

**ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH**



LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC

**HIỆN THỰC CHỨC NĂNG NHẬN DIỆN
LÀN ĐƯỜNG CHO XE TỰ HÀNH**

Ngành : KỸ THUẬT MÁY TÍNH

**HỘI ĐỒNG : KỸ THUẬT MÁY TÍNH 2
GVHD : ThS. TRẦN THANH BÌNH
GVPB : ThS. HUỲNH HOÀNG KHA**

---o0o---

**SVTH : HUỲNH TRƯƠNG QUỐC KHÁNH
(1810990)**

TP. HỒ CHÍ MINH, 10/2023

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA: KH & KT Máy tính
BỘ MÔN: KTMT**NHIỆM VỤ LUẬN VĂN/ ĐỒ ÁN TỐT NGHIỆP**
Chú ý: Sinh viên phải dán tờ này vào trang nhất của bản thuyết trình

HỌ VÀ TÊN: Huỳnh Trương Quốc Khanh

MSSV: 1810990

NGÀNH: Kỹ Thuật Máy Tính

LỚP: MT18KT

1. Đầu đề luận văn/ đồ án tốt nghiệp: HIỆN THỰC CHỨC NĂNG NHẬN DIỆN LÀN ĐƯỜNG CHO XE TỰ HÀNH.**2. Nhiệm vụ (yêu cầu về nội dung và số liệu ban đầu):**

- ❖ Bấm làn đường dùng camera và xử lý ảnh.
- ❖ Hiện thực giải thuật nhận diện đèn giao thông dùng camera và xử lý ảnh.
- ❖ Hiện thực giải thuật nhận diện biển báo giao thông dùng camera và học máy.
- ❖ Thủ nghiệm bằng cách sử dụng công cụ mô phỏng việc phát hiện làn đường (robot trong mô phỏng là turtlebot 3)
- ❖ Viết báo cáo.

3. Ngày giao nhiệm vụ: 30/06/2023**4. Ngày hoàn thành nhiệm vụ:** 18/10/2023**5. Họ tên giảng viên hướng dẫn:**

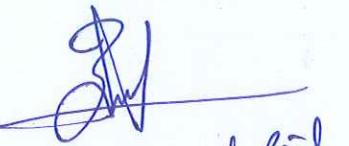
1) ThS. Trần Thanh Bình

Phần hướng dẫn:

100%

Nội dung và yêu cầu LVTN/ ĐATN đã được thông qua Bộ môn.

Ngày tháng năm

CHỦ NHIỆM BỘ MÔN
(Ký và ghi rõ họ tên)GIẢNG VIÊN HƯỚNG DẪN CHÍNH
(Ký và ghi rõ họ tên)

 Trần Thanh Bình
PHẦN DÀNH CHO KHOA, BỘ MÔN:

Người duyệt (chấm sơ bộ):

Đơn vị:

Ngày bảo vệ:

Điểm tổng kết:

Nơi lưu trữ LVTN/ĐATN:

Ngày 27 tháng 09 năm 2023

PHIẾU CHẤM BẢO VỆ LVTN
(Dành cho người hướng dẫn)

1. Họ và tên SV: Huỳnh Trương Quốc Khanh

MSSV: 1810990

Ngành (chuyên ngành): Kỹ Thuật Máy Tính

2. Đề tài: HIỆN THỰC CHỨC NĂNG NHẬN DIỆN LÀN ĐƯỜNG CHO XE TỰ HÀNH.

3. Họ tên người hướng dẫn: Ths. Trần Thanh Bình.

4. Tổng quát về bản thuyết minh:

Số trang: 41

Số chương: 05

Số bảng số liệu: 00

Số hình vẽ: 48

Số tài liệu tham khảo: 05

Phần mềm tính toán:

Hiện vật (sản phẩm)

5. Tổng quát về các bản vẽ:

- Số bản vẽ:

Bản A1:

Bản A2:

Khô khác:

- Số bản vẽ vẽ tay

Số bản vẽ trên máy tính:

6. Những ưu điểm chính của LVTN:

- Tìm hiểu hệ điều hành ROS, Gazebo để mô phỏng xe tự hành.
- Sử dụng OpenCV để phát hiện làn đường, mô hình CNN để nhận diện biển báo giao thông, và mô phỏng thành công việc di chuyển theo làn đường.

7. Những thiếu sót chính của LVTN:

- Kỹ thuật nhận diện làn đường còn đơn giản, quá phụ thuộc vào threshold, không phân biệt được các làn đường đa dạng, các làn đường phức tạp.
- Kết quả thử nghiệm còn ít, báo cáo còn đơn giản.

8. Đề nghị: Được bảo vệ

Bổ sung thêm để bảo vệ

Không được bảo vệ

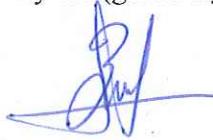
9. câu hỏi SV phải trả lời trước Hội đồng:

- a. Nếu tiếp tục phát triển, sinh viên sẽ cải tiến những điểm gì để giúp việc nhận diện chính xác và nhận diện trên nhiều dạng làn đường hơn? Giải thích?

10. Đánh giá chung (bằng chữ: giỏi, khá, TB): TB

Điểm : 6.0 /10

Ký tên (ghi rõ họ tên)



Trần Thanh Bình

Ngày 03 tháng 10 năm 2023

PHIẾU ĐÁNH GIÁ LUẬN VĂN TỐT NGHIỆP
(Dành cho người phản biện)

1. Họ và tên SV: Huỳnh Trương Quốc Khanh
MSSV: 1810990

Ngành (chuyên ngành): Kỹ Thuật Máy Tính

2. Đề tài: HIỆN THỰC CHỨC NĂNG NHẬN DIỆN LÀN ĐƯỜNG CHO XE TỰ HÀNH.

3. Họ tên người phản biện: Huỳnh Hoàng Kha

4. Tổng quát về bản thuyết minh:

Số trang:

Số chương:

Số bảng số liệu

Số hình vẽ:

Số tài liệu tham khảo:

Phần mềm tính toán:

Hiện vật (sản phẩm)

5. Những ưu điểm chính của LV/ĐATN:

- Sinh viên có tìm hiểu và sử dụng được ROS, Gazebo.
- Các bước giải quyết vấn đề tương đối hợp lý.

6. Những thiếu sót chính của LV/ĐATN:

- Hàm lượng kiến thức tìm hiểu, nghiên cứu còn hạn chế.
- Chỉ đưa ra một hướng giải quyết vấn đề duy nhất và không có luận giải phù hợp.
- Kết quả thử nghiệm còn ít, báo cáo sơ sài.

7. Đề nghị: Được bảo vệ Bổ sung thêm để bảo vệ Không được bảo vệ

8. Các câu hỏi SV phải trả lời trước Hội đồng:

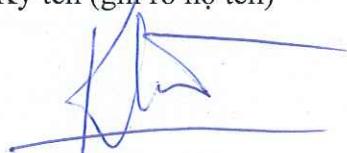
a. Tốc độ tối đa của robot là bao nhiêu?

b. Ưu và nhược điểm của giải pháp được đề xuất là gì?

c. Các tiêu chí đánh giá kết quả thử nghiệm là gì, quá trình đánh giá như thế nào?

9. Đánh giá chung (bằng chữ: Xuất sắc, Giỏi, Khá, TB): Trung bình Đánh giá : 5.5 /10

Ký tên (ghi rõ họ tên)



Huỳnh Hoàng Kha

Lời cam đoan

Em xin cam đoan rằng luận văn tốt nghiệp: "**HIỆN THỰC CHỨC NĂNG NHẬN DIỆN LÀN ĐƯỜNG CHO XE TỰ HÀNH**" là kết quả nghiên cứu do em thực hiện dưới sự hướng dẫn của thầy ThS. Trần Thanh Bình. Những tài liệu tham khảo, nội dung trích dẫn được trình bày chi tiết, cụ thể. Còn lại những nội dung khác chưa từng được công bố hoặc sử dụng để nhận bằng cấp ở những nơi khác.

Nếu phát hiện có bất kỳ sự gian lận nào, em xin hoàn toàn chịu trách nhiệm về nội dung luận văn tốt nghiệp của mình. Trường Đại học Bách Khoa TP. Hồ Chí Minh không liên quan đến những vi phạm (nếu có) về tác quyền, bản quyền do em gây ra trong quá trình thực hiện.

TP.Hồ Chí Minh, ngày 15 tháng 10 năm 2023

Sinh viên thực hiện

Huỳnh Trương Quốc Khanh

Lời cảm ơn

Để hoàn thành luận văn tốt nghiệp này, em xin gửi lời cảm ơn tới:

Quý thầy, cô trong Khoa Khoa học và Kỹ thuật máy tính, bộ môn Kỹ thuật máy tính, trường Đại học Bách Khoa TP. Hồ Chí Minh, những người đã hết lòng truyền đạt những kiến thức quý báu cho chúng em trong suốt thời gian học tập tại trường. Đặc biệt, em xin gửi lời cảm ơn chân thành và trân trọng nhất đến thầy **ThS. Trần Thanh Bình** đã tận tình hướng dẫn, giúp đỡ em trong suốt quá trình nghiên cứu và thực hiện đề tài. Ngoài ra, em xin được gửi lời cảm ơn chân thành đến thầy **TS. Nguyễn Trần Hữu Nguyên** - GVCN lớp Kỹ thuật máy tính khoá 2018 đã giúp đỡ, định hướng cho chúng em trong quá trình học tập, những kiến thức được các thầy cô truyền đạt là hành trang quý báu cho chúng em trên con đường học tập, công tác và nghiên cứu sau này. Xin gửi lời cảm ơn chân thành đến tất cả các bạn lớp Kỹ thuật máy tính khoá 2018 đã luôn giúp đỡ và hỗ trợ trong suốt quá trình học tập tại trường.

Xin chân thành cảm ơn!

Tóm tắt nội dung

Trong quá trình thực hiện đề tài luận văn tốt nghiệp kỹ thuật máy tính **Hiện thực chức năng nhận diện làn đường cho xe tự hành**, em đã tìm hiểu các kiến thức liên quan đến xe tự hành cũng như các hệ thống đã được triển khai. Từ đó nghiên cứu, chọn lọc và bổ sung các chức năng cần thiết để tạo ra một hệ thống phù hợp với yêu cầu mà đề tài đặt ra. Hệ thống là sự tích hợp của các thành phần như: module nhận diện làn đường, module phát hiện và nhận diện biển báo giao thông, module điều khiển robot.

Luận văn tốt nghiệp này tập trung vào việc nghiên cứu và mô phỏng hệ thống nhận diện làn đường cho xe tự hành có khả năng nhận diện và di chuyển theo làn đường và biển báo giao thông sử dụng Hệ điều hành robot (ROS) và công cụ xử lý ảnh OpenCV. Phần phát hiện và nhận diện biển báo giao thông được thực hiện với sự hỗ trợ của cấu trúc mạng Convolutional Neural Network (CNN). Hệ thống được đánh giá thử nghiệm trên môi trường mô phỏng sử dụng phần mềm Gazebo và robot Turtlebot3.

Mục lục

Lời cam đoan	i
Lời cảm ơn	ii
Tóm tắt nội dung	iii
Mục lục	iv
Danh sách hình vẽ	vii
1 Giới thiệu đề tài	1
1.1 Lý do và động lực thực hiện đề tài	1
1.2 Mục tiêu của đề tài	1
1.2.1 Về kiến thức	1
1.2.2 Về kết quả thực tế	2
1.3 Phạm vi nghiên cứu	2
1.4 Ý nghĩa thực tiễn	3
1.5 Bố cục của luận văn	3
2 Cơ sở lý thuyết	5
2.1 Robot Operating System (ROS)	5
2.2 Thị giác máy tính (Computer Vision)	7
2.2.1 Giới thiệu về Thị giác máy tính	7
2.2.2 Giới thiệu về OpenCV	8
2.2.3 So sánh bài toán phát hiện đối tượng và phân loại đối tượng (Object Detection vs Classification)	8
2.3 Machine Learning	9
2.3.1 Giới thiệu về Machine Learning	9
2.3.2 Giới thiệu về Deep Learning	10
2.3.3 Giới thiệu về Convolutional Neural Network (CNN)	11
2.3.4 Neuron	12
2.4 Gazebo	13
2.5 Roboflow	14
2.6 Camera	15

3 Đề xuất giải pháp và thiết kế hệ thống	17
3.1 Các phương pháp phát hiện làn đường phổ biến	17
3.2 Đánh giá thuật toán nhận diện làn đường dựa trên Sliding Window	18
3.2.1 Ưu điểm:	19
3.2.2 Nhược điểm:	19
3.3 Thuật toán phát hiện làn đường đề xuất	20
3.3.1 Camera Calibration	20
3.3.2 Các phương pháp Thresholding	21
3.3.3 Sobel Thresholding	22
3.3.4 Perspective Transformation và Bird's Eye View	24
3.3.5 Phát hiện các làn đường	26
3.3.6 Đặt sliding window để phát hiện pixel trắng	27
3.3.7 Đặt đa thức bậc hai vào các điểm thuộc làn đường	28
3.3.8 Xác định độ lệch vị trí của robot	30
3.4 Thuật toán nhận diện biển báo giao thông	31
3.4.1 YOLO (You Only Look Once)	31
3.4.2 Quá trình phát triển YOLO	32
3.4.3 Kiến trúc và đặc điểm	32
3.4.4 So sánh YOLO và một số mô hình CNN khác trong bài toán phát hiện vật thể	34
3.5 Bộ dữ liệu dành cho huấn luyện model	36
3.5.1 GTSDB (German Traffic Sign Detection Benchmark)	36
3.5.2 GTSRB (German Traffic Sign Recognition Benchmark)	37
3.6 PID Controller cho module điều khiển	37
3.7 Áp dụng PID để điều khiển Turtlebot3	39
4 Hiện thực hệ thống	41
4.1 TurtleBot3	41
4.2 Mô phỏng	42
4.3 Module nhận diện làn đường	44
4.4 Module phát hiện biển báo	45
4.4.1 Đánh giá model	47
4.5 Module phân loại biển báo	48
4.6 Module điều khiển	51
4.6.1 Tinh chỉnh PID	53
5 Kết quả thực nghiệm	55
5.1 Module nhận diện làn đường	55
5.2 Module phát hiện và phân loại biển báo giao thông	58
5.3 Module điều khiển	61
5.4 Một số thông số khi thực nghiệm tích hợp hệ thống	64

6	Tổng kết đề tài	65
6.1	Kết quả đạt được và hạn chế	65
6.1.1	Kết quả đạt được	65
6.1.2	Hạn chế	65
6.2	Thuận lợi và khó khăn khi thực hiện đề tài	66
6.2.1	Thuận lợi	66
6.2.2	Khó khăn	66
6.3	Hướng phát triển đề tài	66
	Tài liệu tham khảo	68

Danh sách hình vẽ

2.1	Robot Operating System	5
2.2	Sơ đồ hệ thống publish-subscribe message	6
2.3	Bài toán nhận diện vật thể trong Thị giác máy tính	7
2.4	OpenCV	8
2.5	Các vấn đề trong Thị giác máy tính	9
2.6	Machine Learning và Deep Learning	10
2.7	Deep Learning	11
2.8	Cấu trúc của một CNN	11
2.9	Neuron trong neural network	12
2.10	Gazebo	13
2.11	Roboflow	14
2.12	Raspberry Pi camera	15
2.13	Bốn hệ toạ độ của camera	16
3.1	Bài toán nhận diện làn đường	17
3.2	Kỹ thuật Sliding window	18
3.3	Camera Calibration	20
3.4	Radial distortion	21
3.5	Tangential distortion	21
3.6	Kênh màu RGB và HSL	21
3.7	Color Thresholding	22
3.8	Binary Thresholding	22
3.9	Sobel Thresholding với các ngưỡng khác nhau	23
3.10	Kernel	24
3.11	Kernel	24
3.12	directional Sobel	24
3.13	Làn đường theo góc nhìn camera	25
3.14	Perspective Transformation	26
3.15	Vùng quan tâm (ROI)	26
3.16	Bird-eye's view	27
3.17	Histogram	27
3.18	Sliding window và biểu diễn làn đường	28

