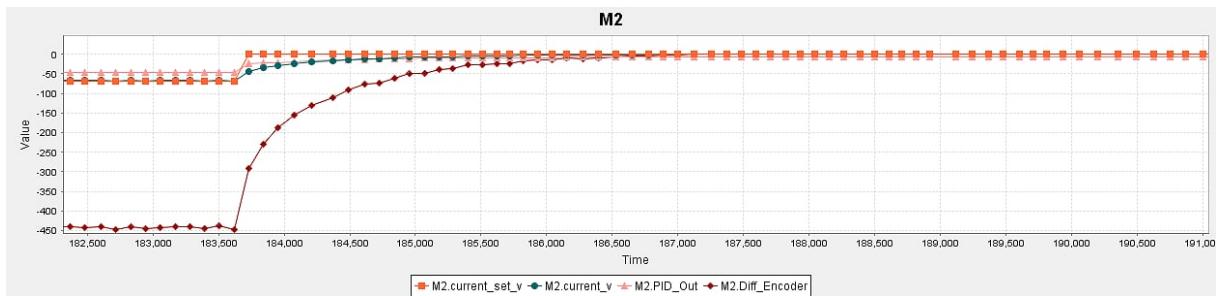
- Do tải trọng mô hình khá nặng (15kg) nên tín hiệu encoder có tải không đều như lúc không tải. Tuy nhiên đã hạn chế được gai nhiễu giúp vận tốc tính toán ổn định hơn.
- Xử lý encoder chính xác là nền tảng cốt lõi cho việc vận hành các thuật toán và các bộ điều khiển. Bởi vì sau cùng thì bài toán đều quy về điều khiển vận tốc 2 động cơ.
- Có thể kết hợp thêm các cách xử lý số liệu từ phương pháp tính và bộ lọc FIR để giảm các gai nhiễu.

5.2.2. Kết quả điều khiển vận tốc PID

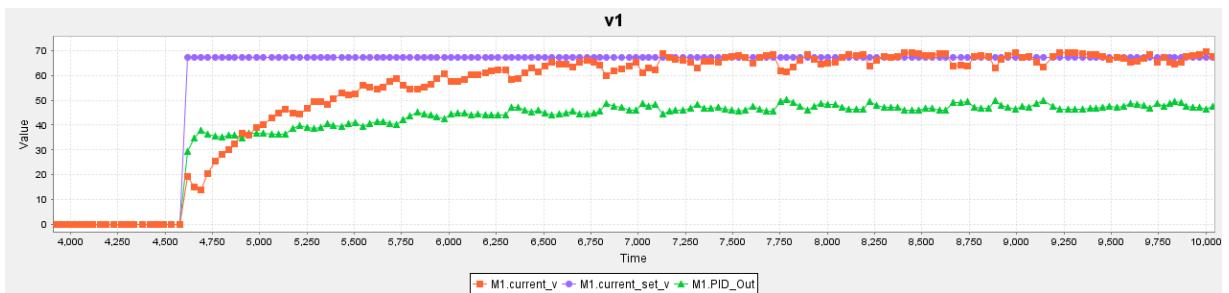
Từ nền tảng của việc xử lý chính xác tín hiệu encoder, chúng ta cần điều chỉnh PID hợp lý để tìm một bộ số cho việc đáp ứng vận tốc động cơ



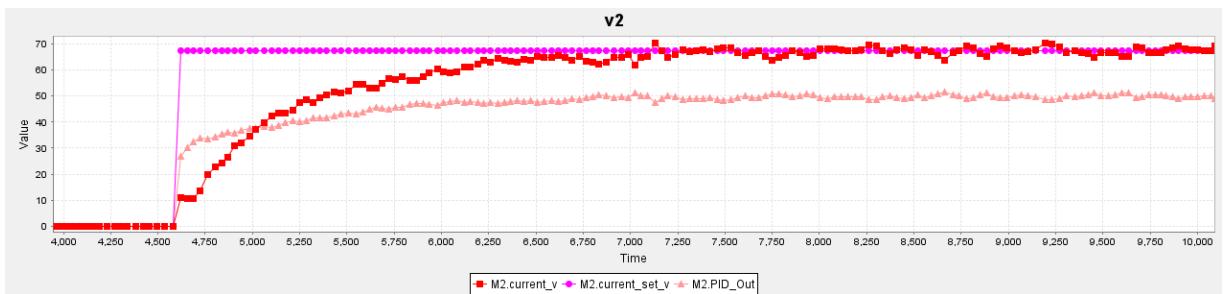
Hình 5.7 Đáp ứng điều khiển vận tốc M1 không tải, $K_p=0.66$ $K_i=1.3$ $K_d=0.0005$



Hình 5.8 Đáp ứng điều khiển vận tốc M2 không tải, $K_p=0.5$ $K_i=1.35$ $K_d=0.0002$



Hình 5.9 Đáp ứng điều khiển vận tốc M1 có tải, $K_p=0.66$ $K_i=1.3$ $K_d=0.0005$



Hình 5.10 Đáp ứng điều khiển vận tốc M2 có tải, $K_p=0.5$ $K_i=1.35$ $K_d=0.0002$

Nhận xét:

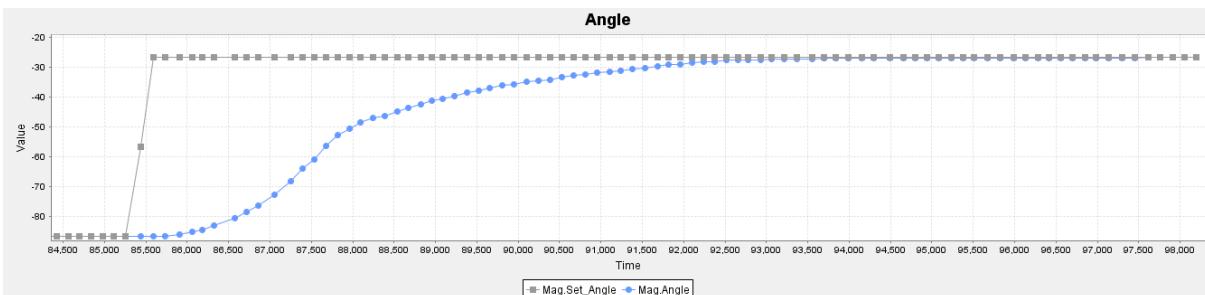
- Đáp ứng có tải tuy không ổn định như không tải nhưng vẫn đáp ứng được yêu cầu điều khiển vận tốc với sai số xác lập thấp.
- Vì bài toán của chúng ta cần điều khiển chính xác góc lái của robot, nên yêu cầu chọn các hệ số PID của từng động cơ sao cho đáp ứng của 2 động cơ càng giống nhau càng tốt. Vì nếu hai đáp ứng khác nhau, khi cho cùng một vận tốc đặt thì động cơ sẽ chạy lệch về một hướng, ảnh hưởng đến góc lái mong muốn.

- Tuy có cùng thông số kỹ thuật trên lý thuyết, hai động cơ khác nhau hoàn toàn có đáp ứng khác nhau trong thực tế, điều này đòi hỏi phải tự mò và điều chỉnh hệ số PID cho từng động cơ chứ không dùng chung một bộ số PID cho cả hai động cơ.

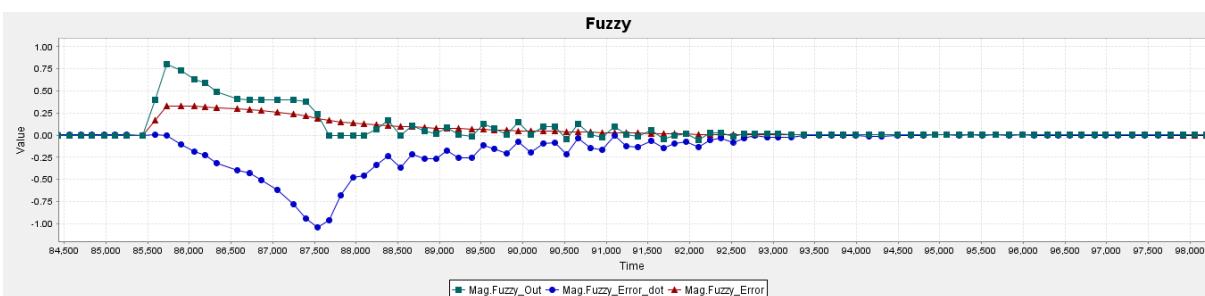
5.2.3. Kết quả điều khiển góc Fuzzy

Output của thuật toán Stanley cho chúng ta một góc $\Delta\varphi$ (delta phi) so với góc heading của robot hiện tại để có thể đưa robot vào quỹ đạo mong muốn, góc $\Delta\varphi$ này được qua bộ điều khiển mờ để đưa góc heading trở về mong muốn, kết quả của bộ điều khiển góc thực nghiệm thể hiện bởi các hình sau:

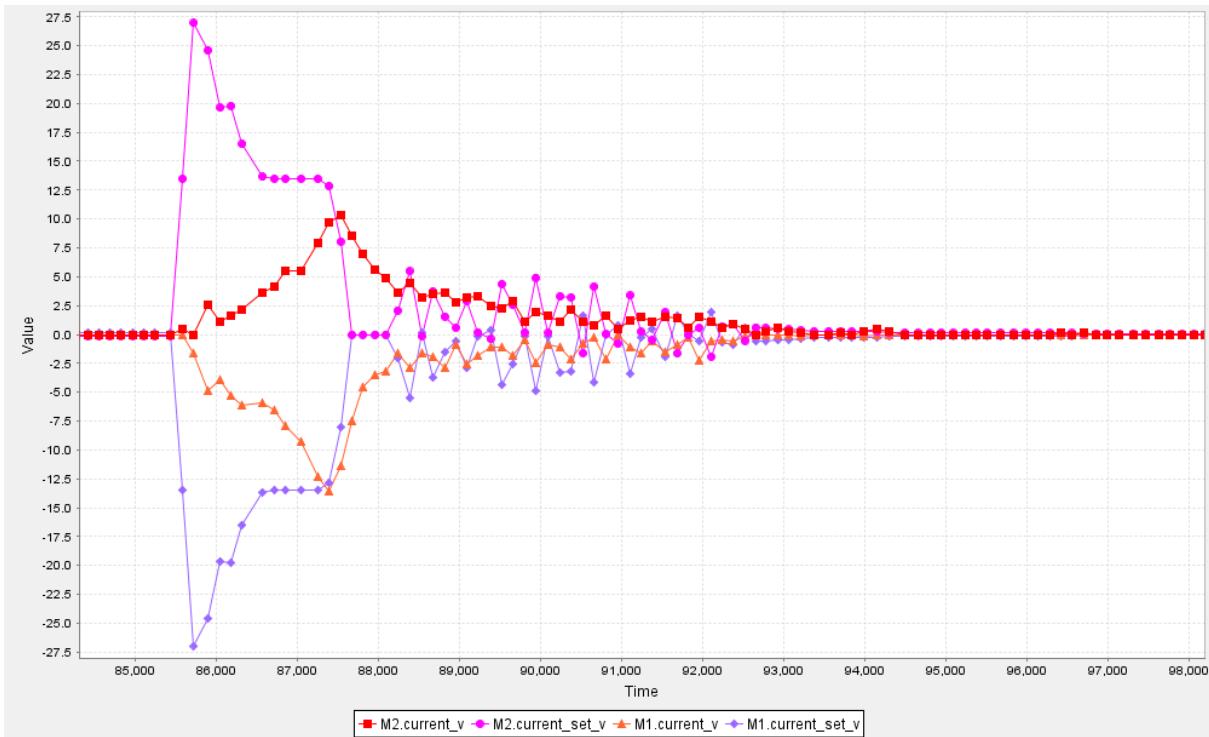
Trường hợp 1: Góc $\Delta\varphi = +60^\circ$ so với heading, yêu cầu robot quẹo phải một góc 60°



a) Bộ điều khiển góc của robot^o



b) Bộ điều khiển Fuzzy



c) Vận tốc của 2 động cơ

Hình 5.11 Đáp ứng với $\Delta\varphi = +60^\circ$

Nhận xét:

- Việc đáp ứng góc còn phụ thuộc vào vận tốc tối đa cho phép của 2 động cơ, bởi vì output của Fuzzy là u thuộc $[-1, 1]$ với $K_u = 1$, từ đó vận tốc đặt của mỗi động cơ được tính bằng $v_1 = +u \times v_{max}$ và $v_2 = -u \times v_{max}$ nếu robot quẹo trái hoặc $v_1 = -u \times v_{max}$ và $v_2 = +u \times v_{max}$ nếu robot quẹo phải. Vì vậy v_{max} càng lớn, đáp ứng góc càng nhanh.

5.2.4. Kết quả điều khiển bám quỹ đạo

Để khảo sát tác động và ảnh hưởng của vận tốc đến chất lượng bộ điều khiển, các thí nghiệm được thực hiện lần lượt bằng cách khảo sát đáp ứng cùng quỹ đạo với các vận tốc khác nhau, và với vận tốc hỗn hợp.

Thực nghiệm cho thấy với vận tốc $V = [0.2, 0.5]$ (m/s), $K = [0.7, 0.8]$ và $K_{soft} = 0.001$, kết quả thu được có sai số dao động từ 3.8 (cm) tới 7 (cm).

Với vận tốc $V \geq 0.7$ (m/s), $K = [0.7, 0.8]$ và $K_{soft} = 0.001$, kết quả thu được có sai số dao động từ 10 (cm) tới 16 (cm).

Chú thích:

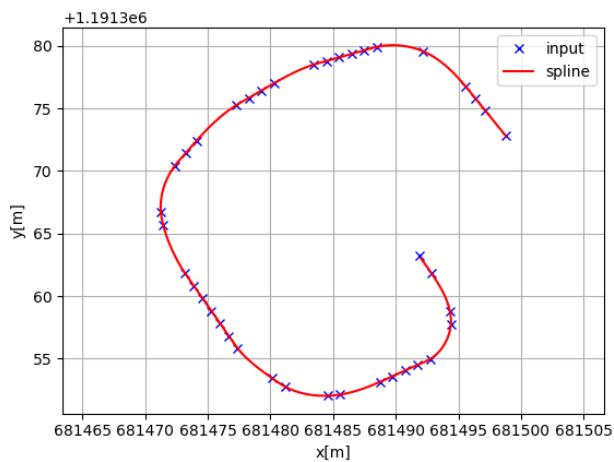
- Đường màu xanh: Quỹ đạo đặt trước của mô hình.
- Đường màu đỏ: Đáp ứng thực tế của mô hình.

Quỹ đạo được chọn có dạng đường cong tròn, độ dài 75.5m.

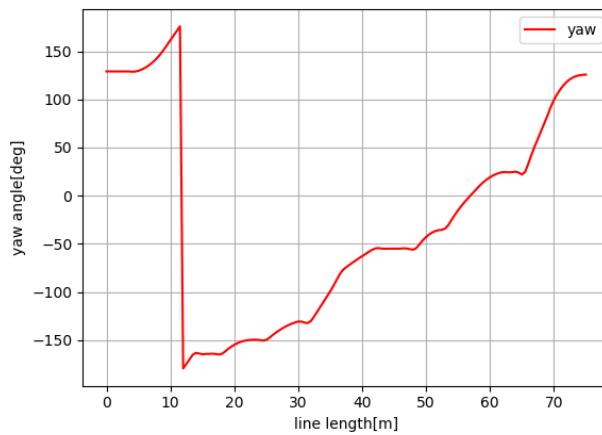


Hình 5.12 Đáp ứng thực tế quỹ đạo 1, $V = 0.2$ (m/s)

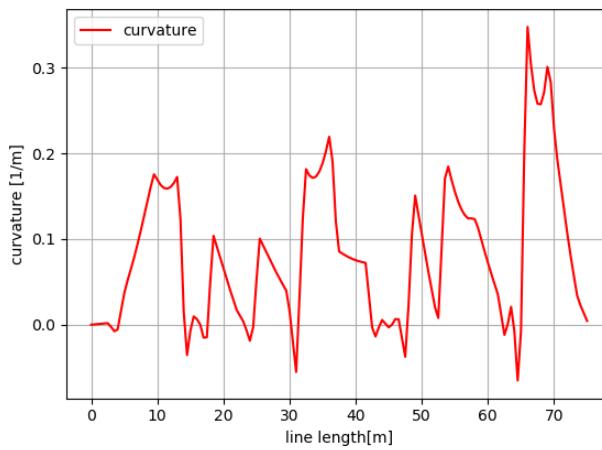
Một số hình ảnh xây dựng và phân tích thông số quỹ đạo:



Hình 5.13 Mô phỏng quỹ đạo 1 trên hệ tọa độ UTM



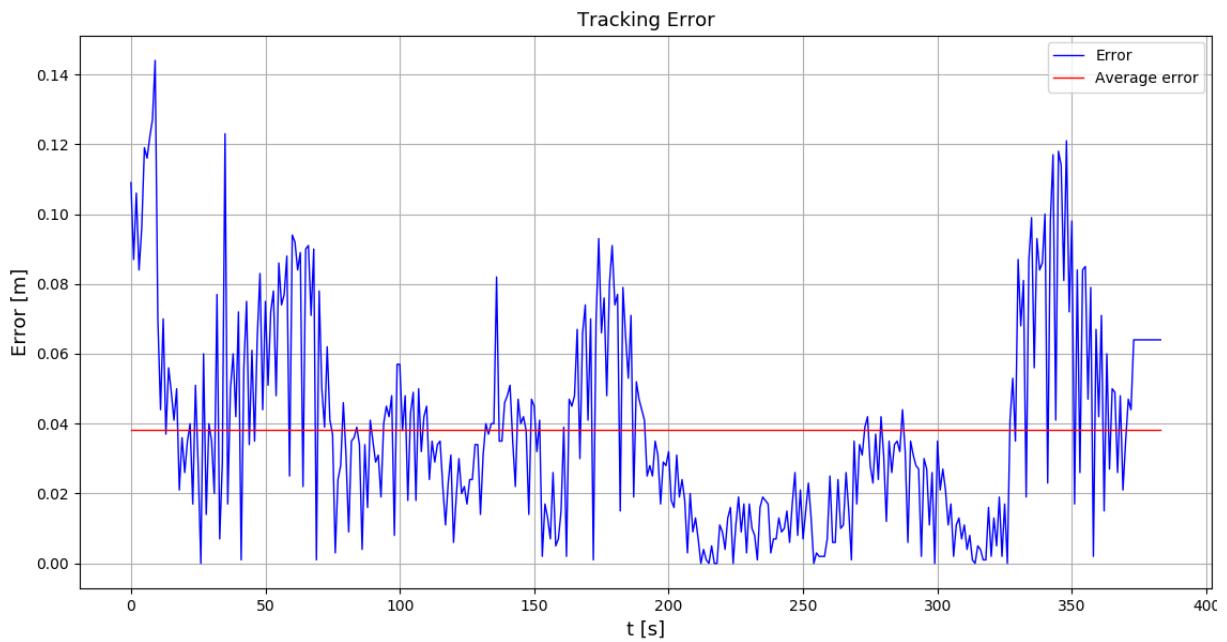
Hình 5.14 Góc tiếp tuyến của quỹ đạo so với trục x trên hệ tọa độ UTM



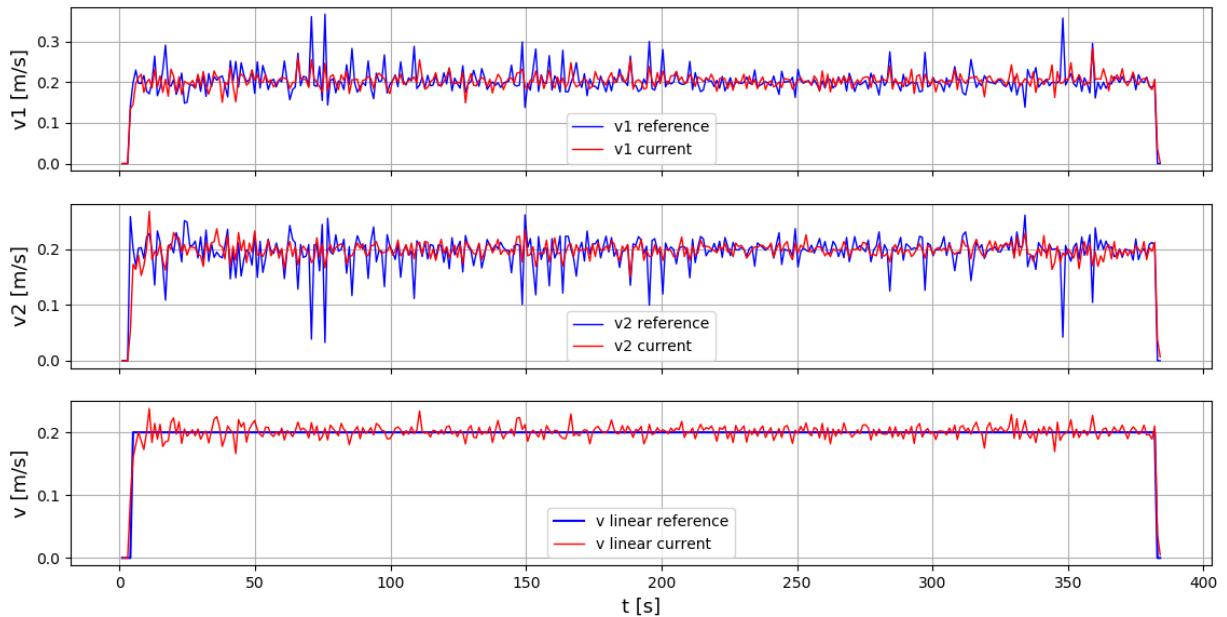
Hình 5.15 Đồ thị thể hiện đặc tính cong của quỹ đạo