3.19 Biểu diễn các làn đường	29
3.20 Xác định tâm làn đường	30
3.21 Cấu trúc mạng YOLO	31
3.22 Kiến trúc cập nhật của YOLOv8	33
3.23 So sánh một số CNN	35
3.24 So sánh một số CNN với mAP là độ chính xác trung bình	35
3.25 Hình ảnh đường giao thông với các biển báo trong GTSDB	36
3.26 Hình ảnh biển báo trong GTSRB	37
3.27 Bộ điều khiển PID	38
3.28 TurtleBot3 và các vận tốc	39
 4.1 TurtleBot3	41
4.2 Bản đồ mô phỏng làn đường được sử dụng với Turtlebot3 và các biển báo	42
4.3 Công cụ tạo bản đồ làn đường	43
4.4 Bản đồ làn đường được tạo	43
4.5 Góc nhìn từ camera của robot	44
4.6 Binary thresholding ở mức 120	44
4.7 Binary thresholding ở mức 190	45
4.8 Model phát hiện biển báo	45
4.9 Phát hiện biển báo	46
4.10 Kiểm tra sau khi train trên bộ dữ liệu GTSDB	46
4.11 Một số thông số đánh giá model	47
4.12 Model phân loại biển báo	48
4.13 Bổ sung các biển báo trong mô phỏng vào bộ dữ liệu	49
4.14 Kiểm tra sau khi train trên bộ dữ liệu GTSRB	49
4.15 Một số thông số đánh giá model	50
4.16 Biển báo được nhận diện và diện tích bounding box	50
4.17 Làn đường trong bird's eye view	51
4.18 Hình ảnh làn đường được phát hiện	51
4.19 Vị trí tâm làn đường	52
4.20 Diện tích biển báo	52
4.21 Phân loại biển báo	52
4.22 Bộ điều khiển PID để tính vận tốc góc	53
4.23 Vận tốc góc sau khi tính toán	53
 5.1 Làn đường nhận diện được	55
5.2 Biểu diễn làn đường nhận diện được	56
5.3 Vị trí tâm làn đường khi robot lệch trái	56
5.4 Vị trí tâm làn đường khi robot lệch phải	57
5.5 Vị trí tâm làn đường khi robot đang ở vị trí lý tưởng	57
5.6 Nhận diện biển báo đi thẳng	58

5.7	Nhận diện biển báo rẽ trái	58
5.8	Tốc độ xử lý của model phát hiện biển báo	59
5.9	Tốc độ khung hình (FPS) của model phát hiện biển báo	59
5.10	Diện tích biển báo ở khoảng cách xa	60
5.11	Diện tích biển báo ở khoảng cách gần	60
5.12	Vận tốc góc khi robot di chuyển	61
5.13	Vận tốc góc khi robot di chuyển	61
5.14	Robot phát hiện biển báo và bắt đầu quá trình rẽ trái	62
5.15	Robot đang rẽ trái	62
5.16	Robot sắp kết thúc quá trình rẽ và tiếp tục với module nhận diện làn đường	62
5.17	Robot phát hiện biển báo và bắt đầu quá trình rẽ phải	63
5.18	Robot đang rẽ phải	63

Chương 1

Giới thiệu đề tài

1.1 Lý do và động lực thực hiện đề tài

Xe tự hành không người lái là mục tiêu phát triển trong tương lai của ngành giao thông vận tải. Một chiếc xe để có thể tự lái cần phải hiểu rõ môi trường xung quanh và tìm được đường đi với sự trợ giúp tối thiểu của con người. Giữ an toàn cho người lái và hành khách là yêu cầu quan trọng trong việc phát triển xe tự hành. Bám theo làn đường và hỗ trợ rẽ là những hệ thống cơ bản, quan trọng và cần thiết nhất, chúng cho phép các phương tiện phát hiện làn đường và giúp giữ xe ở vị trí lý tưởng so với làn đường và hỗ trợ trong việc rẽ của xe.

Việc phát triển và sử dụng xe tự hành không người lái góp phần giải quyết nhiều vấn đề về giao thông, đảm bảo cho sự an toàn của con người, nâng cao chất lượng cuộc sống. Xe tự hành không người lái ngày càng trở thành mối quan tâm của nhiều đề tài nghiên cứu có tính ứng dụng cao, đã và đang được nhiều nhóm nghiên cứu ưu tiên thực hiện với mục tiêu thương mại hóa, biến xe tự hành không người lái trở thành một thiết bị thiết yếu trong đời sống con người. Những lý do kể trên là động lực để em thực hiện đề tài nghiên cứu này cho Luận văn tốt nghiệp.

1.2 Mục tiêu của đề tài

1.2.1 Về kiến thức

- Nắm vững kiến thức về ROS, sử dụng ROS để hiện thực chương trình điều khiển robot.
- Nắm vững các kiến thức lập trình Python, C++, ROS.
- Nắm vững các kỹ thuật xử lý hình ảnh với OpenCV.
- Phân tích, giải quyết yêu cầu bài toán nhận diện hình ảnh bằng OpenCV.
- Hiện thực giải thuật phát hiện làn đường với các kỹ thuật xử lý ảnh.
- Tiếp cận với các kiến thức về Trí tuệ nhân tạo, đặc biệt là Học sâu (Deep Learning).

-
- Phân tích, giải quyết yêu cầu bài toán phát hiện và nhận diện vật thể mà cụ thể là biển báo giao thông, từ đó xây dựng thành module hoàn chỉnh để hỗ trợ xe tự hành.
 - Khả năng tích hợp các module thành hệ thống hoàn chỉnh.
 - Xây dựng các kịch bản thử nghiệm hệ thống trong môi trường mô phỏng

1.2.2 Về kết quả thực tế

- Robot có thể nhận diện được sự xuất hiện của làn đường, biết được giới hạn của làn đường dựa trên hình ảnh từ camera.
- Robot có khả năng di chuyển bám theo làn đường được thiết kế phỏng theo làn đường trên thực tế.
- Robot có thể hiện thực được giải thuật phát hiện biển báo trong hình ảnh từ camera và nhận diện một số biển báo giao thông ở các ngã ba, ngã tư.
- Robot có khả năng di chuyển đúng luật giao thông theo yêu cầu của biển báo ở các ngã rẽ.
- Nắm được quy trình phát triển giải pháp trong ngành Khoa học và Kỹ thuật máy tính: phân tích - thiết kế - hiện thực - kiểm tra.

1.3 Phạm vi nghiên cứu

Do quy mô, tính chất của một đề tài luận văn tốt nghiệp cộng với giới hạn về mặt thời gian và nguồn lực cho việc thực hiện nghiên cứu, phạm vi nghiên cứu của đề tài đã được giới hạn lại để đảm bảo có thể thực hiện đúng tiến độ và đáp ứng yêu cầu về khả năng hiện thực. Đề tài sẽ được tập trung vào các dự án thành phần sau:

- **Nhận diện làn đường:** Thực hiện nhận diện chủ yếu trên các làn đường thẳng mô phỏng đường giao thông thực tế cộng với sự xuất hiện của các ngã rẽ (ngã ba, ngã tư).
- **Nhận diện và phát hiện biển báo giao thông:** Trong điều kiện thực tế của Luật Giao thông đường bộ, có đến hàng trăm loại biển báo giao thông khác nhau, tuy nhiên đề tài đã được giới hạn lại ở mức nhận diện 4 loại biển báo giao thông thông dụng:
 - **Dừng:** Biển báo được cắm ở đích đến, đánh dấu vị trí kết thúc của đoạn đường thực nghiệm. Khi gặp biển báo này, xe sẽ dừng lại hoàn toàn.
 - **Rẽ trái:** Biển báo được cắm ở ngã ba hoặc ngã tư, xe sẽ tự động rẽ trái theo yêu cầu của biển báo.
 - **Rẽ phải:** Tương tự như biển báo rẽ trái, khi gặp biển báo này ở ngã ba hoặc ngã tư, xe sẽ tự động rẽ phải.

-
- **Đi thẳng:** Khi gặp biển báo này, xe sẽ tự động bỏ qua các ngã rẽ và tiếp tục di chuyển theo đường thẳng.

Để đảm bảo khả năng quan sát và xử lý thông tin của xe, mỗi giao lộ chỉ được cắm một biển báo duy nhất.

1.4 Ý nghĩa thực tiễn

Đề tài này được thực hiện với mong muốn đóng góp thêm một giải pháp trong việc nghiên cứu và phát triển xe tự hành. Nếu các nghiên cứu về xe tự hành được đầu tư và quan tâm đúng mức, các sản phẩm hoàn toàn có thể được thương mại hóa, đưa vào sử dụng trong thực tế để phục vụ cho đời sống con người. Một số ứng dụng tiêu biểu của các nghiên cứu này có thể kể đến như:

- **Ô tô tự lái:** Ô tô tự lái có lẽ là sản phẩm nổi tiếng nhất của nghiên cứu robot tự hành. Các công ty như Tesla, Google và Uber đã đầu tư rất nhiều vào việc phát triển các phương tiện tự lái, sử dụng kết hợp các cảm biến, thuật toán học máy và hệ thống điều khiển tiên tiến để giúp ô tô tự lái trên các con đường một cách an toàn và hiệu quả.
- **Robot trong nhà kho:** Xe/robot tự hành cũng đã được phát triển để trợ giúp con người trong các công việc như lấy hàng hoá và quản lý hàng hoá trong các kho hàng/nhà xưởng công nghiệp. Xe/robot có thể di chuyển theo các lùn đường có sẵn với sự hỗ trợ của các biển báo. Các công ty như Amazon và Alibaba đã đầu tư rất nhiều vào công nghệ này vì nó có khả năng cải thiện đáng kể hiệu quả công việc và giảm chi phí.

1.5 Bộ cục của luận văn

- **Chương 1: Giới thiệu**
 - Giới thiệu tổng quan về đề tài.
 - Trình bày lý do thực hiện đề tài, yêu cầu, mục tiêu, đối tượng và phạm vi nghiên cứu.
 - Trình bày ý nghĩa thực tiễn của đề tài
- **Chương 2: Cơ sở lý thuyết** Thực hiện nghiên cứu các lý thuyết có liên quan, các vấn đề nổi bật và các giải pháp đã có đối với bài toán nhận diện lùn đường, bài toán phát hiện và nhận diện biển báo giao thông và bài toán điều khiển robot. Nghiên cứu các công cụ có thể được sử dụng để hiện thực hoá, mô phỏng và xây dựng hệ thống.
 - Hệ điều hành robot (ROS)
 - Machine Learning
 - Deep Learning

-
- Convolutional Neural Network (CNN)
 - OpenCV
 - Vấn đề phát hiện làn đường
 - Vấn đề nhận diện biển báo giao thông
 - **Chương 3: Đề xuất giải pháp và thiết kế hệ thống**
 - Mô tả giải pháp đề xuất
 - Kiến trúc hệ thống đề xuất
 - **Chương 4: Hiện thực hệ thống**
 - Hiện thực các thuật toán
 - Thiết lập thiết bị thử nghiệm
 - Mô phỏng hệ thống
 - **Chương 5: Kết quả thực nghiệm**
 - Kết quả của module nhận diện làn đường
 - Kết quả của module nhận diện và phát hiện biển báo
 - Kết quả của module điều khiển
 - Đánh giá kết quả thử nghiệm tích hợp hệ thống
 - **Chương 6: Tổng kết đề tài**
 - Kết quả đạt được và hạn chế
 - Thuận lợi và khó khăn khi thực hiện đề tài
 - Hướng phát triển đề tài

Chương 2

Cơ sở lý thuyết

2.1 Robot Operating System (ROS)



Hình 2.1: Robot Operating System

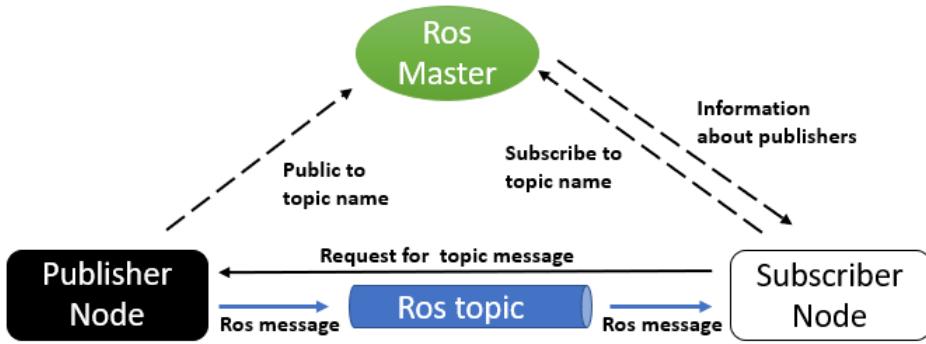
ROS (Robot Operating System) là một hệ thống (framework) mã nguồn mở được thiết kế để cung cấp cách giao tiếp tiêu chuẩn giữa các robot với nhau và giữa robot với các thiết bị khác trong môi trường làm việc. ROS cung cấp một bộ sưu tập các thư viện và các công cụ giúp các lập trình viên tạo ra các ứng dụng robot phức tạp bằng cách hỗ trợ kiến trúc kiểu module và phân tán. ROS không phụ thuộc vào nền tảng, nghĩa là nó có thể được sử dụng trên nhiều nền tảng phần cứng và phần mềm khác nhau.

ROS cũng bao gồm một thư viện lớn chứa các thuật toán, các driver và cảm biến được xây dựng sẵn có thể được sử dụng để nhanh chóng phát triển các ứng dụng robot mới. ROS được sử dụng rộng rãi trong ngành công nghiệp robot và đã được nhiều công ty và tổ chức nghiên cứu lớn áp dụng. Sự phổ biến của nó một phần là do tính linh hoạt và dễ sử dụng, cũng như tài liệu phong phú và khả năng hỗ trợ của cộng đồng.

Một số khái niệm trong ROS:

- **Package:** Trong ROS, package là đơn vị tổ chức cơ bản dành cho code và các tài nguyên liên quan trong một ứng dụng dành cho robot. Một package thường chứa các thư viện, file thực thi, file cấu hình và các tài nguyên khác cần thiết để chạy một hoặc nhiều node ROS.

Mỗi package trong ROS tuân theo một cấu trúc tiêu chuẩn, bao gồm file kê khai (package.xml) mô tả package và liệt kê các phần phụ thuộc (dependencies). Mã nguồn của



Hình 2.2: Sơ đồ hệ thống publish-subscribe message

package thường được lưu trong thư mục con có tên là src, còn các thư viện và file thực thi đã biên dịch được lưu trong thư mục con có tên là devel.

Các package trong ROS có thể dễ dàng được chia sẻ và tái sử dụng, đây là một trong những lợi ích chính của hệ thống ROS. Hệ thống package cung cấp một cách để tổ chức code và các tài nguyên khác theo module một cách thuận tiện cho các lập trình viên.

- **Node:** Trong ROS, một node là một process thực hiện một tác vụ cụ thể trong các ứng dụng dành cho robot. Các node thường được thiết kế theo module và có thể được kết hợp để tạo thành các hệ thống phức tạp hơn. Ví dụ: một node có thể chịu trách nhiệm đọc dữ liệu từ cảm biến, trong khi node khác có thể chịu trách nhiệm xử lý dữ liệu đó và gửi lệnh đến bộ điều khiển động cơ.

Các node giao tiếp với nhau bằng hệ thống publish-subscribe message, trong đó các node có thể publish message cho một topic cụ thể và các node khác có thể subscribe vào topic đó để nhận các message. Điều này cho phép các node giao tiếp với nhau theo cách phi tập trung và không đồng bộ, rất phù hợp cho các ứng dụng robot.

Các node trong ROS có thể được viết bằng nhiều ngôn ngữ lập trình, bao gồm C++, Python và Java. Chúng cũng có thể chạy trên nhiều nền tảng phần cứng, bao gồm máy tính cá nhân, hệ thống nhúng và máy chủ đám mây.

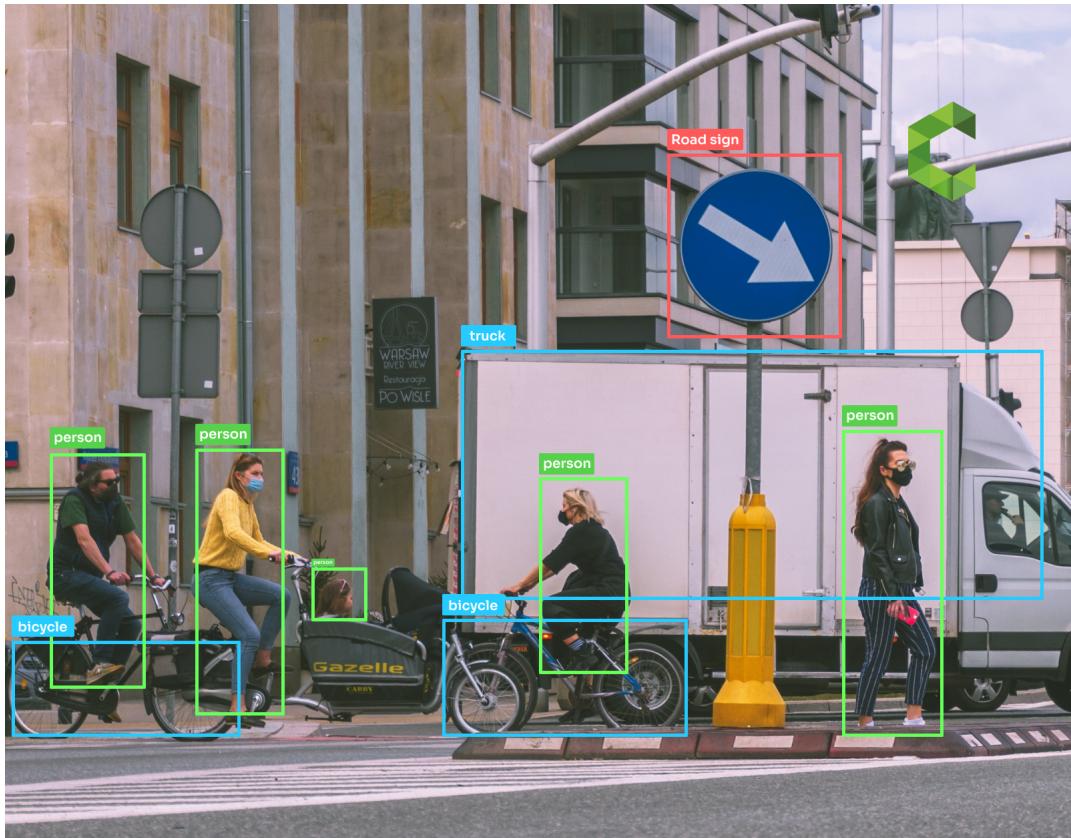
- **Message:** Trong ROS, một message là một cấu trúc dữ liệu đơn giản được sử dụng để truyền đạt thông tin giữa các node. Message được sử dụng để biểu thị dữ liệu như thông số đọc từ cảm biến, các lệnh và cập nhật trạng thái.

Message được định nghĩa trong package riêng và được lưu trong thư mục con có tên là 'msg'. Khi một message được định nghĩa, nó có thể được sử dụng bởi bất kỳ node nào trong hệ thống ROS cần truyền nhận thông tin thuộc loại đó.

- **Topic:** Trong ROS, một topic là một đường truyền được đặt tên mà các node có thể trao đổi message trên đó. Các topic được sử dụng để triển khai mô hình publish-subscribe message, trong đó các node tạo dữ liệu publish message lên một topic và các node cần sử dụng dữ liệu đó subscribe vào cùng một topic để nhận các message đó.

Mỗi topic được xác định bằng một tên duy nhất, tên này thường được chọn để phản ánh loại dữ liệu được trao đổi. Ví dụ: một topic mang thông số đọc từ cảm biến robot có thể được đặt tên là "/sensor_data", trong khi một topic mang các lệnh đến động cơ của robot có thể được đặt tên là "/motor_commands".

2.2 Thị giác máy tính (Computer Vision)



Hình 2.3: Bài toán nhận diện vật thể trong Thị giác máy tính

2.2.1 Giới thiệu về Thị giác máy tính

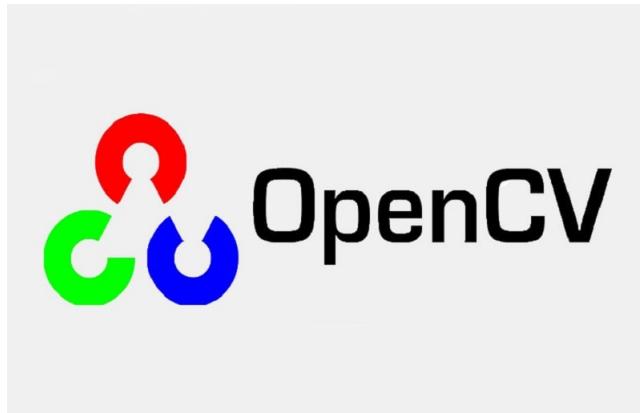
Thị giác máy tính là một lĩnh vực nghiên cứu tập trung vào việc cho phép máy tính diễn giải và hiểu thông tin hình ảnh từ thế giới xung quanh. Nó liên quan đến việc phát triển các thuật toán và kỹ thuật để phân tích và diễn giải các hình ảnh và video kỹ thuật số.

Các thuật toán thị giác máy tính có thể được sử dụng để giải quyết nhiều vấn đề, bao gồm nhận dạng đối tượng, phân đoạn hình ảnh, phân tích chuyển động và tái tạo 3D. Những kỹ thuật này được sử dụng rộng rãi trong các ứng dụng như xe tự hành, robot, chẩn đoán trong y tế bằng hình ảnh và hệ thống an ninh.

Một trong những thách thức chính trong thị giác máy tính là phát triển các thuật toán có thể trích xuất một cách chính xác và hiệu quả thông tin hữu ích từ dữ liệu trực quan. Điều này liên quan đến việc xử lý một lượng lớn dữ liệu và nhận dạng các mẫu hình cũng như cấu trúc trong hình ảnh và video.

Các kỹ thuật trong thị giác máy tính thường liên quan đến sự kết hợp giữa xử lý hình ảnh, học máy và đồ họa máy tính. Các kỹ thuật này được sử dụng để trích xuất các đặc điểm từ hình ảnh, phân loại đối tượng hoặc sự kiện và tạo ra thông tin hữu ích từ dữ liệu trực quan.

2.2.2 Giới thiệu về OpenCV



Hình 2.4: OpenCV

OpenCV (Open Source Computer Vision Library) là một thư viện phần mềm mã nguồn mở dành cho thị giác máy tính và học máy. Ban đầu nó được Intel phát triển vào cuối những năm 1990 và kể từ đó đã trở thành một công cụ tiêu chuẩn công nghiệp cho các ứng dụng thị giác máy tính.

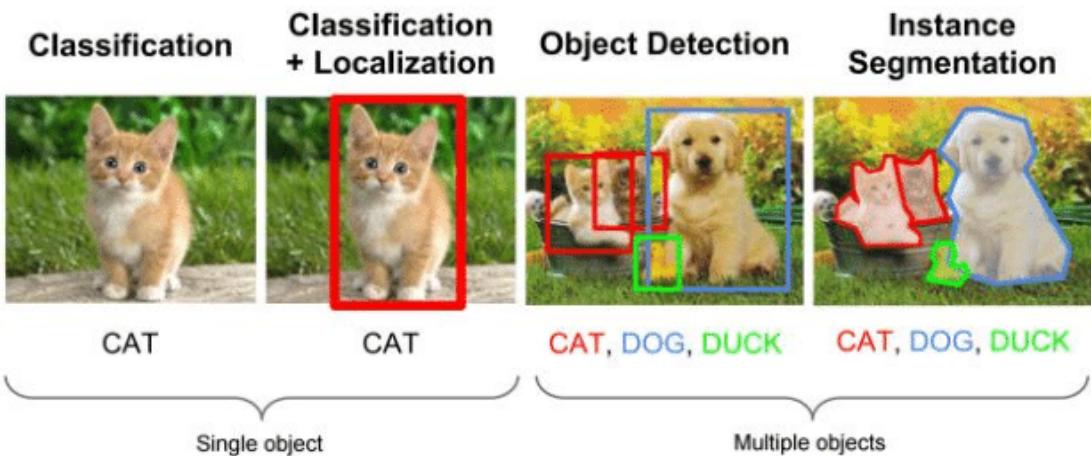
OpenCV cung cấp nhiều chức năng và thuật toán để xử lý hình ảnh và video, bao gồm các tính năng như lọc hình ảnh, phát hiện đặc điểm, nhận dạng đối tượng và hiệu chỉnh camera. Nó được thiết kế để tối ưu hóa cho hiệu suất cao và có thể được sử dụng với nhiều ngôn ngữ lập trình, bao gồm C++, Python và Java.

Một trong những điểm mạnh chính của OpenCV là tập hợp các thuật toán và chức năng có sẵn phong phú của nó. Các chức năng này có thể được tích hợp nhanh chóng và dễ dàng vào các ứng dụng thị giác máy tính, tiết kiệm thời gian và công sức cho các lập trình viên. Ngoài ra, OpenCV cung cấp một bộ công cụ phong phú để trực quan hóa, gỡ lỗi và thử nghiệm các ứng dụng thị giác máy tính.

2.2.3 So sánh bài toán phát hiện đối tượng và phân loại đối tượng (Object Detection vs Classification)

Phát hiện và phân loại đối tượng đều là những nhiệm vụ quan trọng trong thị giác máy tính, nhưng chúng có các mục tiêu và cách tiếp cận khác nhau.

Phát hiện đối tượng liên quan đến việc xác định và bản địa hóa các đối tượng quan tâm trong luồng hình ảnh hoặc video. Mục tiêu của phát hiện đối tượng là xác định tất cả các phiên bản của một loại đối tượng cụ thể trong luồng hình ảnh hoặc video và cung cấp thông tin về vị trí cũng như phạm vi của chúng. Tính năng phát hiện đối tượng thường liên quan đến việc sử dụng thuật toán máy học được đào tạo trên tập dữ liệu lớn gồm các hình ảnh được gắn nhãn và thuật



Hình 2.5: Các vấn đề trong Thị giác máy tính

toán đó có khả năng nhận dạng các đặc điểm và mẫu hình được liên kết với đối tượng cần quan tâm.

Mặt khác, bài toán phân loại liên quan đến việc xác định danh mục hoặc lớp mà một đối tượng thuộc về. Mục tiêu của phân loại là gán một đối tượng vào một danh mục hoặc lớp cụ thể dựa trên hình thức bên ngoài hoặc các tính năng khác của nó. Phân loại có thể là nhị phân (ví dụ: đó có phải là mèo hay không?) hoặc đa lớp (ví dụ: đó là mèo, chó hay chim?). Phân loại thường liên quan đến việc sử dụng thuật toán học máy được đào tạo trên tập dữ liệu lớn gồm các hình ảnh được gắn nhãn và có khả năng nhận dạng các tính năng và mẫu được liên kết với từng lớp.

Sự khác biệt chính giữa phát hiện và phân loại đối tượng là phát hiện đối tượng liên quan đến việc xác định vị trí và mức độ của các đối tượng trong luồng hình ảnh hoặc video, trong khi phân loại liên quan đến việc xác định danh mục hoặc lớp mà đối tượng thuộc về. Phát hiện đối tượng thường phức tạp hơn so với phân loại, vì nó liên quan đến việc xác định nhiều đối tượng trong luồng hình ảnh hoặc video và cung cấp thông tin về vị trí cũng như phạm vi của chúng. Tuy nhiên, cả hai nhiệm vụ đều quan trọng đối với nhiều ứng dụng trong thị giác máy tính, bao gồm robot, lái xe tự hành và giám sát.

2.3 Machine Learning

2.3.1 Giới thiệu về Machine Learning

Học máy là một lĩnh vực con của trí tuệ nhân tạo tập trung vào việc phát triển các thuật toán và mô hình có thể học hỏi từ dữ liệu và đưa ra dự đoán hoặc quyết định dựa trên dữ liệu đó. Nó liên quan đến việc sử dụng các kỹ thuật thống kê và tính toán để cho phép máy tính học các mẫu hình trong dữ liệu.

Mục tiêu chính của học máy là phát triển các thuật toán có thể học từ dữ liệu đã có và khai thác hóa việc học đó cho những dữ liệu mới và không cần phải thực hiện lại việc lập trình cho mỗi bộ dữ liệu mới. Điều này được thực hiện thông qua việc sử dụng dữ liệu huấn luyện, được

dùng để huấn luyện mô hình học máy nhận dạng các mẫu hình và đưa ra dự đoán dựa trên các mẫu hình đó.

Có ba loại học máy chính: học có giám sát, học không giám sát và học tăng cường.

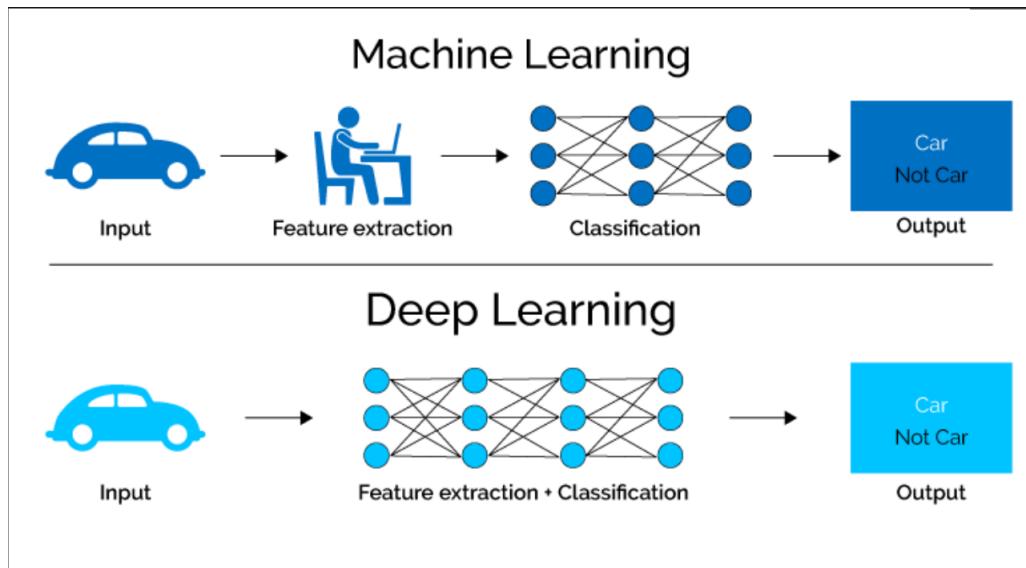
Trong học có giám sát, mô hình học máy được huấn luyện trên tập dữ liệu được gắn nhãn, trong đó mỗi điểm dữ liệu được liên kết với một đầu ra hoặc nhãn cụ thể. Mô hình học cách đưa ra dự đoán dựa trên các đặc điểm đầu vào và các nhãn tương ứng, sau đó có thể được sử dụng để đưa ra dự đoán về dữ liệu mới mà nó chưa từng nhìn thấy.