Các trường hợp thực nghiệm:

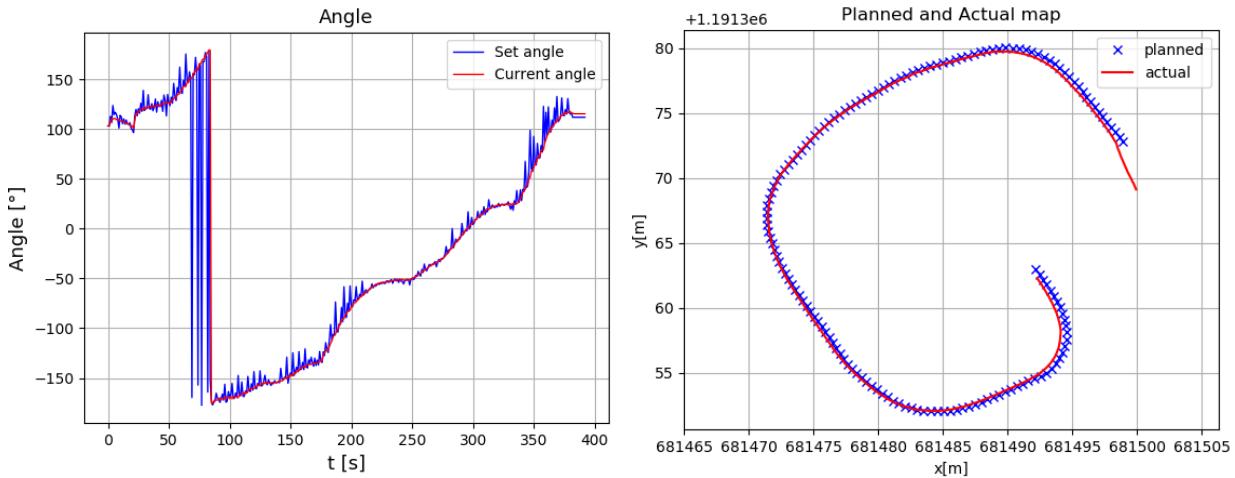
Trường hợp 1: $V = 0.2$ (m/s), $K = 0.8$, $K_{soft} = 0.001$



Hình 5.16 Đồ thị error thể hiện sai lệch vị trí robot so với quỹ đạo, $V = 0.2$ (m/s)



Hình 5.17 Đồ thị đáp ứng vận tốc, $V = 0.2$ (m/s)



Hình 5.18 Đáp ứng góc và quỹ đạo từ A đến B, $V = 0.2 \text{ m/s}$

Nhận xét:

- Các đường gai trong đồ thị đáp ứng góc là do tầm tính toán nằm trong khoảng $(0, 360^\circ)$, không ảnh hưởng chất lượng điều khiển.
- Đồ thị đáp ứng góc có hình dạng giống đồ thị góc tiếp tuyến quỹ đạo cho thấy góc lái robot khá chính xác.

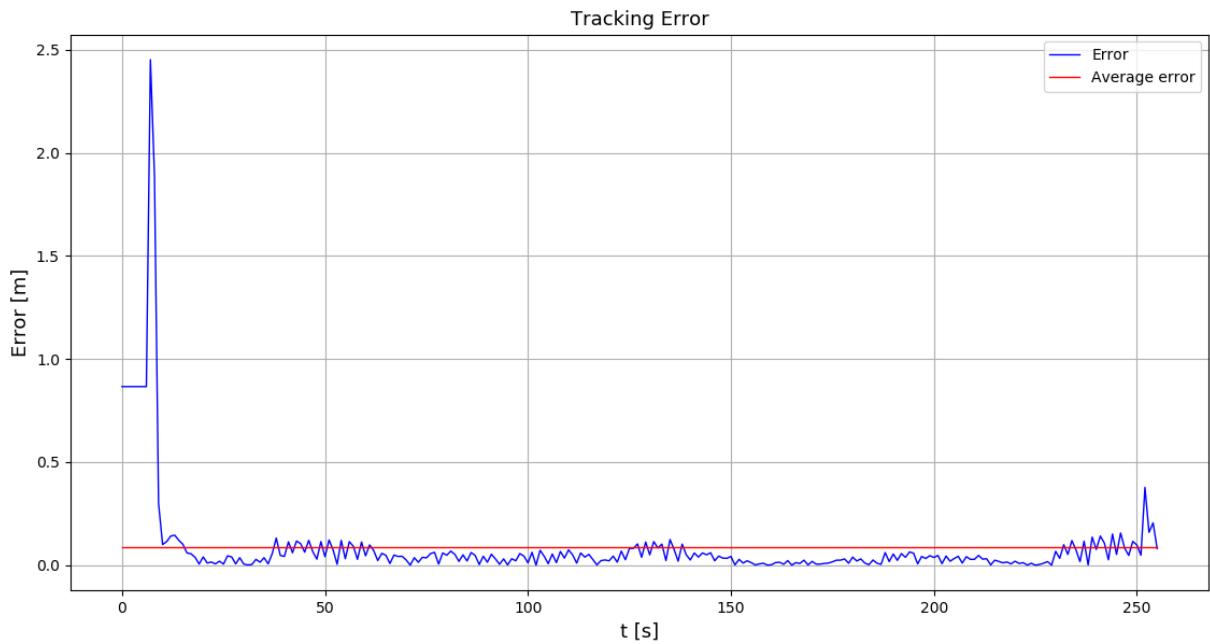
Ta có bảng kết quả sau, với:

- Sai số xác lập trung bình \bar{E} .
- Sai số góc trung bình \bar{E}_θ .
- Vận tốc dài trung bình \bar{V} , sai số trung bình \bar{E}_v .
- Vận tốc phải trung bình \bar{V}_1 , sai số trung bình \bar{E}_{v1} .
- Vận tốc trái trung bình \bar{V}_2 , sai số trung bình \bar{E}_{v2} .

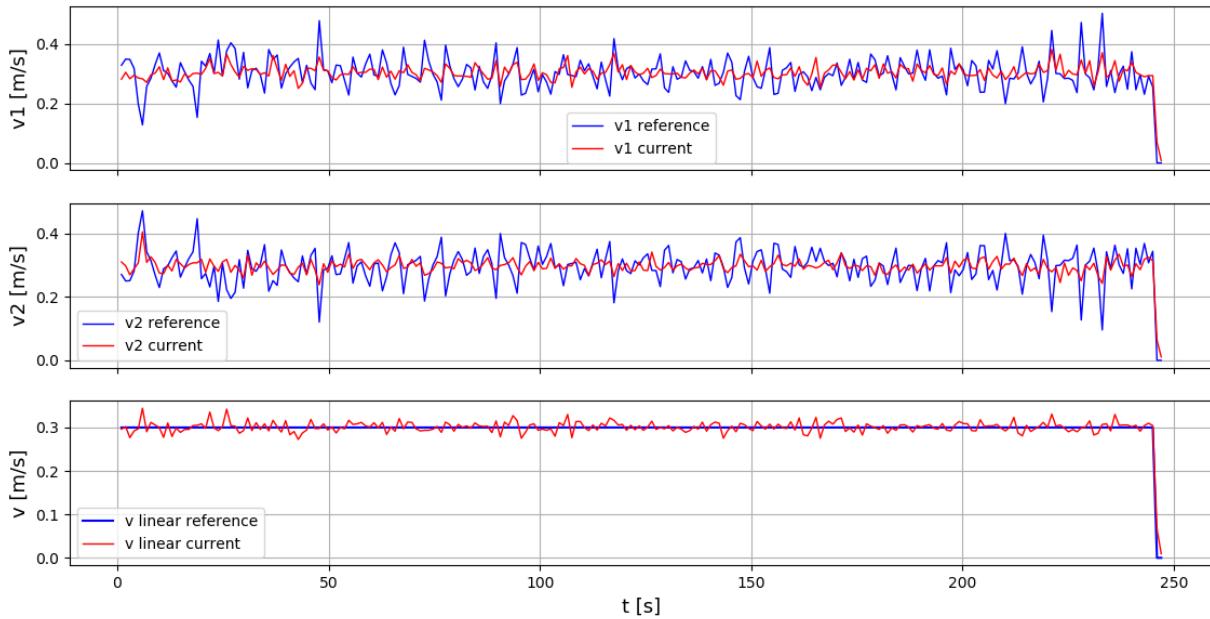
Bảng 5.4 Kết quả điều khiển bám quỹ đạo với $V = 0.2 \text{ m/s}$

Vận tốc đặt (m/s)	\bar{E} (m)	\bar{E}_θ ($^\circ$)	\bar{V} (m/s)	\bar{E}_v (m/s)	\bar{V}_1 (m/s)	\bar{E}_{v1} (m/s)	\bar{V}_2 (m/s)	\bar{E}_{v2} (m/s)
0.2	0.0079	4.5823	0.2011	0.0079	0.2037	0.0192	0.1985	0.0171

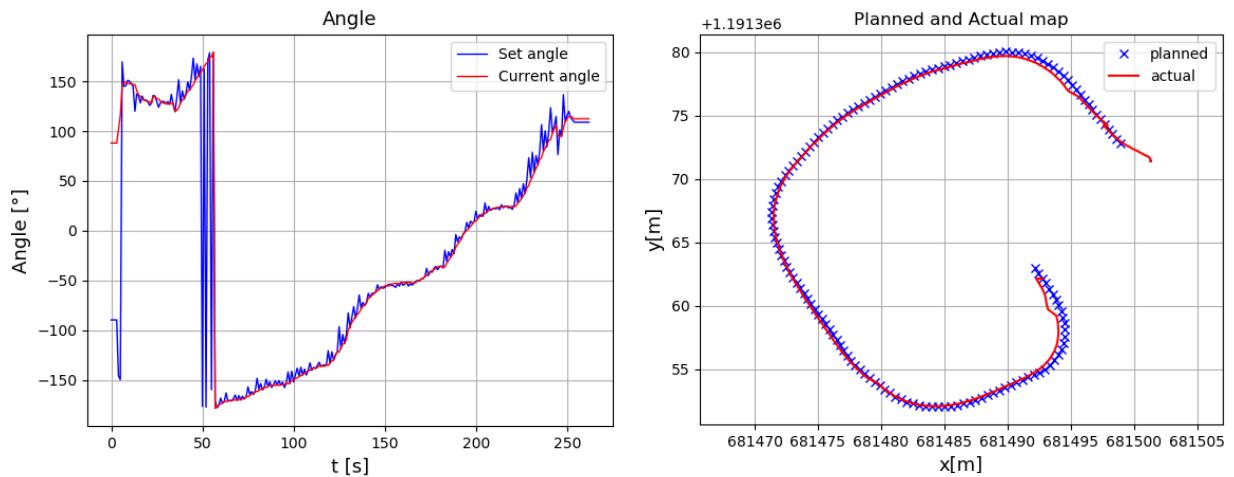
Trường hợp 2: $V = 0.3$ (m/s), $K = 0.8$, $K_{soft} = 0.001$