Trong học không giám sát, mô hình được huấn luyện trên tập dữ liệu không được gắn nhãn, trong đó mục tiêu là xác định các mẫu hình hoặc mối quan hệ trong dữ liệu mà không có bất kỳ nhãn hoặc đầu ra cụ thể nào.

Trong học tăng cường, mô hình học bằng cách tương tác với môi trường và nhận phản hồi dưới dạng phần thưởng hoặc hình phạt dựa trên hành động của nó. Mục tiêu của mô hình là tìm ra một giải pháp tối đa hóa phần thưởng của nó theo thời gian.

Nhìn chung, học máy là một công cụ mạnh mẽ để giải quyết nhiều vấn đề trong các lĩnh vực như nhận dạng hình ảnh và giọng nói, xử lý ngôn ngữ tự nhiên và phân tích dự đoán.

2.3.2 Giới thiệu về Deep Learning



Hình 2.6: Machine Learning và Deep Learning

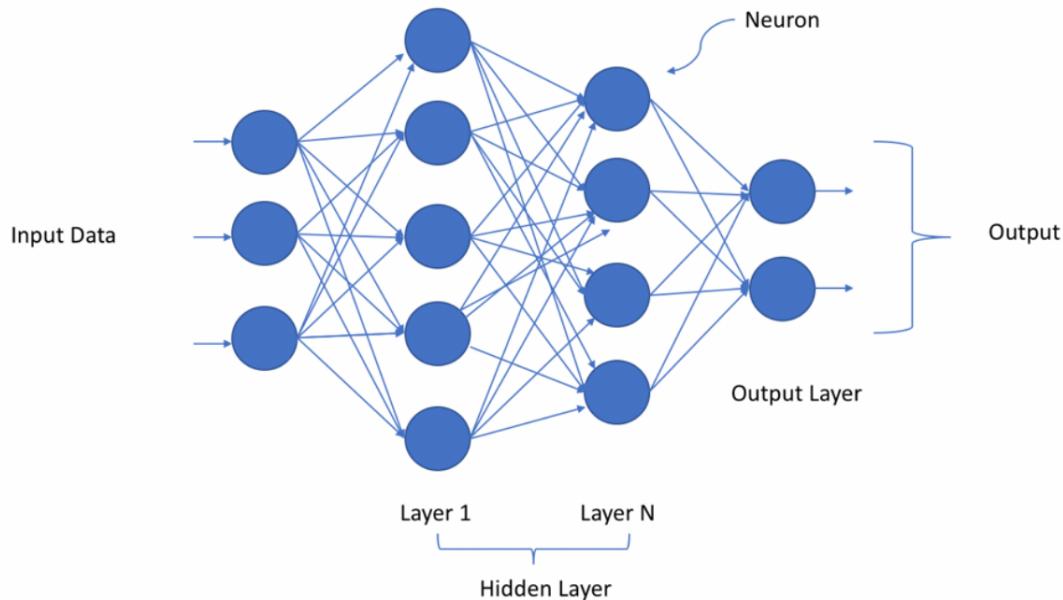
Học sâu là một lĩnh vực con của học máy liên quan đến việc sử dụng mạng thần kinh nhân tạo để lập mô hình và giải quyết các vấn đề phức tạp. Các mô hình học sâu thường bao gồm nhiều lớp tế bào thần kinh nhân tạo được huấn luyện trên các bộ dữ liệu lớn để học các mẫu hình phức tạp trong dữ liệu.

Một trong những điểm mạnh chính của học sâu là khả năng tự động học các biểu diễn phân cấp của dữ liệu. Mỗi lớp trong mạng neural sâu trích xuất các đặc điểm ngày càng phức tạp từ dữ liệu đầu vào, cho phép mô hình học các biểu diễn dữ liệu ở mức độ cao và trừu tượng hơn.

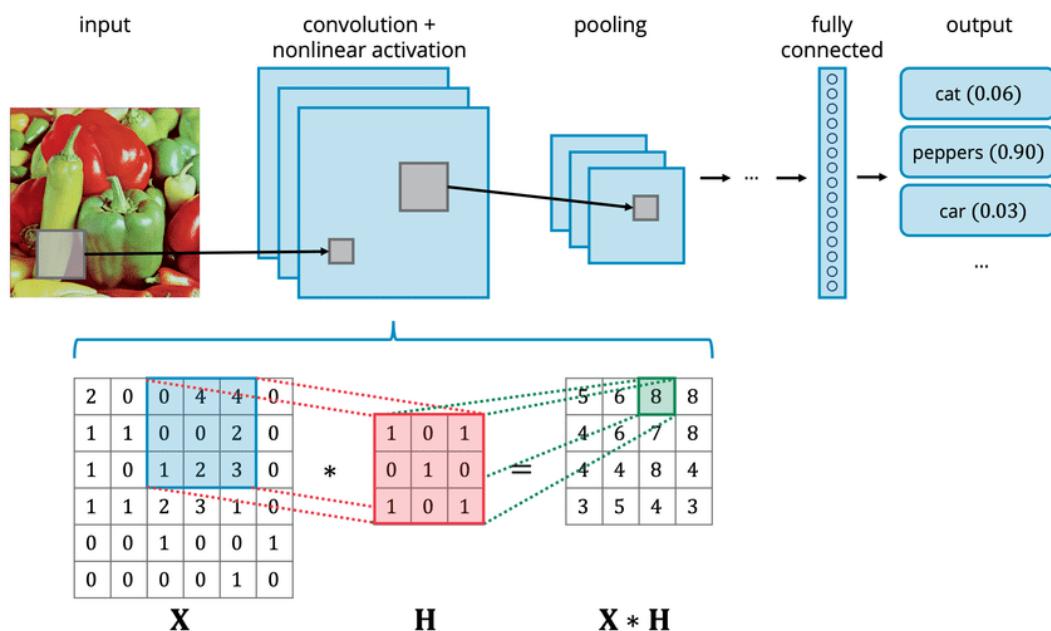
Học sâu đã đặc biệt thành công trong các bài toán như nhận dạng hình ảnh và giọng nói, xử

lý ngôn ngữ tự nhiên và chơi trò chơi. Một trong những thách thức chính trong học sâu là cần một lượng lớn dữ liệu huấn luyện được dán nhãn. Huấn luyện mạng thần kinh sâu có thể đòi hỏi tính toán chuyên sâu và yêu cầu phần cứng chuyên dụng như GPU hoặc TPU.

2.3.3 Giới thiệu về Convolutional Neural Network (CNN)



Hình 2.7: Deep Learning



Hình 2.8: Cấu trúc của một CNN

Mạng thần kinh tích chập (CNN) là một loại mạng thần kinh sâu thường được sử dụng cho các tác vụ xử lý hình ảnh và video, như phân loại hình ảnh, phát hiện đối tượng và phân đoạn.

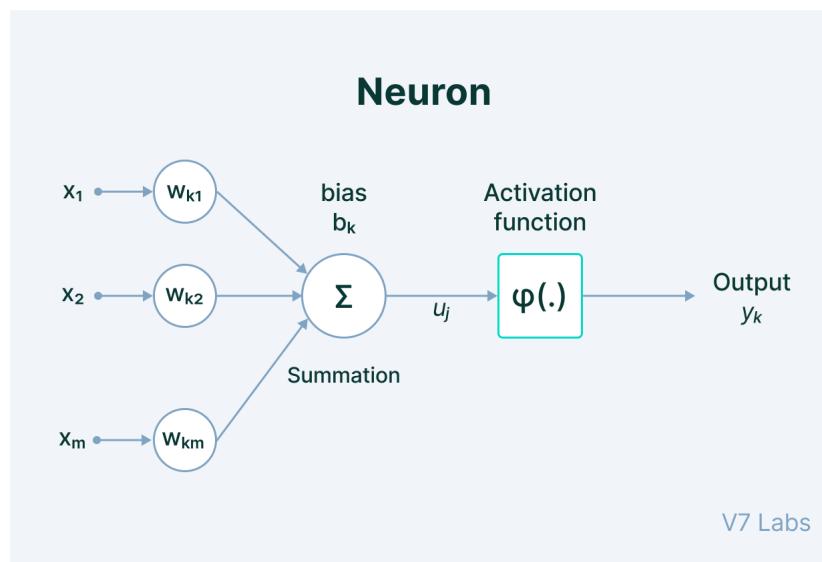
CNN được thiết kế để tận dụng cấu trúc không gian của hình ảnh bằng cách sử dụng các lớp tích chập để trích xuất các đặc điểm cục bộ và biểu diễn phân cấp từ dữ liệu đầu vào. Mỗi lớp tích chập áp dụng một tập hợp các bộ lọc cho hình ảnh đầu vào và tạo ra một tập hợp các bản đồ đặc điểm làm nổi bật các mẫu hình hoặc cấu trúc cụ thể trong hình ảnh.

Đầu ra của mỗi lớp tích chập thường được đưa vào một lớp tổng hợp, giúp giảm kích thước không gian của các bản đồ đặc điểm bằng cách lấy mẫu giảm (down-sampling) hoặc gộp các giá trị trong mỗi bản đồ đặc điểm. Điều này giúp giảm độ phức tạp tính toán của mô hình và cải thiện khả năng khai quát hóa cho dữ liệu mới.

Sau vài lớp tích chập và tổng hợp, các bản đồ đặc điểm được làm phẳng và đưa vào một hoặc nhiều lớp được kết nối đầy đủ để tạo ra đầu ra cuối cùng, chẳng hạn như đưa ra dự đoán phân lớp hoặc vẽ hộp giới hạn đối tượng (bounding box).

Các CNN được huấn luyện bằng cách sử dụng backpropagation và gradient descent, trong đó mô hình học cách giảm thiểu hàm mất mát (minimize loss function) đo lường sự khác biệt giữa đầu ra dự đoán và đầu ra thực tế cho một đầu vào nhất định.

2.3.4 Neuron



Hình 2.9: Neuron trong neural network

Trong mạng nơ-ron, nơ-ron là một đơn vị tính toán cơ bản nhận một hoặc nhiều đầu vào, thực hiện tính toán trên các đầu vào đó và tạo ra đầu ra. Một nơ-ron thường được mô phỏng theo các tế bào thần kinh sinh học được tìm thấy trong não người, nhận tín hiệu điện từ các tế bào thần kinh khác và sử dụng các tín hiệu này để tạo ra đầu ra điện của chính chúng.

Trong một mạng nơ-ron, một nơ-ron thường nhận được nhiều đầu vào, được kết hợp bằng cách sử dụng tổng trọng số. Các trọng số được liên kết với mỗi đầu vào xác định cường độ của tín hiệu đầu vào và có thể được điều chỉnh trong quá trình đào tạo để tối ưu hóa hiệu suất của mạng. Sau đó, nơ-ron áp dụng một hàm kích hoạt cho tổng trọng số để tạo ra đầu ra của nó.

Hàm kích hoạt là một hàm phi tuyến tính đưa tính phi tuyến tính vào mạng, cho phép mạng tìm hiểu các ánh xạ phức tạp giữa đầu vào và đầu ra. Việc lựa chọn chức năng kích hoạt phụ

thuộc vào ứng dụng cụ thể và đặc điểm của dữ liệu được phân tích. Các hàm kích hoạt phổ biến bao gồm hàm sigmoid, hàm tiếp tuyến hyperbol và hàm đơn vị tuyến tính được chỉnh lưu (ReLU).

Một mạng nơ-ron thường bao gồm nhiều lớp nơ-ron, được tổ chức thành một lớp đầu vào, một hoặc nhiều lớp ẩn và một lớp đầu ra. Lớp đầu vào nhận dữ liệu đầu vào, được truyền qua các lớp ẩn và được mạng biến đổi để tạo ra đầu ra. Trong quá trình đào tạo, các trọng số liên quan đến mỗi nơ-ron được điều chỉnh để giảm thiểu sự khác biệt giữa đầu ra dự đoán và đầu ra thực.

Nơ-ron là đơn vị xây dựng cơ bản của mạng nơ-ron, cho phép chúng học các ánh xạ phức tạp giữa đầu vào và đầu ra thông qua quá trình điều chỉnh trọng số và ứng dụng chức năng kích hoạt.

2.4 Gazebo



Hình 2.10: Gazebo

Gazebo là một môi trường mô phỏng 3D được thiết kế cho các ứng dụng robot. Đây là gói phần mềm mã nguồn mở cho phép người dùng tạo và mô phỏng các hệ thống robot phức tạp trong môi trường ảo.

Gazebo cung cấp nhiều tính năng và công cụ để lập mô hình và mô phỏng robot, bao gồm hỗ trợ mô phỏng dựa trên vật lý, mô phỏng cảm biến và trực quan hóa. Người dùng có thể tạo và thao tác với robot ảo bằng giao diện đồ họa đơn giản và có thể mô phỏng hành vi của những robot này trong nhiều điều kiện và môi trường khác nhau.

Một trong những điểm mạnh chính của Gazebo là khả năng mở rộng của nó. Người dùng có thể tạo các mô hình robot tùy chỉnh, thêm cảm biến và bộ truyền động mới, đồng thời phát triển các kịch bản mô phỏng của riêng mình bằng nhiều ngôn ngữ lập trình, bao gồm C++, Python và MATLAB.

Gazebo đã được áp dụng rộng rãi trong cộng đồng nghiên cứu robot và được sử dụng trong nhiều ứng dụng, bao gồm nghiên cứu, giáo dục và công nghiệp. Nó cho phép người dùng thử nghiệm và đánh giá hệ thống robot của họ trong một môi trường an toàn và được kiểm soát, đồng thời có thể giúp giảm thời gian và chi phí liên quan đến thử nghiệm và chế tạo nguyên mẫu vật lý.

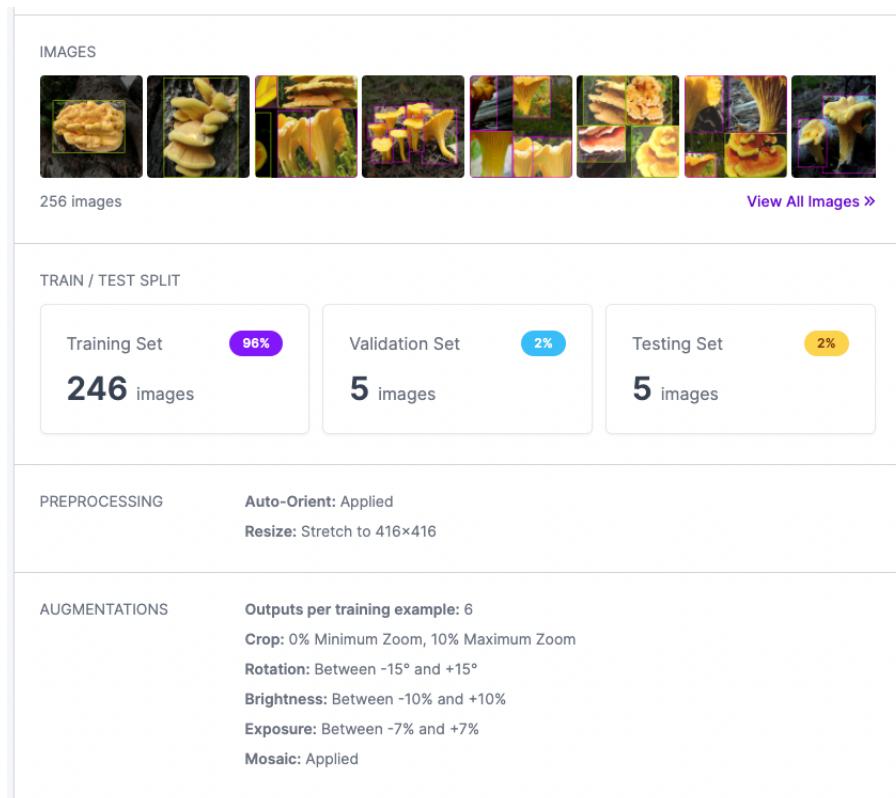
2.5 Roboflow

Roboflow là một nền tảng thị giác máy tính cung cấp một bộ công cụ giúp các nhà phát triển và nhà khoa học dữ liệu xây dựng và triển khai các mô hình thị giác máy tính. Roboflow cung cấp nhiều tính năng, bao gồm chú thích dữ liệu, tiền xử lý, đào tạo mô hình và triển khai, tất cả trong một nền tảng duy nhất.

Một trong những tính năng chính của Roboflow là công cụ chú thích dữ liệu, cho phép người dùng gắn nhãn hình ảnh bằng các hộp giới hạn, mặt nạ phân đoạn, điểm chính và các chú thích khác cần thiết cho các tác vụ thị giác máy tính. Nền tảng này cũng bao gồm một quy trình tiền xử lý dữ liệu có thể giúp thực hiện các tác vụ như thay đổi kích thước, tăng cường và chuẩn hóa hình ảnh.

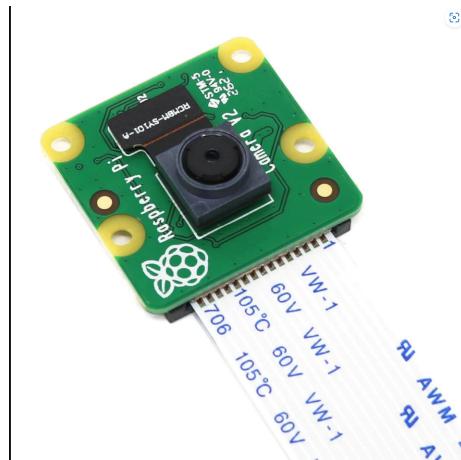
Roboflow hỗ trợ nhiều framework deep learning, bao gồm TensorFlow, PyTorch và Keras, đồng thời cung cấp các mô hình được huấn luyện trước có thể tinh chỉnh cho các trường hợp sử dụng cụ thể. Nền tảng này cũng bao gồm một mô-đun triển khai cho phép người dùng triển khai các mô hình của họ trên nhiều nền tảng khác nhau, bao gồm thiết bị di động, ứng dụng web và thiết bị biên.

Roboflow được tạo ra nhằm mục đích đơn giản hóa quá trình xây dựng và triển khai các mô hình thị giác máy tính, cho phép các nhà phát triển và nhà khoa học dữ liệu tập trung vào các khía cạnh sáng tạo hơn trong công việc của họ.



Hình 2.11: Roboflow

2.6 Camera



Hình 2.12: Raspberry Pi camera

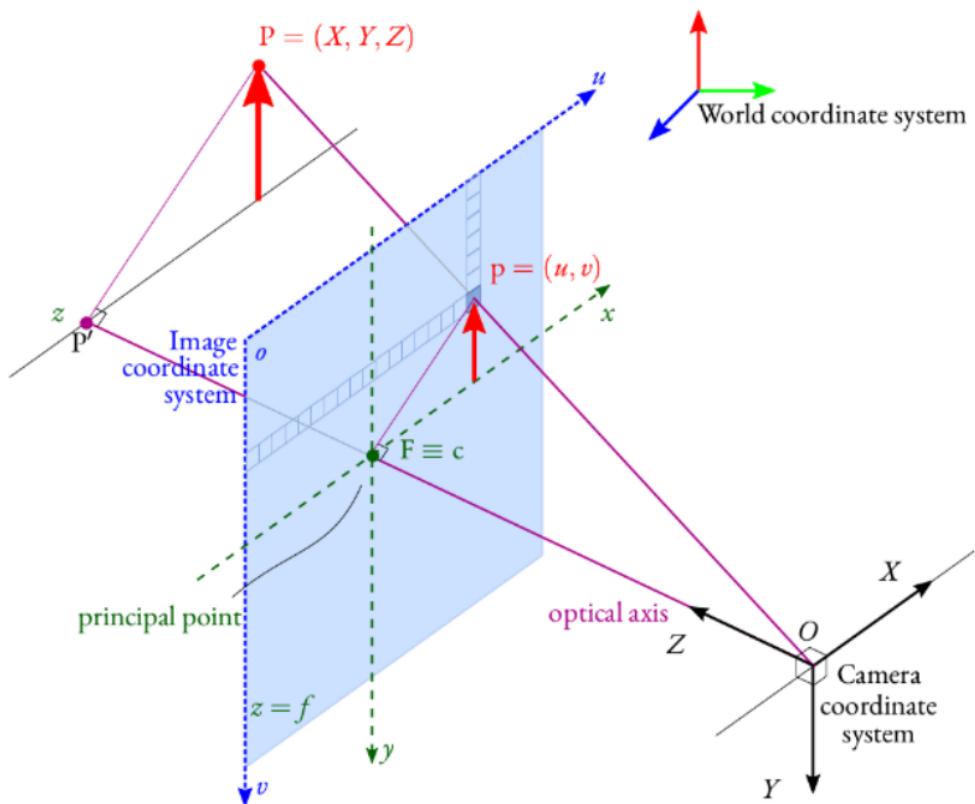
Camera thu thập thông tin dưới dạng pixel, tương tự như những gì mắt người nhìn thấy. Tuy nhiên, đôi mắt người là một bộ xử lý tự nhiên cực kỳ thông minh, có thể dễ dàng nhận ra làn đường, phương tiện và người đi bộ. Trong khi đó đối với máy móc, thông tin dưới dạng pixel chẳng là gì ngoài những con số. Lượng dữ liệu khổng lồ phải trải qua các quy trình phức tạp như trừu tượng hóa và tái cấu trúc sử dụng công nghệ học sâu.

Bởi vì góc nhìn của camera gần giống như của con người, các thuật toán học sâu cũng được sử dụng nhiều trong máy ảnh. Hiện nay, công nghệ camera cho xe tự hành cũng rất chuẩn xác với chi phí thấp, và phần chi phí lớn nhất là dành cho phần mềm xử lý. Do đó, khả năng hoạt động của phần mềm xử lý ảnh hưởng rất lớn đến công nghệ xe tự hành. Tuy nhiên, camera cũng có vấn đề tương tự như đôi mắt của con người, đó là nó không thể xử lý tốt được trong bóng tối hoặc với ánh sáng chói, nhưng lợi thế về độ phân giải và thông tin ẩn của nó có thể đủ để giúp ta giải quyết mọi vấn đề về mặt lý thuyết.

Trong mô hình camera, một điểm nhất định trong thế giới ba chiều và pixel tương ứng của nó có được thông qua việc chuyển đổi hệ tọa độ. Bốn hệ tọa độ tham gia vào quá trình này, đó là hệ tọa độ thế giới, hệ tọa độ camera, hệ tọa độ hình ảnh và hệ tọa độ pixel. Quá trình chuyển đổi của bốn hệ tọa độ sẽ được mô tả chi tiết dưới đây:

- world là hệ tọa độ thế giới, ta có thể tùy ý chỉ định trục xw và trục yw , là hệ tọa độ tại đó điểm P trong hình bên dưới là xác định vị trí.
- Camera là hệ tọa độ camera, gốc tọa độ nằm ở lỗ chốt O , điểm trục z trùng với trục quang và trục x và trục y song song với mặt phẳng chiểu, chính là hệ trục tọa độ XYZ trong hình bên.
- Ảnh là hệ tọa độ ảnh, gốc tọa độ là giao điểm của quang trục và mặt phẳng hình chiểu, trục x và trục y song song với hình chiểu mặt phẳng, là hệ trục tọa độ xyz trong hình bên.
- Pixel là hệ tọa độ điểm ảnh. Nhìn từ lỗ kim đến hình chiểu bề mặt, góc trên bên trái của

bề mặt hình chiếu là gốc và trục uv trùng với bề mặt hình chiếu. Hệ tọa độ và hình ảnh hệ tọa độ nằm trong cùng một mặt phẳng nhưng khác gốc tọa độ.



Hình 2.13: Bốn hệ tọa độ của camera

Chương 3

Đề xuất giải pháp và thiết kế hệ thống

3.1 Các phương pháp phát hiện làn đường phổ biến



Hình 3.1: Bài toán nhận diện làn đường

Một số thuật toán thường được sử dụng để nhận diện làn đường[8], bao gồm:

- Biến đổi Hough: Thuật toán Hough Transform là một kỹ thuật cổ điển được sử dụng để phát hiện làn đường. Nó chuyển đổi các điểm cạnh từ hình ảnh thành biểu diễn không gian tham số, trong đó các đường thẳng có thể được xác định là các đỉnh. Nó có thể phát hiện các làn đường trong một số điều kiện nhất định nhưng có thể gặp khó khăn với các làn đường cong hoặc không liên tục.
- Canny Edge detect: Thuật toán Canny Edge detect thường được sử dụng như một bước tiền xử lý để phát hiện làn đường. Nó phát hiện các cạnh trong ảnh bằng cách xác định các vùng có độ lớn gradient. Bằng cách trích xuất các cạnh, các thuật toán tiếp theo có thể tập trung vào việc trích xuất làn đường.

- Biến đổi Hough xác suất: Biến đổi Hough xác suất là một phần mở rộng của Biến đổi Hough khắc phục một số hạn chế của nó. Nó lấy mẫu ngẫu nhiên các điểm từ hình ảnh cạnh và xác định các đường bằng cách bỏ phiếu cho sự hiện diện của các đường trong không gian tham số. Nó hiệu quả hơn và có thể phát hiện vạch chính xác hơn, phù hợp cho việc phát hiện làn đường.
- RANSAC (Đồng thuận mẫu ngẫu nhiên): RANSAC là một thuật toán lặp được sử dụng để điều chỉnh các mô hình phù hợp với dữ liệu có các ngoại lệ. Trong phát hiện làn đường, RANSAC thường được sử dụng để đặt các làn đường hoặc đường cong vào các điểm cạnh được phát hiện, lọc ra các ngoại lệ do nhiễu hoặc các vật thể khác trong quang cảnh gây ra.
- Kỹ thuật Cửa sổ trượt (Sliding window): Kỹ thuật Cửa sổ trượt thường được sử dụng để phát hiện làn đường trong các tình huống mà trong đó các làn đường được xác định rõ ràng và liên tục. Thuật toán này bao gồm việc chia hình ảnh thành các phần theo chiều dọc và tìm kiếm các pixel thuộc về làn đường trong mỗi phần. Từ đó có thể xác định được các làn đường bằng cách trượt cửa sổ theo chiều dọc theo hình ảnh.
- Phương pháp tiếp cận dựa trên Deep Learning: Với những tiến bộ trong deep learning, nhiều thuật toán phát hiện làn đường hiện sử dụng mạng neural. Convolutional Neural Networks (CNN) thường được sử dụng để tìm hiểu các đặc điểm của làn đường trực tiếp từ dữ liệu hình ảnh. Một số kiến trúc phổ biến bao gồm ENet, LaneNet và SCNN (CNN không gian). Các phương pháp tiếp cận dựa trên học sâu này có thể xử lý các tình huống làn đường phức tạp, bao gồm làn đường cong và điều kiện ánh sáng phức tạp.

3.2 Đánh giá thuật toán nhận diện làn đường dựa trên Sliding Window

4.

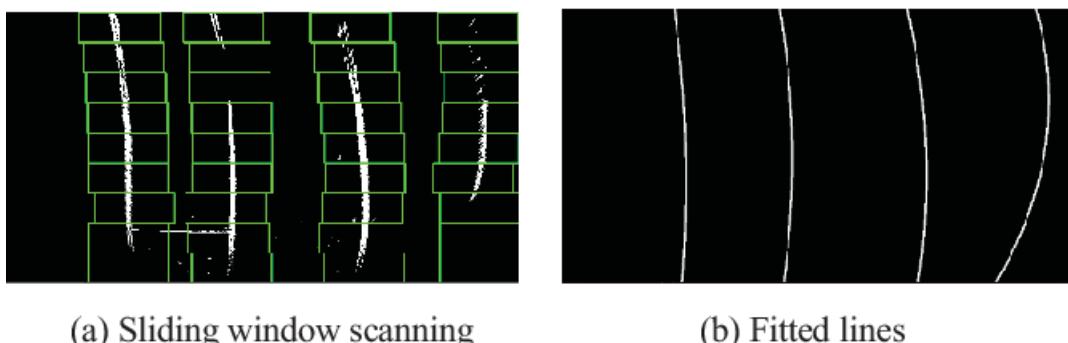


Fig. 4. Sliding window scanning and lane fitting

Hình 3.2: Kỹ thuật Sliding window

3.2.1 Ưu điểm:

- Tính đơn giản: Kỹ thuật Cửa sổ trượt tương đối đơn giản để thực hiện và hiểu. Nó liên quan đến việc tìm kiếm một cách có hệ thống các pixel làn đường bằng cách trượt một cửa sổ theo chiều dọc trên hình ảnh.
- Tính linh hoạt: Kỹ thuật Cửa sổ trượt có thể được áp dụng cho các tình huống phát hiện làn đường khác nhau, bao gồm làn đường thẳng, làn đường cong và chiều rộng làn đường thay đổi. Nó có thể thích ứng với các kiểu làn đường khác nhau bằng cách điều chỉnh kích thước và vị trí của cửa sổ trượt.
- Khả năng hiện thực: Bằng cách sử dụng kỹ thuật Cửa sổ trượt, có thể thu được vị trí gần đúng của các làn đường. Thông tin này có giá trị cho các bước xử lý tiếp theo, chẳng hạn như điều chỉnh đường hoặc đường cong cho các pixel làn đường được phát hiện.
- Tính mạnh mẽ: Kỹ thuật Cửa sổ trượt có thể xử lý các tình huống khó khăn, chẳng hạn như bị che khuất, bóng và nhiễu trong hình ảnh. Bằng cách bản địa hóa tìm kiếm trong mỗi cửa sổ, nó làm giảm ảnh hưởng của các đặc điểm không liên quan và tập trung vào khu vực làn đường.