Hình 5.19 Đồ thị sai số quỹ đạo của robot, $V = 0.3$ (m/s)



Hình 5.20 Đồ thị đáp ứng vận tốc, $V = 0.3$ (m/s)

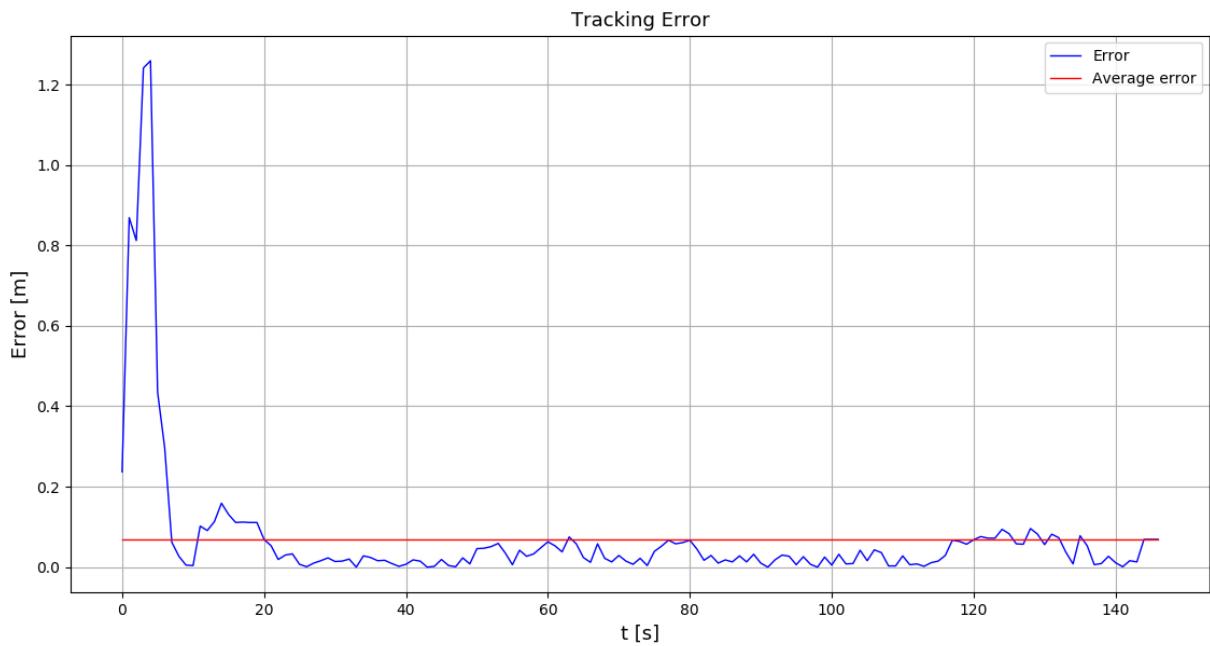


Hình 5.21 Đáp ứng góc và quỹ đạo từ A đến B, $V = 0.3 \text{ m/s}$

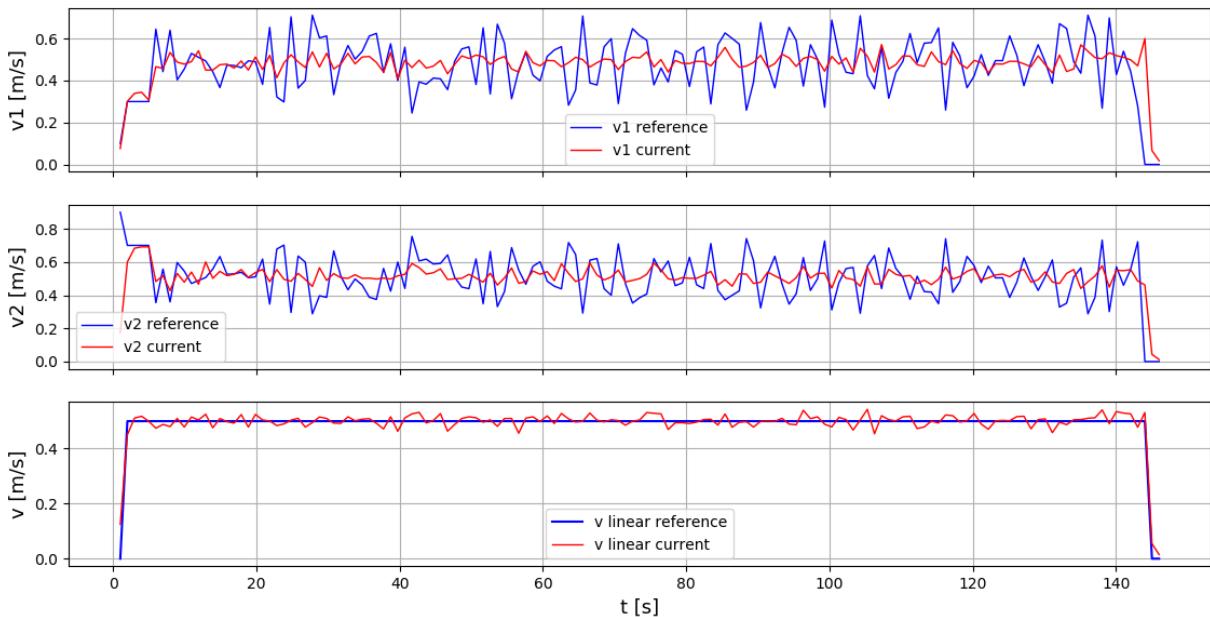
Bảng 5.5 Kết quả điều khiển bám quỹ đạo với $V = 0.3 \text{ m/s}$

Vận tốc đặt (m/s)	\bar{E} (m)	\bar{E}_θ (°)	\bar{V} (m/s)	\bar{E}_v (m/s)	\bar{V}_1 (m/s)	\bar{E}_{v1} (m/s)	\bar{V}_2 (m/s)	\bar{E}_{v2} (m/s)
0.3	0.0542	4.4452	0.2992	0.0091	0.3024	0.0352	0.2959	0.0347

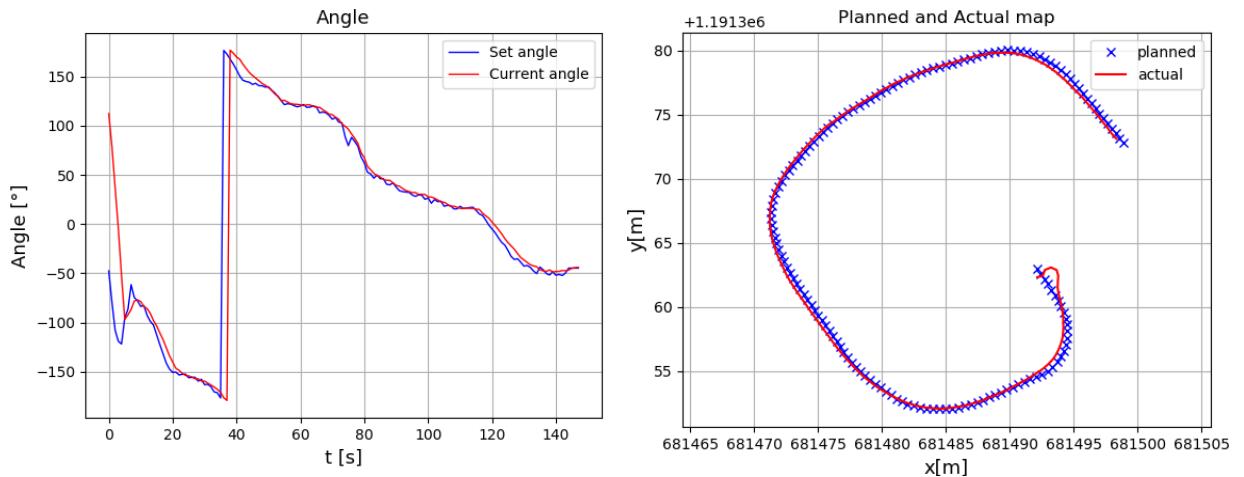
Trường hợp 3: $V = 0.5 \text{ (m/s)}$, $K = 0.8$, $K_{soft} = 0.001$



Hình 5.22 Đồ thị sai số quỹ đạo của robot, $V = 0.5$ (m/s)



Hình 5.23 Đồ thị đáp ứng vận tốc, $V = 0.5$ (m/s)



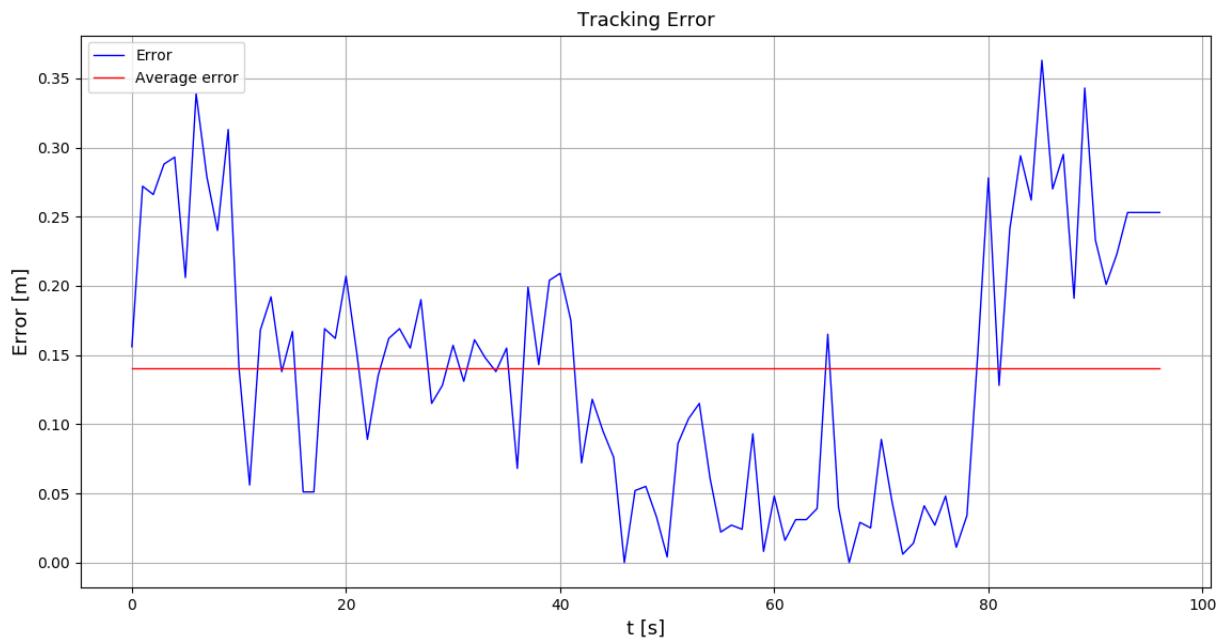
Hình 5.24 Đáp ứng góc và quỹ đạo từ A đến B, $V = 0.5 \text{ m/s}$

Đồ thị đáp ứng góc ngược với đồ thị góc tiếp tuyến quỹ đạo vì robot đang di chuyển từ B về A.