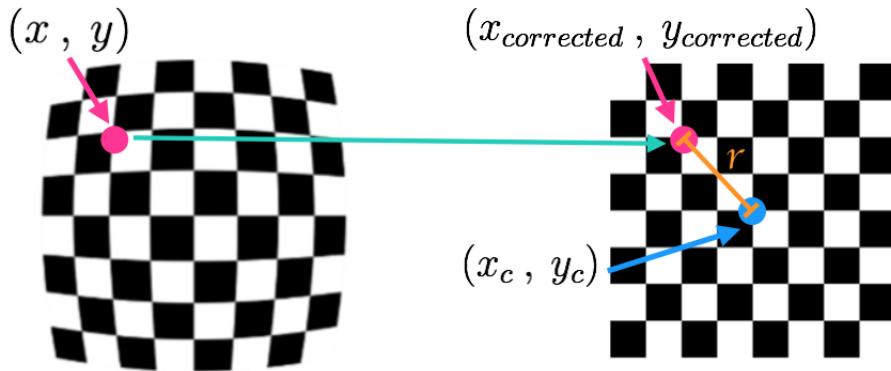
3.2.2 Nhược điểm:

- Chi phí tính toán: Kỹ thuật Cửa sổ trượt có thể tốn kém về mặt tính toán, đặc biệt khi áp dụng cho hình ảnh có độ phân giải cao hoặc trong các ứng dụng thời gian thực. Nhu cầu trượt cửa sổ trên toàn bộ hình ảnh và thực hiện phân tích pixel cho từng cửa sổ có thể làm chậm tốc độ xử lý.
- Kích thước và vị trí cửa sổ cố định: Kỹ thuật Cửa sổ trượt thường dựa trên kích thước và vị trí cửa sổ cố định, có thể không tối ưu cho tất cả các kiểu làn đường. Nó có thể gặp khó khăn trong việc thích ứng với các làn đường có độ rộng khác nhau hoặc các làn đường cong đòi hỏi hình dạng cửa sổ linh hoạt hơn.
- Độ nhạy khởi tạo: Kỹ thuật Cửa sổ trượt phụ thuộc rất nhiều vào vị trí ban đầu của các cửa sổ. Nếu vị trí ban đầu không chính xác hoặc nếu cửa sổ có kích thước không phù hợp thì có thể dẫn đến việc phát hiện làn đường không chính xác.
- Thiếu tính chắc chắn đối với các tình huống phức tạp: Kỹ thuật Cửa sổ trượt có thể gặp phải thách thức trong các tình huống phức tạp, chẳng hạn như cảnh đường đông đúc với nhiều làn đường, nhập làn hoặc giao lộ. Nó có thể gặp khó khăn trong việc xử lý các tình huống trong đó làn đường không được xác định rõ ràng hoặc khi có vạch kẻ làn đường đột ngột.
- Hiệu quả hạn chế đối với làn đường cong: Kỹ thuật Cửa sổ trượt được thiết kế chủ yếu để phát hiện làn đường thẳng và có thể có những hạn chế trong việc phát hiện chính xác làn

đường cong. Hình dạng và vị trí cửa sổ cố định có thể không nắm bắt được độ cong của lằn đường, dẫn đến kết quả kém chính xác hơn.

3.3 Thuật toán phát hiện lằn đường để xuất

3.3.1 Camera Calibration



Hình 3.3: Camera Calibration

Một trong những mục tiêu của xử lý hình ảnh là chỉnh sửa biến dạng. Biến dạng hình ảnh xảy ra khi camera nhìn vào các đối tượng 3D trong thế giới thực qua ống kính và biến chúng thành hình ảnh 2D khi quá trình chuyển đổi này không hoàn hảo. Các biến dạng trong một hình ảnh có thể:

- Thay đổi kích thước xuất hiện của một đối tượng
- Thay đổi hình dạng rõ ràng của một đối tượng
- Làm cho hình thức của một đối tượng thay đổi tùy thuộc vào vị trí của nó trong trường nhìn
- Làm cho các đối tượng xuất hiện gần hơn hoặc xa hơn so với thực tế.

Camera sử dụng các thấu kính cong để tạo thành hình ảnh và các tia sáng thường bị bẻ cong hơi nhiều hoặc quá ít ở các cạnh của các thấu kính này. Điều này tạo ra hiệu ứng làm biến dạng các cạnh của hình ảnh, làm cho các đường hoặc đối tượng có vẻ cong nhiều hơn hoặc ít hơn so với thực tế. Điều này được gọi là biến dạng radial, và nó là loại biến dạng phổ biến nhất. Một loại biến dạng khác là biến dạng tangential. Điều này xảy ra khi thấu kính của camera không được căn chỉnh hoàn toàn song song với mặt phẳng hình ảnh, nơi có film hoặc cảm biến của camera. Điều này làm cho hình ảnh bị nghiêng để một số đối tượng có vẻ xa hơn hoặc gần hơn so với thực tế. Để khắc phục hai loại biến dạng này, ta sẽ sử dụng một kỹ thuật gọi là hiệu chỉnh camera trong đó các bức ảnh đen trắng của các hình ảnh bàn cờ đã biết được đặt ở nhiều vị trí khác nhau, sau đó được chụp bằng camera và ống kính được sử dụng để chụp ảnh hoặc quay video tiếp theo.

OpenCV có một tập hợp các function có thể sử dụng các hình ảnh bàn cờ này và tính toán ma trận hiệu chỉnh cần thiết bằng cách sử dụng các công thức sau để hiệu chỉnh biến dạng radial: và các công thức hiệu chỉnh biến dạng tangential sau đây:

$$x_{\text{corrected}} = x(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

$$y_{\text{corrected}} = y(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

Hình 3.4: Radial distortion

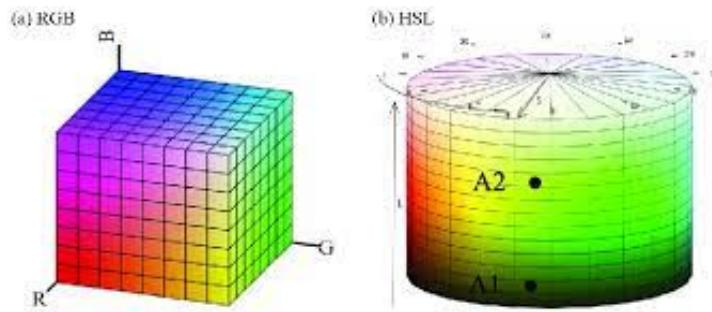
$$x_{\text{corrected}} = x + [2p_1xy + p_2(r^2 + 2x^2)]$$

$$y_{\text{corrected}} = y + [p_1(r^2 + 2y^2) + 2p_2xy]$$

Hình 3.5: Tangential distortion

3.3.2 Các phương pháp Thresholding

Đầu tiên cần chuyển đổi khung hình video từ không gian màu BGR (xanh dương, xanh lục, đỏ) sang HLS (màu sắc, độ bão hòa, độ sáng). Có rất nhiều cách để thể hiện màu sắc trong một hình ảnh, trong đó một cách để thể hiện màu là sử dụng không gian màu RGB (trong OpenCV là BGR thay vì RGB), trong đó mọi màu là hỗn hợp của ba màu đỏ, lục và lam (Red, Green, Blue). Không gian màu HLS được sử dụng hiệu quả hơn không gian màu BGR trong việc phát



Hình 3.6: Kênh màu RGB và HLS

hiện các vấn đề về hình ảnh do ánh sáng, chấn động như bóng, ánh sáng chói từ mặt trời, đèn pha, v.v. Module phát hiện làn đường cần phải loại bỏ tất cả những điều này để giúp phát hiện các vạch làn đường dễ dàng hơn. Vì lý do này, ta sử dụng không gian màu HLS, phân chia tất cả các màu thành các giá trị sắc độ, độ bão hòa và độ sáng (Hue, Lightness, Saturation).

Thực hiện phát hiện cạnh Sobel trên kênh L (độ sáng) của hình ảnh để phát hiện các điểm không liên tục sắc nét về cường độ pixel dọc theo trục x và y của khung hình video

Những thay đổi mạnh về cường độ từ một pixel sang pixel lân cận có nghĩa là có khả năng xuất hiện một cạnh. Ta cần phát hiện các cạnh mạnh nhất trong hình ảnh để có thể cô lập các cạnh của làn đường tiềm năng.

Sau đó, thực hiện binary thresholding trên kênh S của khung hình. Ta làm điều này giúp loại bỏ màu đường tối. Giá trị saturation càng cao có nghĩa là màu sắc càng tinh khiết. Các vạch kẻ

đường cần phải rõ ràng, có màu càng thuần khiết càng tốt, chẳng hạn như màu trắng đậm và màu vàng đậm. Cả hai màu trắng và vàng đậm đều có giá trị kênh saturation cao.

Binary Thresholding tạo ra một hình ảnh có đầy đủ các giá trị cường độ 0 (màu đen) và 255 (màu trắng). Các pixel có giá trị độ saturation cao (ví dụ: > 80 trên thang điểm từ 0 đến 255) sẽ được đặt thành màu trắng, trong khi mọi pixel khác sẽ được đặt thành màu đen.

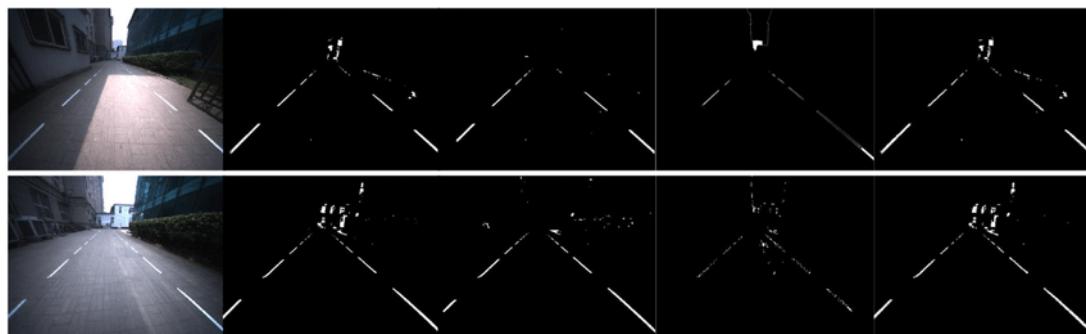
Tiếp theo cần thực hiện binary thresholding trên kênh R (màu đỏ) của khung video BGR gốc. Bước này giúp trích xuất các giá trị màu vàng và trắng, là màu đặc trưng của các làn đường.

Màu trắng tinh khiết là bgr(255, 255, 255). Màu vàng tinh khiết là bgr(0, 255, 255). Cả hai đều có giá trị kênh màu đỏ cao.

Để tạo hình ảnh nhị phân ở giai đoạn này, các pixel có giá trị kênh màu đỏ cao (ví dụ: > 120 trên thang điểm từ 0 đến 255) sẽ được đặt thành màu trắng. Tất cả các pixel khác sẽ được đặt thành màu đen.

Thực hiện thao tác bitwise AND để giảm nhiễu trong ảnh do bóng và sự thay đổi màu đường gây ra.

Các đường phân làn phải có màu thuần khiết và có giá trị kênh màu đỏ cao. Thao tác bitwise AND giúp giảm nhiễu và bôi đen bất kỳ pixel nào có vẻ không rõ ràng, thuần khiết, có màu đồng nhất.



Hình 3.7: Color Thresholding



Hình 3.8: Binary Thresholding

3.3.3 Sobel Thresholding

Sobel Thresholding là một kỹ thuật để trích xuất các cạnh từ hình ảnh bằng cách sử dụng toán tử Sobel, đây là một loại bộ lọc không gian tính toán gradient của cường độ hình ảnh tại

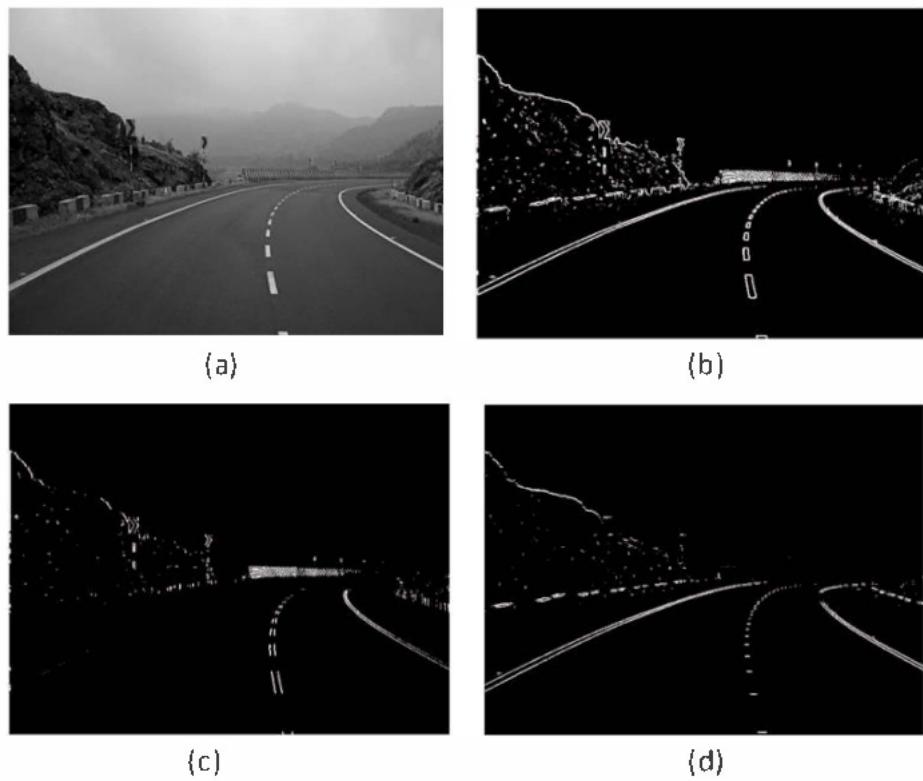


Fig. 10 Experiment Result (a) Original image (b) Edge detected image of

Hình 3.9: Sobel Thresholding với các ngưỡng khác nhau

mỗi pixel. Toán tử Sobel bao gồm hai kernel 3x3, một để phát hiện các cạnh ngang và một để phát hiện các cạnh dọc.

Các toán tử Sobel kết hợp ý tưởng về các bộ lọc phân biệt và làm mịn Gaussian và là trung tâm của thuật toán phát hiện cạnh Canny. Vì bộ lọc lấy dẫn xuất hình ảnh thứ nhất, thứ hai, thứ ba hoặc hỗn hợp của hình ảnh grayscale, kết quả ít nhiều có khả năng chống lại nhiễu trong hình ảnh và tạo ra các cạnh tương đối chính xác.

Để thực hiện Sobel thresholding, các bước sau đây thường được thực hiện:

- Chuyển đổi hình ảnh đầu vào thành ảnh grayscale.
- Áp dụng toán tử Sobel cho ảnh grayscale để tính độ lớn và hướng của gradient tại mỗi pixel.
- Tính giá trị ngưỡng để tách các pixel cạnh khỏi các pixel không thuộc cạnh. Giá trị ngưỡng này có thể được xác định bằng nhiều phương pháp khác nhau, chẳng hạn như phương pháp của Otsu hoặc tìm ngưỡng thủ công.
- Áp dụng ngưỡng cho hình ảnh cường độ gradient để tạo ra một hình ảnh nhị phân trong đó các pixel cạnh được đặt thành màu trắng và các pixel không thuộc cạnh được đặt thành màu đen.

Hình ảnh nhị phân kết quả có thể được sử dụng để xử lý thêm, chẳng hạn như phát hiện đường viền hoặc nhận dạng đối tượng.

Sobel Thresholding là một kỹ thuật phát hiện cạnh đơn giản và hiệu quả, thường được sử dụng trong các ứng dụng thị giác máy tính như robot, giám sát và lái xe tự hành. Tuy nhiên, nó có thể không phù hợp với tất cả các loại hình ảnh và các kỹ thuật phát hiện cạnh khác có thể phù hợp hơn tùy thuộc vào ứng dụng cụ thể và đặc điểm hình ảnh.

Các đạo hàm ảnh Sobel theo hướng X được cho bởi công thức sau đây với giả sử kernel 3x3.

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Hình 3.10: Kernel

Đạo hàm ảnh Sobel theo hướng Y được đưa ra bởi công thức sau đây giả sử kernel 3x3.

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Hình 3.11: Kernel

Kernel phải có kích thước 1, 3, 5 hoặc 7. Kernel có kích thước lớn hơn sẽ mang lại hình ảnh gradient mềm mại, mượt mà hơn. Đối với mục đích của bài toán phát hiện làn đường, ta cần các cạnh sắc nét, do đó, kernel 3 đã được sử dụng. Khi có kết quả, cần lấy giá trị tuyệt đối của đạo hàm hoặc gradient và scale chúng thành số nguyên không dấu 8 bit, vì về bản chất, ta sẽ sử dụng nó như một bitmap để cô lập các làn đường.

Ý tưởng lấy Sobel định hướng là cô gắng cô lập các cạnh dựa trên một hướng, đó là arctan của y-gradient chia cho x-gradient. Công thức là:

$$\tan^{-1}(sobel_y / sobel_x)$$

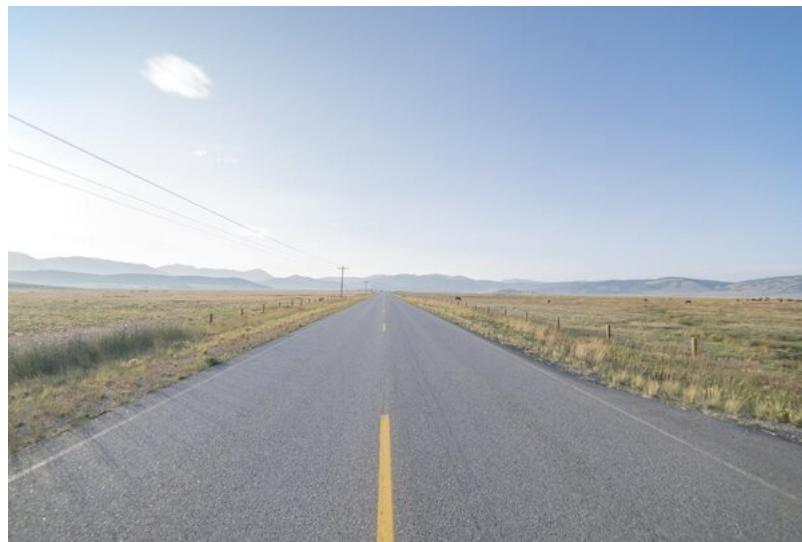
Hình 3.12: directional Sobel

Mỗi pixel của hình ảnh kết quả chứa một giá trị cho góc của gradient so với phương ngang tính theo đơn vị radian, trong phạm vi từ $-\pi/2$ đến $\pi/2$. Hướng 0 nghĩa là đường nằm ngang và hướng + hoặc $-\pi/2$ là đường dọc.

3.3.4 Perspective Transformation và Bird's Eye View

Từ góc nhìn bird's eye, các vạch ở hai bên làn đường trông giống như song song. Tuy nhiên, từ góc nhìn của camera gắn trên một chiếc ô tô bên dưới, các vạch phân làn có dạng giống hình thang. Ta không thể tính toán chính xác độ cong của làn đường bởi vì, từ góc nhìn của camera, chiều rộng làn đường dường như giảm khi càng rời xa camera.

Trên thực tế, trên đường chân trời, các đường phân làn dường như hội tụ tại một điểm (được biết đến trong thuật ngữ thị giác máy tính là điểm biến mất). Có thể thấy hiệu ứng này trong hình dưới đây:



Hình 3.13: Làn đường theo góc nhìn camera

Do đó, phối cảnh của máy ảnh không thể hiện chính xác những gì đang diễn ra trong thế giới thực. Cần sửa điều này để có thể tính toán độ cong của làn đường (sau này sẽ giúp ích cho chúng ta khi muốn điều chỉnh xe theo bám làn đường).

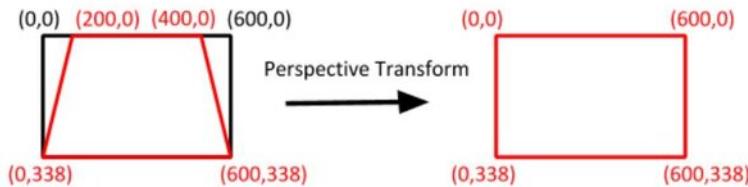
OpenCV có các phương thức giúp ta thực hiện phép biến đổi phối cảnh. Các phương pháp này biến phối cảnh của máy ảnh thành phối cảnh của chế độ xem mắt chim (tức là chế độ xem từ trên không).

Đối với bước đầu tiên của quá trình chuyển đổi góc nhìn, ta cần xác định vùng quan tâm (Region of interest - ROI). Bước này giúp loại bỏ những phần hình ảnh mà chúng ta không quan tâm. Ta chỉ quan tâm đến đoạn làn đường ngay phía trước xe. Để có được vùng quan tâm phù hợp, các bước sau đây cần được thực hiện:

- Xác định vùng quan tâm khái quát: Bắt đầu bằng cách xác định vùng quan tâm hình chữ nhật chung bao quanh khu vực nơi các làn đường dự kiến sẽ xuất hiện. Vùng quan tâm này phải bao gồm khu vực đường thường có vạch kẻ làn đường.
- Thích ứng với góc nhìn của camera: Tính đến góc nhìn của camera và điều chỉnh hình dạng vùng quan tâm để bù cho hiện tượng méo mó do góc nhìn. Thông thường, vùng quan tâm phải rộng hơn ở phần dưới cùng của hình ảnh và hẹp hơn về phía trên để giải quyết vấn đề các đường hội tụ do góc nhìn camera gây ra.
- Tinh chỉnh vùng quan tâm: Phân tích hình ảnh và tinh chỉnh vùng quan tâm dựa trên đặc điểm cụ thể của các làn đường trong cảnh. Xem xét vị trí và hướng diễn hình của các làn đường để tinh chỉnh hình dạng vùng quan tâm.
- Xem xét các yếu tố môi trường: Điều chỉnh vùng quan tâm dựa trên các yếu tố môi trường có thể ảnh hưởng đến tầm nhìn của làn đường. Ví dụ: nếu có chướng ngại vật chắn tầm nhìn trên đường, cần phải điều chỉnh vùng quan tâm để loại trừ những khu vực đó.
- Xác thực và lặp lại: Kiểm tra vùng quan tâm đã chọn trên nhiều hình ảnh hoặc khung hình video khác nhau trong môi trường thử nghiệm để đảm bảo rằng nó có thể bao gồm các

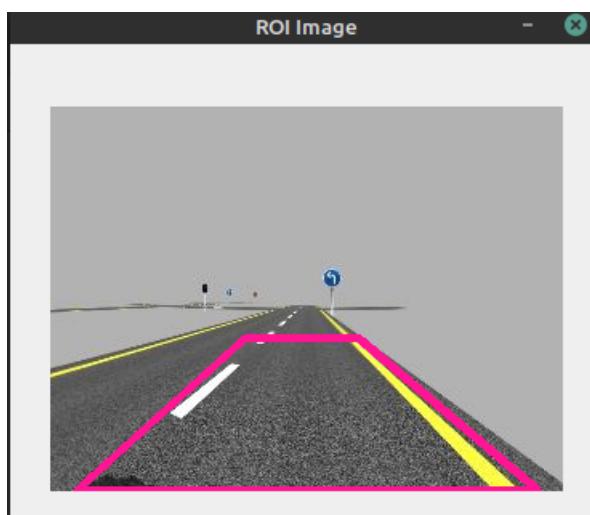
làn đường một cách nhất quán và loại trừ các khu vực không liên quan. Cần phải thiền độ chính xác và thích ứng với các tình huống khác nhau.

Bây giờ ta đã có vùng quan tâm, chúng ta sử dụng các phương thức getPerspectiveTransform và warpPerspective của OpenCV để chuyển đổi phối cảnh giống hình thang thành phối cảnh giống hình chữ nhật.



Hình 3.14: Perspective Transformation

Đây là một ví dụ về một hình ảnh sau quá trình này. Có thể thấy góc nhìn sau khi được điều chỉnh thành chế độ xem bird's eye view. Các đường trong vùng quan tâm hiện song song với các cạnh của hình ảnh, giúp tính toán độ cong của làn đường dễ dàng hơn.



Hình 3.15: Vùng quan tâm (ROI)

3.3.5 Phát hiện các làn đường

Bây giờ ta cần xác định các pixel trên hình ảnh tạo nên các đường làn đường. Nhìn vào hình ảnh được biến đổi, chúng ta có thể thấy rằng các điểm ảnh màu trắng đại diện cho các phần thuộc về các làn đường.

Ta bắt đầu phát hiện pixel thuộc làn đường bằng cách tạo histogram để xác định vị trí các khu vực của hình ảnh có mật độ điểm ảnh trắng cao.

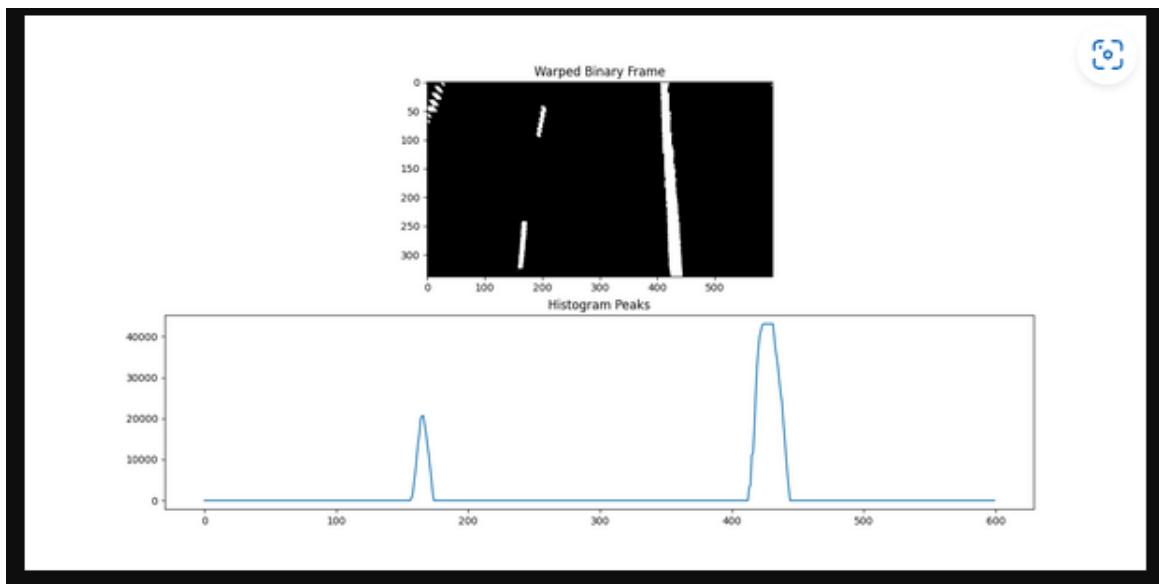
Histogram là một biểu diễn đồ họa hoặc một công cụ thống kê hiển thị phân phối của tập dữ liệu. Nó cung cấp một tóm tắt trực quan về tần suất hoặc sự xuất hiện của các giá trị hoặc phạm vi giá trị khác nhau trong tập dữ liệu. Histogram thường được sử dụng trong nhiều lĩnh vực khác nhau, bao gồm thống kê, phân tích dữ liệu, xử lý hình ảnh và xử lý tín hiệu.[1]



Hình 3.16: Bird-eye's view

Trong histogram, trục x biểu thị phạm vi giá trị hoặc các mức giá trị và trục y biểu thị tần suất hoặc số lần xuất hiện cho mỗi mức. Tập dữ liệu được chia thành các khoảng hoặc mức riêng biệt và chiều cao của mỗi thanh hoặc hình chữ nhật trong histogram tương ứng với tần suất hoặc số lượng giá trị nằm trong mức đó.

Lý tưởng nhất là khi chúng ta vẽ histogram, sẽ có hai đỉnh xuất hiện. Sẽ có một đỉnh bên trái và một đỉnh bên phải, tương ứng với vạch làn bên trái và vạch làn bên phải.

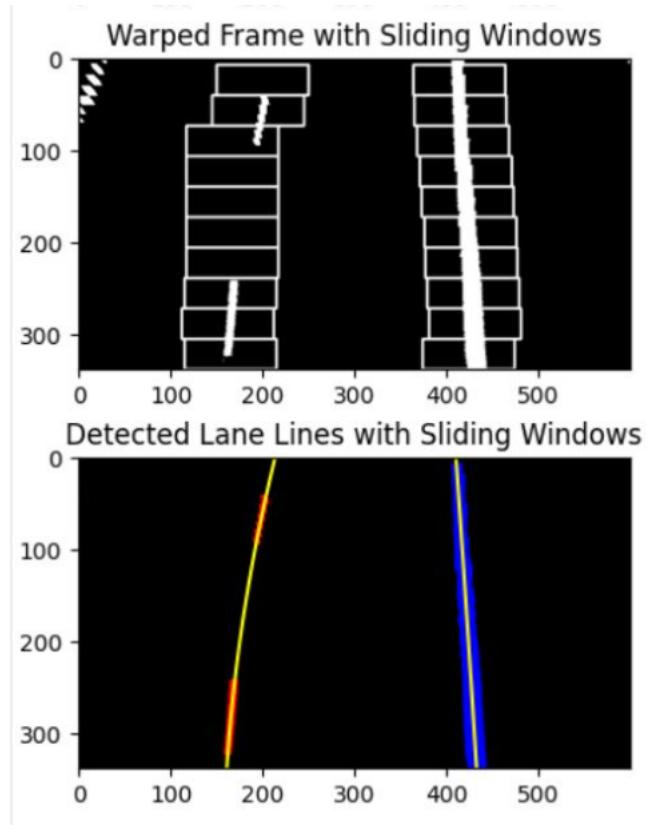


Hình 3.17: Histogram

3.3.6 Đặt sliding window để phát hiện pixel trắng

Trong phát hiện làn đường, kỹ thuật cửa sổ trượt thường được sử dụng để xác định và theo dõi các làn đường trong luồng hình ảnh hoặc video. Kỹ thuật này bao gồm việc chia hình ảnh thành các dải ngang hoặc vùng quan tâm (ROI) và áp dụng cửa sổ trượt có kích thước cố định cho mỗi vùng. Vị trí ban đầu của cửa sổ trượt thường được xác định dựa trên kiến thức hoặc giả

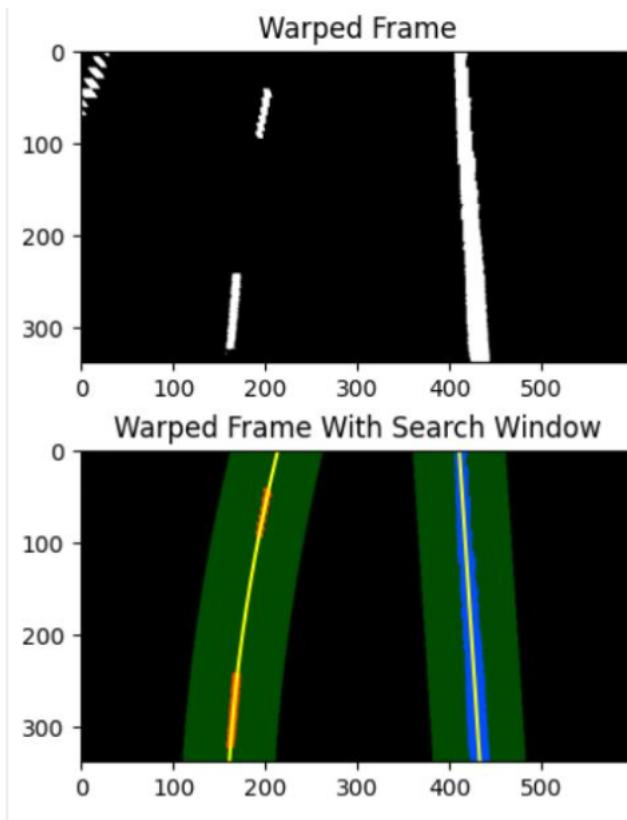
định trước đó về vị trí dự kiến của các vạch làn đường. Ở đây, ta sẽ sử dụng thông tin từ vị trí xuất hiện tập trung các pixel màu trắng thuộc về một trong hai làn đường bên trái và bên phải có thể thu thập được từ việc tính histogram để xác định vị trí ban đầu để đặt cửa sổ trượt. [3] Trong mỗi cửa sổ, sự hiện diện của các pixel hoặc cạnh của làn được đánh giá bằng cách áp dụng các kỹ thuật xử lý hình ảnh thích hợp như phát hiện cạnh hoặc phân ngưỡng màu. Mỗi khi tìm kiếm trong một cửa sổ trượt, ta sẽ thêm các pixel tiềm năng thuộc làn đường vào danh sách. Khi thực hiện thuật toán, ta sẽ tính vị trí trung bình của các pixel màu trắng thuộc làn đường có được từ bước phân ngưỡng ảnh. Sau đó, cửa sổ được định vị lại theo chiều dọc, theo các pixel làn đường được phát hiện để giữ cho các cửa sổ trượt luôn chứa các pixel thuộc làn đường và vị trí trung bình của các pixel này luôn nằm giữa cửa sổ, và quá trình được lặp lại cho đến khi đã tìm kiếm hết dọc theo chiều cao hình ảnh. Bằng cách thu thập các pixel làn đường được xác định trong cửa sổ trượt, mô hình toán học có thể được áp dụng cho các đường làn đường. Kỹ thuật cửa sổ trượt cho phép phát hiện các làn đường ngay cả trong các trường hợp làn đường có thể uốn cong hoặc thay đổi hình dáng. Nó cung cấp một cách tiếp cận cục bộ và có hệ thống để trích xuất thông tin làn đường từ hình ảnh, tạo thành một bước quan trọng trong thuật toán phát hiện làn đường.