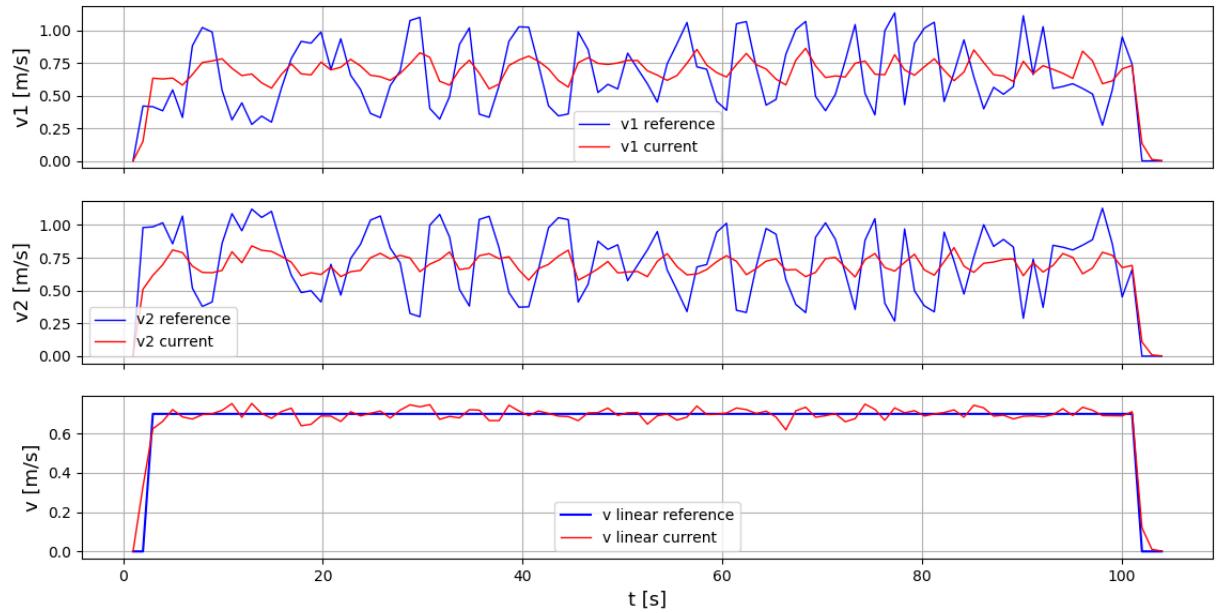
Bảng 5.6 Kết quả điều khiển bám quỹ đạo với $V = 0.5 \text{ m/s}$

Vận tốc đặt (m/s)	\bar{E} (m)	\bar{E}_θ (°)	\bar{V} (m/s)	\bar{E}_v (m/s)	\bar{V}_1 (m/s)	\bar{E}_{v1} (m/s)	\bar{V}_2 (m/s)	\bar{E}_{v2} (m/s)
0.5	0.0634	6.4994	0.5016	0.0157	0.4858	0.0857	0.5173	0.0919

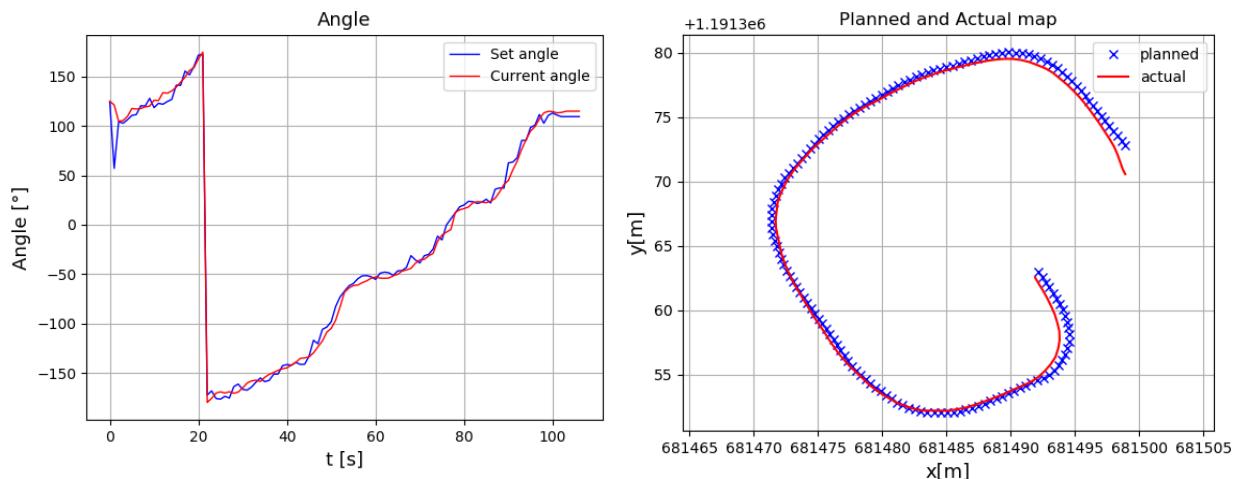
Trường hợp 4: $V = 0.7 \text{ (m/s)}$, $K = 0.8$, $K_{soft} = 0.001$



Hình 5.25 Đồ thị sai số quỹ đạo của robot, $V = 0.7$ (m/s)



Hình 5.26 Đồ thị đáp ứng vận tốc, $V = 0.7$ (m/s)

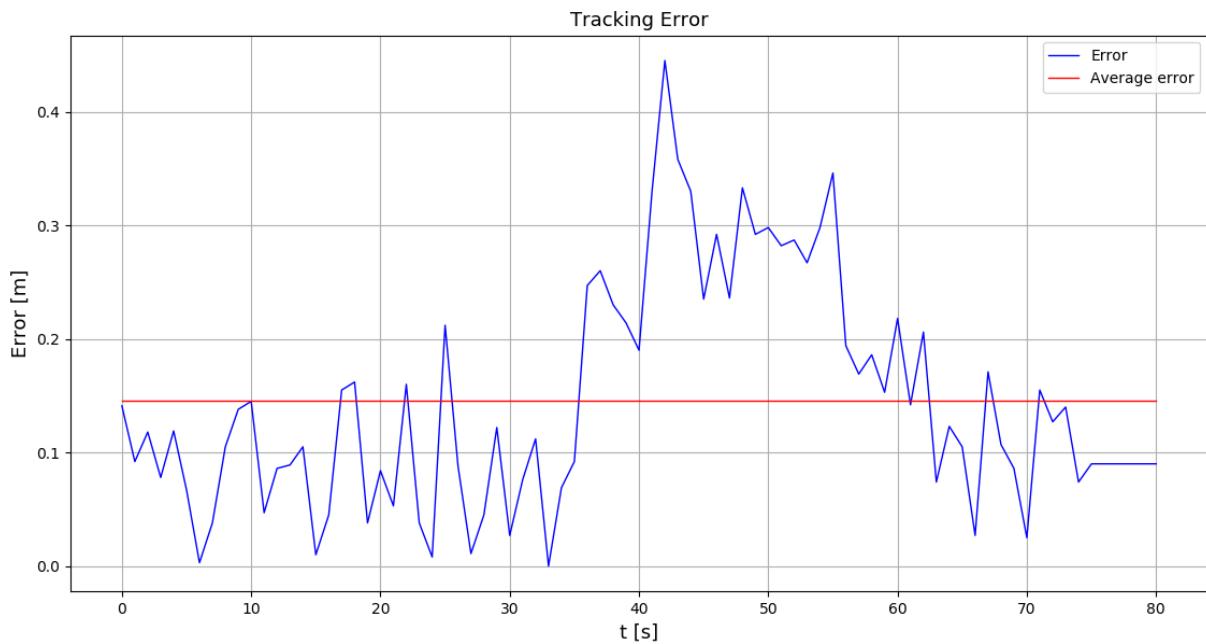


Hình 5.27 Đáp ứng góc và quỹ đạo từ A đến B, $V = 0.7 \text{ m/s}$

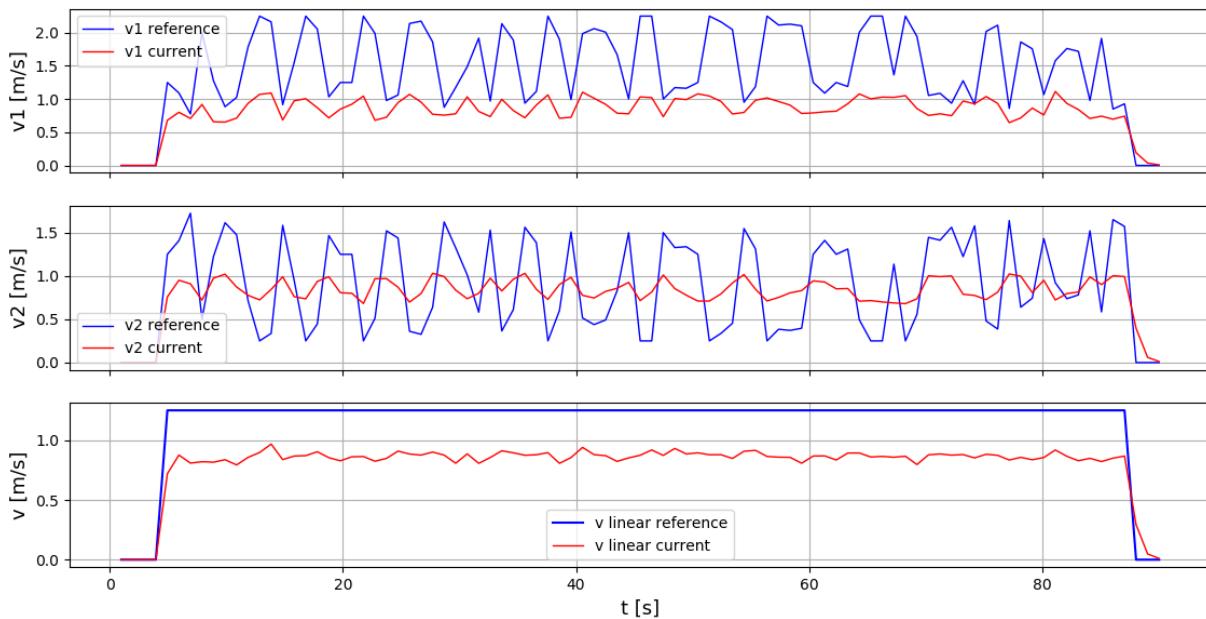
Bảng 5.7 Kết quả điều khiển bám quỹ đạo với $V = 0.7 \text{ m/s}$

Vận tốc đặt (m/s)	\bar{E} (m)	\bar{E}_θ (°)	\bar{V} (m/s)	\bar{E}_v (m/s)	\bar{V}_1 (m/s)	\bar{E}_{v1} (m/s)	\bar{V}_2 (m/s)	\bar{E}_{v2} (m/s)
0.7	0.1402	5.3252	0.6992	0.0245	0.6992	0.1972	0.6991	0.1947

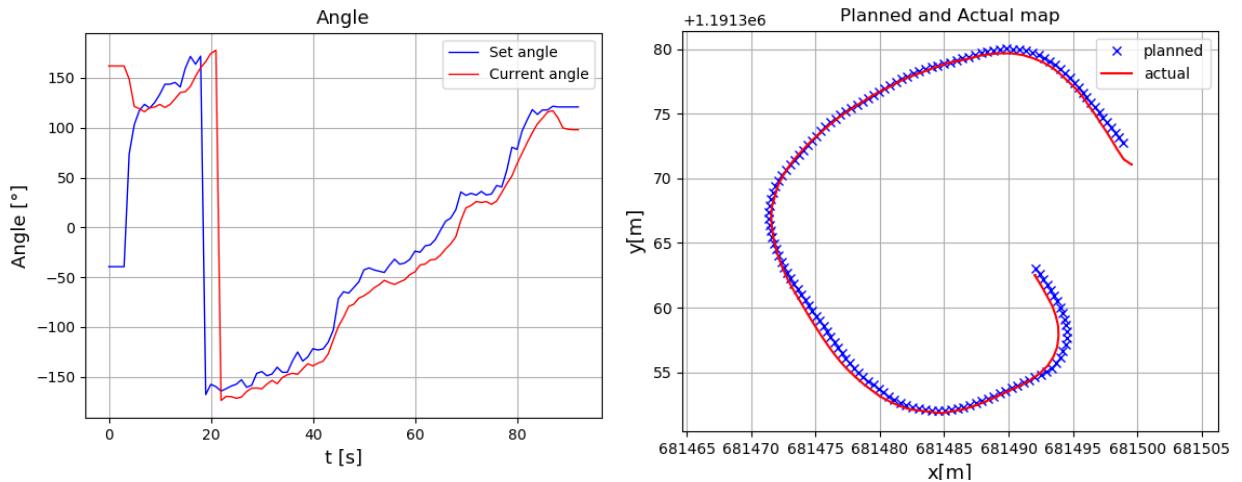
Trường hợp 5: $V = 1.25 \text{ (m/s)}$, $K = 0.8$, $K_{soft} = 0.001$



Hình 5.28 Đồ thị sai số quỹ đạo của robot, $V = 1.25$ (m/s)



Hình 5.29 Đồ thị đáp ứng vận tốc, $V = 1.25$ (m/s)



Hình 5.30 Đáp ứng góc và quỹ đạo từ A đến B, $V = 1.25m/s$

Nhận xét:

- Mặc dù vận tốc tối đa là $1.25 \left(\frac{m}{s}\right) \sim 140 \text{ (rpm)}$ nhưng robot đã đạt đến giới hạn công suất khi vận hành, với tải trọng robot khoảng 15kg, tốc độ tối đa đạt được trong khoảng $[0.8, 0.9] \left(\frac{m}{s}\right)$. Đồ thị cho thấy vận tốc đặt trung bình $v_1 > v_2$ thể hiện robot có xu hướng quẹo về bên trái.

Bảng 5.8 Kết quả điều khiển bám quỹ đạo với $V = 1.25 m/s$

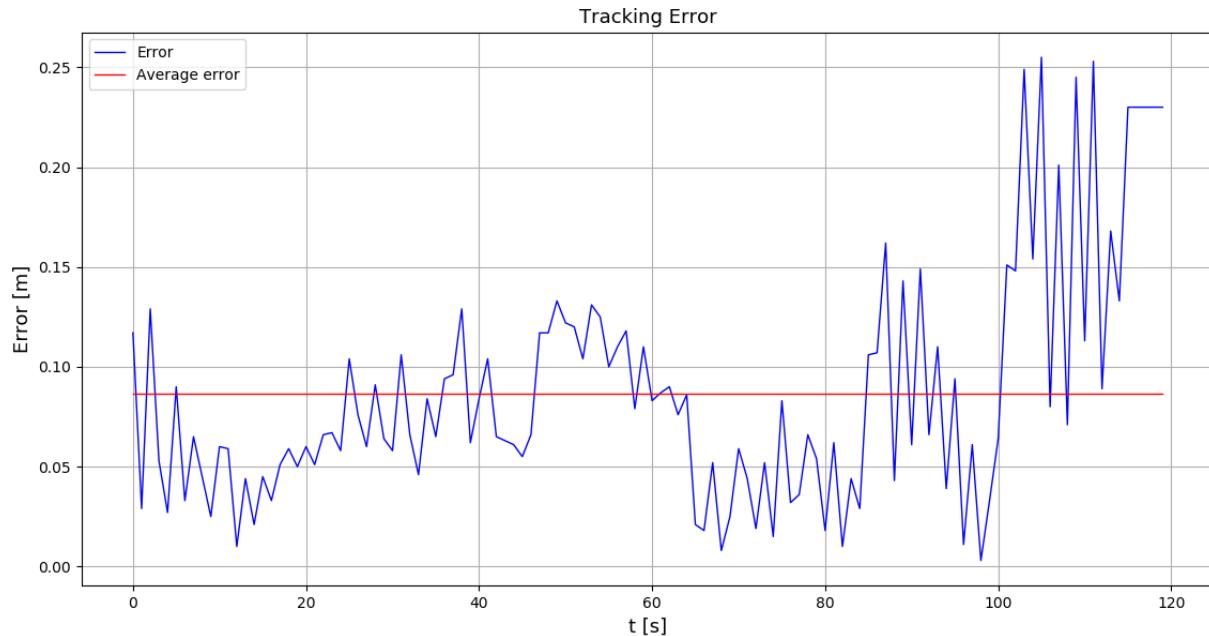
Vận tốc đặt (m/s)	\bar{E} (m)	\bar{E}_θ (°)	\bar{V} (m/s)	\bar{E}_v (m/s)	\bar{V}_1 (m/s)	\bar{E}_{v1} (m/s)	\bar{V}_2 (m/s)	\bar{E}_{v2} (m/s)
1.25	0.1453	22.104	0.8009	0.3607	0.8108	0.639	0.791	0.3995

Tổng hợp các kết quả điều khiển với vận tốc không đổi:

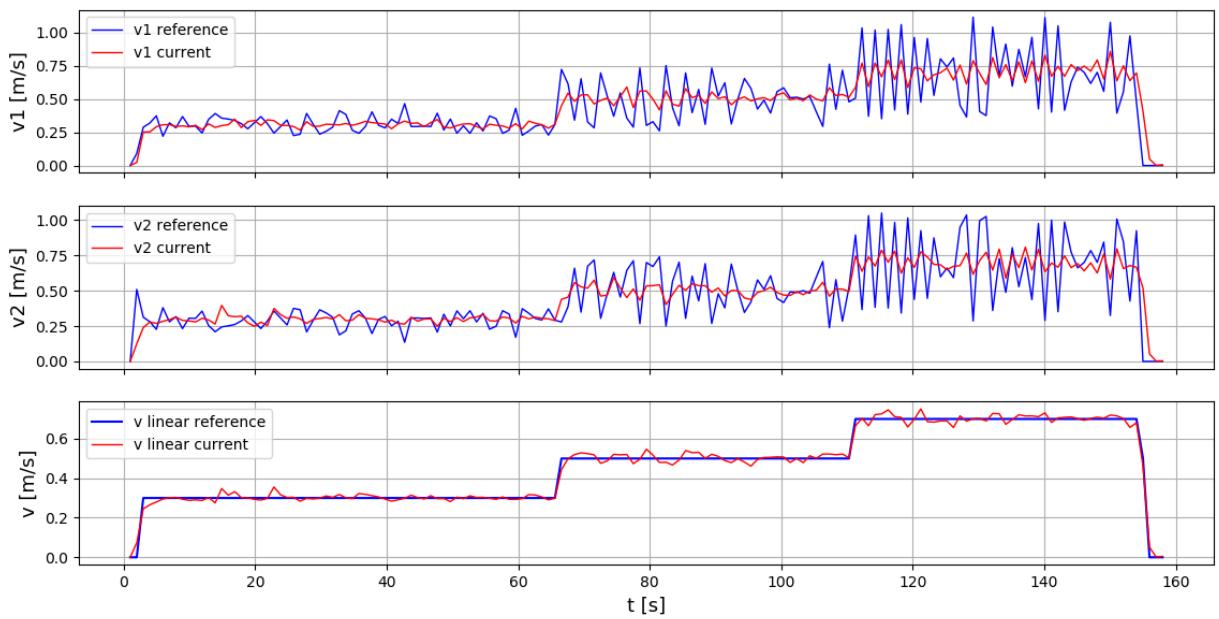
Bảng 5.9 Kết quả điều khiển bám quỹ đạo với vận tốc cố định

Vận tốc đặt (m/s)	\bar{E} (m)	\bar{E}_θ (o)	\bar{V} (m/s)	\bar{E}_v (m/s)	\bar{V}_1 (m/s)	\bar{E}_{v1} (m/s)	\bar{V}_2 (m/s)	\bar{E}_{v2} (m/s)
0.2	0.0079	4.5823	0.2011	0.0079	0.2037	0.0192	0.1985	0.0171
0.3	0.0542	4.4452	0.2992	0.0091	0.3024	0.0352	0.2959	0.0347
0.5	0.0634	6.4994	0.5016	0.0157	0.4858	0.0857	0.5173	0.0919
0.7	0.1402	5.3252	0.6992	0.0245	0.6992	0.1972	0.6991	0.1947
1.25	0.1453	22.104	0.8009	0.3607	0.8108	0.639	0.791	0.3995

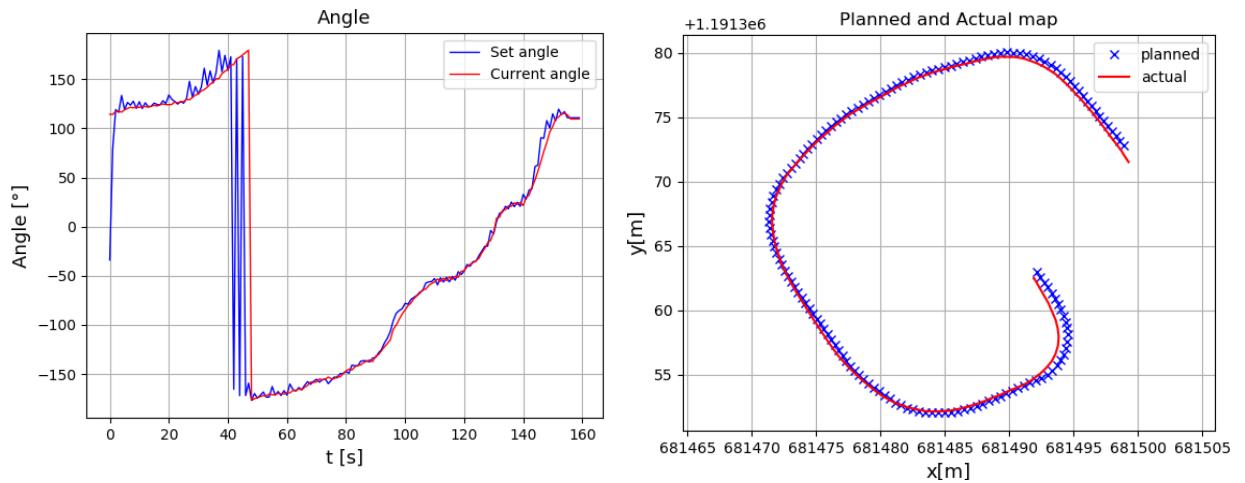
Trường hợp 6: Điều khiển với các cấp vận tốc thay đổi để khảo sát sự ổn định khi thay đổi tốc độ $V_1 = 0.3$ (m/s), $V_2 = 0.5$ (m/s), $V_3 = 0.7$ (m/s), $K = 0.8$, $K_{soft} = 0.001$.



Hình 5.31 Đồ thị sai số quỹ đạo của robot, $V = [0.3, 0.5, 0.7]$ (m/s)



Hình 5.32 Đồ thị đáp ứng vận tốc, $V = [0.3, 0.5, 0.7]$ (m/s)

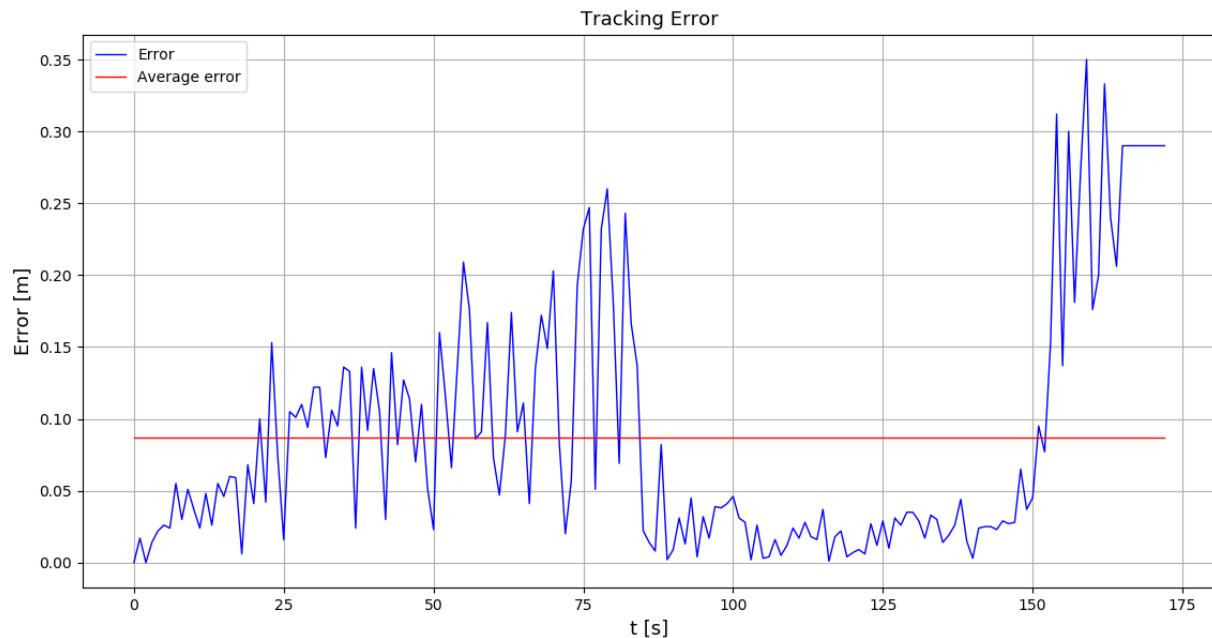


Hình 5.33 Dáp ứng góc và quỹ đạo từ A đến B, $V = [0.3, 0.5, 0.7]$ m/s

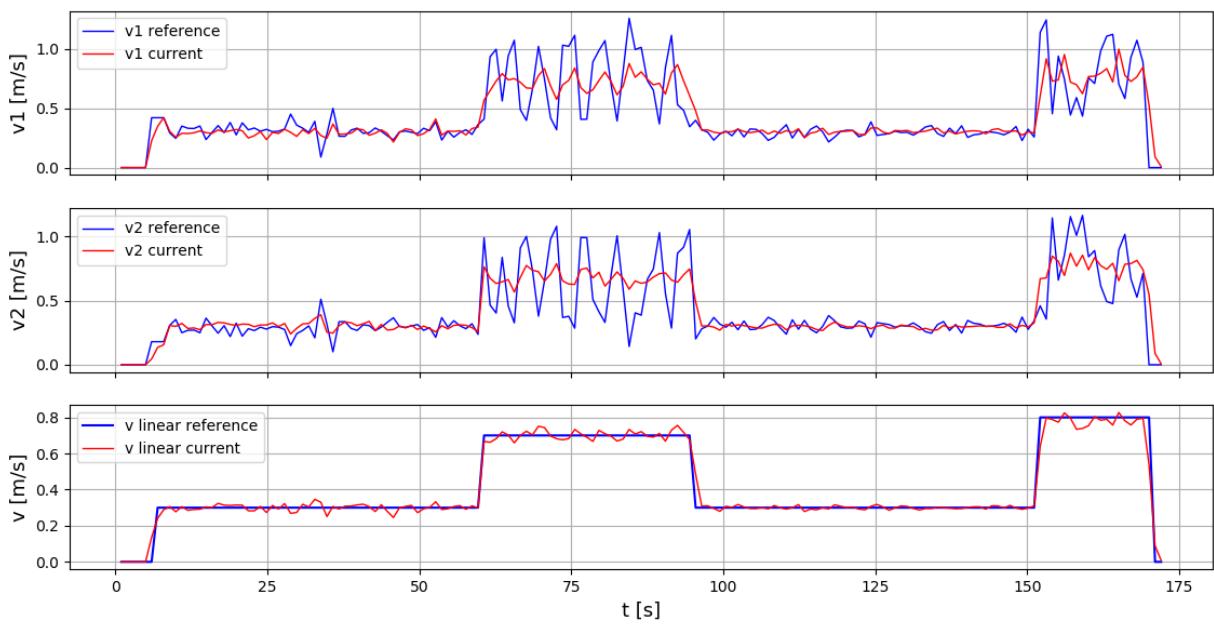
Bảng 5.10 Kết quả điều khiển bám quỹ đạo, $V = [0.3, 0.5, 0.7] \text{ (m/s)}$

Vận tốc đặt (m/s)	\bar{E} (m)	\bar{E}_θ (°)	\bar{V} (m/s)	\bar{E}_v (m/s)	\bar{V}_1 (m/s)	\bar{E}_{v1} (m/s)	\bar{V}_2 (m/s)	\bar{E}_{v2} (m/s)
0.3	0.08677	5.5656	0.3008	0.0116	0.3044	0.0442	0.2971	0.0469
0.5			0.5038	0.0182	0.5074	0.1116	0.4990	0.1075
0.7			0.6945	0.0219	0.6989	0.1856	0.6939	0.1935

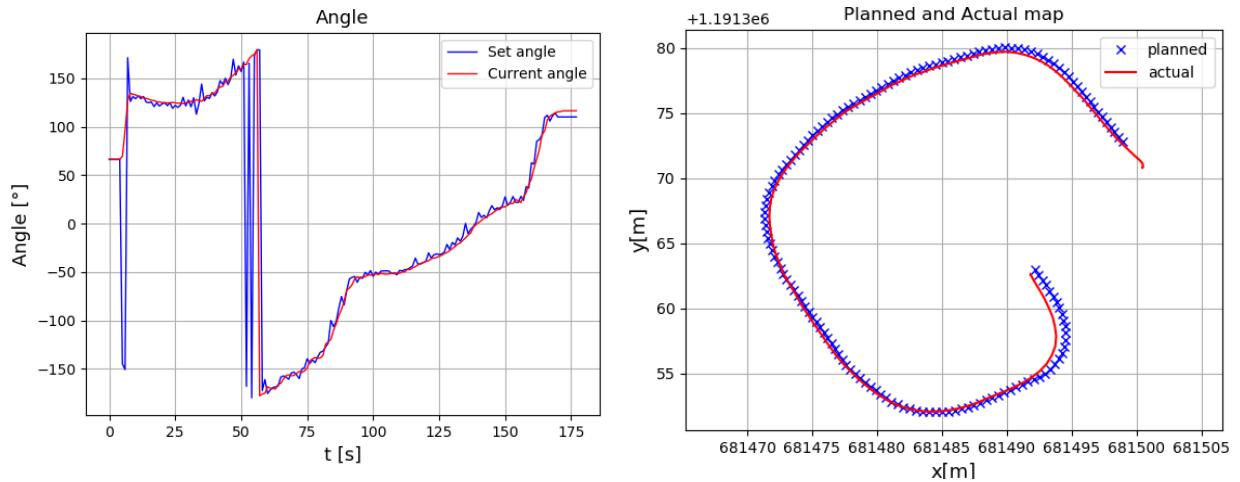
Trường hợp 7: Điều khiển với các cấp vận tốc thay đổi $V_1 = 0.3 \text{ (m/s)}$, $V_2 = 0.7 \text{ (m/s)}$, $V_3 = 0.3 \text{ (m/s)}$, $V_4 = 0.8 \text{ (m/s)}$, $K = 0.8$, $K_{soft} = 0.001$



Hình 5.34 Đồ thị sai số quỹ đạo của robot, $V = [0.3, 0.7, 0.3, 0.8] \text{ (m/s)}$



Hình 5.35 Đồ thị đáp ứng vận tốc, $V = [0.3, 0.7, 0.3, 0.8]$ (m/s)



Hình 5.36 Đáp ứng góc và quỹ đạo từ A đến B, $V = [0.3, 0.7, 0.3, 0.8]$ m/s

Bảng 5.11 Kết quả điều khiển bám quỹ đạo, $V = [0.3, 0.7, 0.3, 0.8]$ (m/s)

Vận tốc đặt (m/s)	\bar{E} (m)	\bar{E}_θ (°)	\bar{V} (m/s)	\bar{E}_v (m/s)	\bar{V}_1 (m/s)	\bar{E}_{v1} (m/s)	\bar{V}_2 (m/s)	\bar{E}_{v2} (m/s)
0.3	0.0872	5.8066	0.2987	0.0166	0.3008	0.0378	0.2967	0.0383
0.7			0.7010	0.0217	0.7188	0.2271	0.6833	0.2240
0.3			0.3036	0.0103	0.3061	0.0281	0.3010	0.0321
0.8			0.7615	0.0447	0.7636	0.2246	0.7594	0.2099

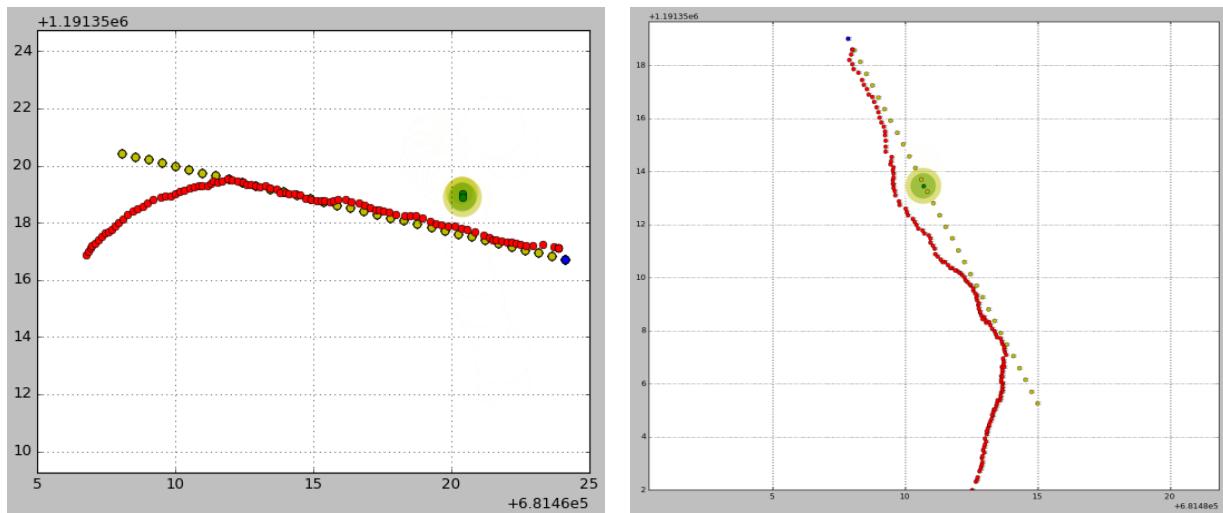
5.2.5. Kết quả điều khiển mô hình tránh vật cản

Với tần số gửi thông tin về vật cản của Object Detection node là 17Hz và tần số gửi thông tin về mô hình của SerialPort node là 20Hz, mô hình có thể tính toán vận tốc và góc lái để tránh vật với tốc độ 15Hz.

Kết quả được vẽ lại từ số liệu thu thập được trong quá trình thực hiện, với các điểm tròn:

- Màu vàng: bản đồ được gửi từ giao diện.
- Màu xanh dương: đích đến.
- Màu xanh lá: vật cản phát hiện được. Vòng tròn màu xanh lá có bán kính bằng bán kính của vật côn với bán kính của mô hình. Vòng tròn màu vàng có bán kính bằng bán kính vòng tròn màu xanh lá cộng với khoảng an toàn.
- Màu đỏ: đường đi của mô hình.

Các giá trị x, y được tính trong hệ tọa độ UTM.



Hình 5.37 Một số kết quả điều khiển tránh vật cản

Nhận xét:

- Thuật toán tránh vật cản hoạt động khá tốt. Khi phát hiện vật cản ở vị trí cách xa bán đồ đồ được nạp sẵn, mô hình tiếp tục bám quỹ đạo. Mô hình chỉ quyết định ra khỏi quỹ đạo để tránh vật cản khi phát hiện vật ở vị trí va chạm với quỹ đạo.
- Việc xác định đích đến tạm thời đảm bảo mô hình không lập tức quay trở lại quỹ đạo khi vừa mới tránh được vật cản và không phát hiện được vật cản đó nữa do giới hạn của tầm quan sát của camera. Tuy nhiên mô hình vẫn có xu hướng quay trở lại quỹ đạo, phát hiện lại vật cản và tiếp tục di chuyển ra xa quỹ đạo. Điều này đôi khi làm cho mô hình di chuyển theo đường zigzag trong một khoảng thời gian ngắn khi đang tránh vật.

Chương 6. ĐÁNH GIÁ, KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. Đánh giá kết quả

- Phần cứng

Mô hình cơ khí sử dụng hầu hết kim loại nhôm, sắt cố định và mica đảm bảo sự chắc chắn, kết hợp thiết kế gọn gàng mang tính thẩm mỹ cao.

Tuy nhiên bánh xe còn khá nhỏ, cần thay bánh to để xe chạy được nhanh hơn, dùng bánh xe cao su để robot ổn định hơn khi đi qua các bãy đá.

Ngoài ra tốc độ cập nhật dữ liệu từ module GPS hiện tại là hạn chế (1Hz) so với tần số lấy mẫu (20Hz), nghĩa là trong 20 chu kỳ điều khiển chỉ có 1 chu kỳ nhận được tọa độ từ GPS, các chu kỳ còn lại phải từ tính toán vị trí từ vận tốc làm gia tăng sai số.

- Bộ điều khiển bám quỹ đạo Stanley Controller

Đã tối ưu từng tính toán xử lý trong vi điều khiển giúp giảm thời gian tính toán, đồng thời khắc phục hầu hết các lỗi

Độ chính xác được cải thiện tốt hơn với 3cm sai số bám quỹ đạo và vẫn còn khả năng tốt hơn nữa.

- Phát hiện, theo dõi và xác định vị trí của vật cản

Do hạn chế về phần cứng (máy tính nhúng không có GPU, tốc độ xử lý chậm), các thuật toán phát hiện, theo dõi và xác định vị trí của vật cản được lựa chọn dựa trên tốc độ xử lý nhanh, nhưng bù lại độ chính xác chưa phải là cao nhất. Ví dụ khi lựa chọn mô hình phát hiện đối tượng Faster R – CNN được huấn luyện trên tập dữ liệu VOC0712 với độ chính xác mAP = 82.0% (cao hơn mô hình MobileNet SSD đang sử dụng với mAP = 72.7%), kết hợp với bộ theo dõi MOSSE, tốc độ xử lý chưa đạt đến 5FPS, chậm hơn rất nhiều so với tốc độ hiện tại của mô hình MobileNet SSD là 17FPS.

Tuy nhiên, đối với mô hình xe tự hành trong đề tài, tốc độ di chuyển chậm và chỉ xem xét các vật cản ở khoảng cách gần, các thuật toán được lựa chọn vẫn đáp ứng đủ nhu cầu phát hiện, theo dõi và xác định vị trí vật cản của mô hình.

- **Điều khiển mô hình tránh vật cản**

Với thuật toán tránh vật cản được xây dựng dựa trên phương pháp The Curvature – Velocity Method, mô hình thực hiện tốt việc tránh các vật cản đứng yên. Tuy nhiên, đối với các vật cản chuyển động, do thuật toán chưa xét đến vận tốc của các vật cản, trong một số trường hợp mô hình sẽ di chuyển chắn phía trước hướng chuyển động của vật cản gây mất an toàn.

Cùng với vấn đề trên, việc xác định làn đường với khoảng cách cố định chưa đáp ứng được nhu cầu của xe tự hành trong thực tế.

- **Giao diện người dùng**

Giao diện được viết lại tối ưu hơn về phần xử lý bên trong (backend), khắc phục hầu hết các lỗi và hoạt động ổn định.

Phần trải nghiệm người dùng (front-end) được nâng cấp thẩm mỹ hơn.

Dữ liệu nhận được từ vi điều khiển nhiều cho đánh giá chi tiết, đồng thời tăng khả năng giám sát từ xa.

- **Phân tích và đánh giá dữ liệu**

Các dữ liệu điều khiển như vị trí robot, quỹ đạo, vận tốc robot từng bánh, góc lái, các giá trị điều khiển Stanley, ... đều được thu thập và lưu lại file với định dạng json giúp dễ thêm bớt trường dữ liệu và xử lý dữ liệu.

Chương trình Python đánh giá dữ liệu được viết để tự động hóa quy trình đánh giá sai số, chỉ cần nhập file dữ liệu input và các options, sai số và đồ thị sẽ được tính toán tự động.

6.2. Kết luận

Tuy vẫn còn một số hạn chế chưa được khắc phục trong quá trình thực nghiệm (như nhiễu GPS, thuật toán tránh vật cản chưa xét đến vận tốc, ...), mô hình đã cơ bản đáp ứng các mục tiêu của đề tài:

- Xây dựng mô hình xe tự hành chạy ngoài trời, sử dụng GPS để bám quỹ đạo cho trước với tốc độ xử lý cao và sai số tương đối nhỏ.
- Phát hiện vật cản và ra quyết định ra khỏi quỹ đạo để tránh khi cần thiết với thông tin từ stereo camera, đảm bảo độ an toàn cho mô hình.
- Phân loại được vật cản vào các lớp (gồm 20 lớp vật cản): người, xe, ô tô, ...
- Xây dựng giao diện người dùng trực quan, dễ sử dụng.

6.3. Hướng phát triển

- Sử dụng các giải thuật bám quỹ đạo như: MPC, Vector Field Histogram.
- Sử dụng bộ lọc xung hữu hạn – FIR để loại bỏ nhiễu lượng tử encoder.
- Tích hợp GPS/INS để có thể chạy ở mode GPS single, không cần dùng đến trạm base.
- Sử dụng các hệ thống Cloud như MQTT Broker, đồng thời thêm wifi router đưa tất cả dữ liệu lên Internet, điều khiển và truy cập từ bất cứ nơi đâu thay vì sử dụng Lora mang phạm vi local.
- Sử dụng database như NoSQL, MySQL trên giao diện và máy tính nhúng để lưu dữ liệu tối ưu và chuyên nghiệp.
- Lắp đặt thêm GPU cho máy tính nhúng để sử dụng các thuật toán phát hiện, theo dõi và xác định vị trí tốt hơn nhằm tăng độ chính xác cho mô hình nhưng vẫn đảm bảo về tốc độ xử lý.
- Phần cứng mô hình còn nhiều hạn chế khi thực nghiệm ở môi trường thực tế như đi qua những bãi đá gồ ghề, kéo theo sự rung lắc antenna GPS và làm rung camera.

TÀI LIỆU THAM KHẢO

- [1] N. A. Dũng, "<https://thethaovanhoa.vn/>," 31 01 2017. [Online].
- [2] M. Tâm, "<https://baotintuc.vn/>," 16 04 2020. [Online].
- [3] N. Phạm, "<https://www.24h.com.vn/>," 03 12 2014. [Online].
- [4] R. Simmons, "The curvature-velocity method for local obstacle avoidance," 1996.
- [5] Andrew G., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision," 2017.
- [6] D. S. Bolme, "Visual Object Tracking using Adaptive Correlation Filters," 2010.
- [7] W. Liu, "SSD: Single Shot MultiBox Detector," 2016.
- [8] G. M. Hoffmann, "Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing," 2007.