Hình 3.18: Sliding window và biểu diễn làn đường

3.3.7 Đặt đa thức bậc hai vào các điểm thuộc làn đường

Việc đặt đa thức trong phát hiện làn đường bao gồm việc tìm một đường cong toán học thể hiện chính xác hình dạng của các làn đường. Quá trình này được sử dụng để mô hình hóa độ



Hình 3.19: Biểu diễn các làn đường

cong của làn đường và ước tính quỹ đạo của chúng. Pixel làn đường được trích xuất từ hình ảnh trong bước áp dụng kỹ thuật cửa sổ trượt, sau đó, một đa thức với bậc phù hợp sẽ được đặt vào vị trí các pixel này bằng phương pháp hồi quy. Các hệ số của đa thức được xác định để giảm thiểu sự khác biệt giữa đường cong được đặt và vị trí các pixel làn đường thực tế. Bằng cách điều chỉnh đa thức, các thuật toán phát hiện làn đường có thể thể hiện một cách hiệu quả tính chất cong của các làn đường, cho phép theo dõi làn đường chính xác và cung cấp thông tin có giá trị cho các tác vụ như cảnh báo chêch làn đường và lái xe tự hành.[13]

Đa thức bậc hai thường được sử dụng để trong việc phát hiện làn đường do khả năng tính gần đúng các đường cong của làn đường một cách hiệu quả. Các làn đường thường có độ cong liên tục, và đa thức bậc hai có thể nắm bắt được độ cong này một cách hợp lý mà không làm mô hình quá phức tạp.

Phương trình đa thức bậc hai có dạng $y = ax^2 + bx + c$, trong đó x và y đại diện cho tọa độ của làn đường và a , b và c là các hệ số của đa thức. Phương trình này biểu diễn một đường cong parabol, cho phép đa thức khớp theo một cách linh hoạt với hình dạng của các làn đường.

Việc sử dụng đa thức bậc cao hơn, chẳng hạn như bậc ba hoặc bốn, có thể gây ra sự phức tạp không cần thiết và làm tăng nguy cơ nhiều quá mức hoặc các giá trị ngoại lệ trong dữ liệu pixel của làn đường. Do các thuật toán phát hiện làn đường thường hoạt động với dữ liệu trong thế giới thực có thể chứa nhiều nên đa thức bậc hai tạo ra sự cân bằng giữa việc nắm bắt độ cong thiết yếu của các làn đường và duy trì tính đơn giản của mô hình.

Việc đặt đa thức bậc hai có hiệu quả tính toán so với đa thức bậc cao hơn, điều này có lợi cho các ứng dụng phát hiện làn đường theo thời gian thực nơi tài nguyên cho việc tính toán bị

hạn chế.

Tuy nhiên, cần lưu ý là việc lựa chọn bậc của đa thức có thể khác nhau tùy thuộc vào đặc điểm cụ thể của vạch làn đường và yêu cầu của hệ thống phát hiện làn đường. Trong một số trường hợp nhất định khi các làn đường có đường cong phức tạp hơn hoặc có các khúc cua gắt hơn, có thể cần phải có đa thức bậc cao hơn để thể hiện chính xác độ cong của các làn đường.

3.3.8 Xác định độ lệch vị trí của robot

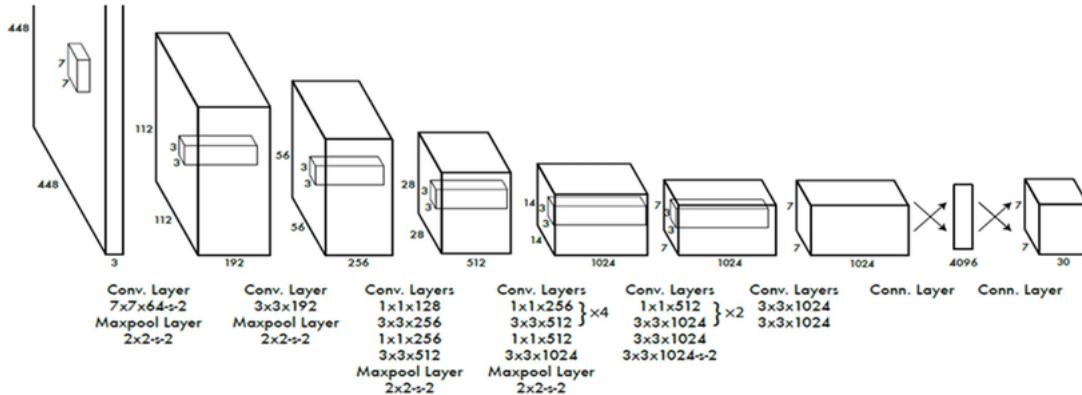


Hình 3.20: Xác định tâm làn đường

Để xác định được độ lệch của robot cần biết vị trí hiện tại của robot và tâm của làn đường. Do camera của robot được đặt giữa phía trước của robot nên robot luôn nằm giữa khung hình ảnh truyền từ camera. Từ vị trí các pixel thuộc làn đường tìm được từ kỹ thuật cửa sổ trượt, ta có thể tính được vị trí giữa tâm làn đường. Vị trí này cần được tính toán ở phía dưới theo chiều cao hình ảnh từ camera vì vị trí này gần với vị trí trong thực tế robot đang di chuyển. Thực hiện so sánh giữa vị trí tâm làn đường tính trên hình ảnh và vị trí giữa khung hình theo chiều ngang hình ảnh từ camera, ta sẽ xác định được độ lệch giữ vị trí hiện tại của robot so với tâm làn đường. Độ lệch giữa hai vị trí càng ít tức là robot đang di chuyển ở vị trí càng gần tâm làn đường, đây là vị trí lý tưởng nhất và robot cần duy trì vị trí này để đảm bảo không di chuyển lệch ra khỏi làn đường. Thông tin về độ lệch của vị trí robot sẽ được cung cấp cho module điều khiển để thực hiện tính toán vận tốc di chuyển của robot nhằm giữ robot luôn bám được làn đường, thực hiện đúng luật giao thông và điều chỉnh khi robot di chuyển có khả năng đi lệch khỏi làn đường.

3.4 Thuật toán nhận diện biển báo giao thông

3.4.1 YOLO (You Only Look Once)



Hình 3.21: Cấu trúc mạng YOLO

YOLO (You Only Look Once) là một thuật toán phát hiện đối tượng sử dụng mạng lưới thần kinh sâu để phát hiện các đối tượng trong một hình ảnh. Thuật toán YOLO được thiết kế để trở nên nhanh chóng và hiệu quả bằng cách xử lý toàn bộ hình ảnh chỉ trong một lần thay vì sử dụng phương pháp cửa sổ trượt như các thuật toán phát hiện đối tượng khác.[4]

Thuật toán YOLO bao gồm ba phần chính: xương sống của mạng thần kinh, một bộ anchor boxes và đầu phát hiện. Xương sống của mạng thần kinh thường là mạng thần kinh tích chập (CNN) trích xuất các đặc điểm từ hình ảnh đầu vào. Các anchor boxes là các hộp được xác định trước có kích thước và tỷ lệ khung hình khác nhau được sử dụng để dự đoán vị trí và kích thước của các đối tượng được phát hiện. Đầu phát hiện là một tập hợp các lớp được kết nối đầy đủ dự đoán xác suất của lớp và tọa độ hộp giới hạn cho mỗi hộp neo.

Trong quá trình huấn luyện, thuật toán YOLO được đào tạo bằng cách sử dụng các hình ảnh được gắn nhãn và các hộp giới hạn tương ứng của chúng. Thuật toán học cách dự đoán xác suất lớp và tọa độ hộp giới hạn cho mỗi hộp neo dựa trên các tính năng được trích xuất bởi đường trực mạng thần kinh.

Trong quá trình phát hiện, thuật toán YOLO xử lý toàn bộ hình ảnh bằng cách sử dụng mạng trực thần kinh, dự đoán xác suất lớp và tọa độ hộp giới hạn cho mỗi hộp neo bằng cách sử dụng đầu phát hiện, sau đó áp dụng triệt tiêu không tối đa để loại bỏ các phát hiện trùng lặp và chọn những phát hiện đáng tin cậy nhất.

Nhìn chung, thuật toán YOLO được biết đến với tốc độ và độ chính xác, khiến nó trở thành lựa chọn phổ biến cho các ứng dụng phát hiện đối tượng theo thời gian thực như ô tô tự lái, robot và giám sát.[9]

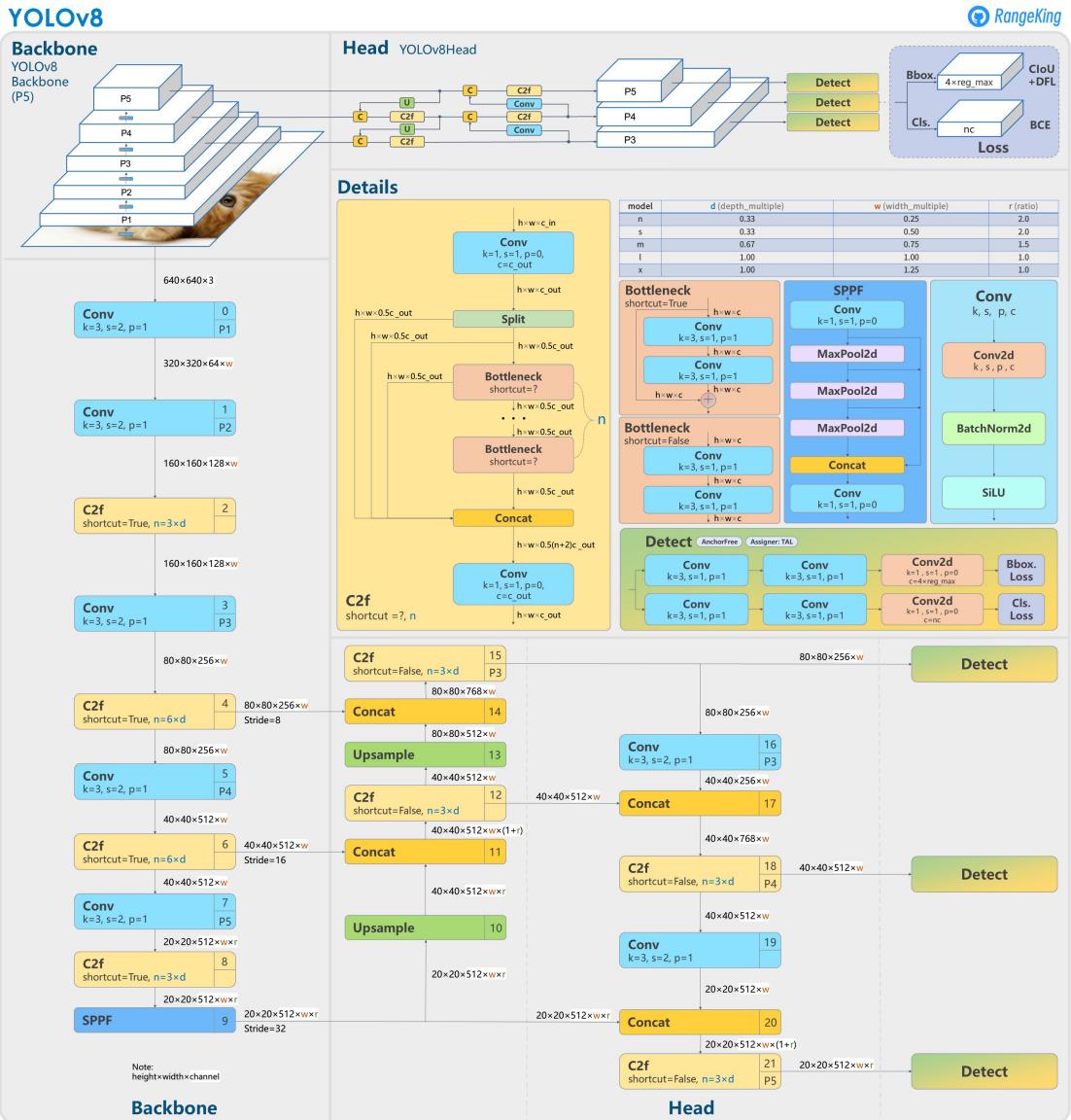
3.4.2 Quá trình phát triển YOLO

- Joseph Redmon và Ali Farhadi từ Đại học Washington đã phát triển YOLO, một mô hình phân đoạn hình ảnh và phát hiện đối tượng được sử dụng rộng rãi, vào năm 2015. YOLO nhanh chóng trở nên phổ biến nhờ tốc độ và độ chính xác cao. Vào năm 2016, YOLOv2 đã được phát hành, cải tiến trên mô hình ban đầu bằng cách kết hợp batch normalization, anchor boxes, và dimension cluster.
- YOLOv3, ra mắt vào năm 2018, đã nâng cao hơn nữa hiệu suất của mô hình bằng cách sử dụng backbone network hiệu quả hơn, nhiều anchors và spatial pyramid pooling. YOLOv4, được phát hành vào năm 2020, đã giới thiệu các tính năng mới như tăng cường dữ liệu Mosaic, anchor-free detection head mới và loss function mới.
- YOLOv5 đã được phát hành với các tính năng bổ sung như tối ưu hóa siêu tham số, theo dõi thử nghiệm tích hợp và xuất tự động sang các định dạng xuất phổ biến. Meituan đã cung cấp mã nguồn mở YOLOv6 vào năm 2022 và nó hiện đang được sử dụng trong nhiều robot giao hàng tự động của công ty. YOLOv7 đã thêm ước tính hình dáng trên bộ dữ liệu chính của COCO.
- Phiên bản mới nhất của YOLO là YOLOv8 của Ultralytics, đây là một mô hình tiên tiến, hiện đại (SOTA) được xây dựng dựa trên sự thành công của các phiên bản trước. YOLOv8 giới thiệu các tính năng và cải tiến mới để nâng cao hiệu suất, tính linh hoạt và hiệu quả, khiến nó trở thành một mô hình linh hoạt cho các tác vụ AI tầm nhìn khác nhau, bao gồm phát hiện, phân đoạn, ước tính tư thế, theo dõi và phân loại.

3.4.3 Kiến trúc và đặc điểm

Mô hình YOLO là một thuật toán phát hiện đối tượng phổ biến được biết đến với hiệu suất thời gian thực. Nó sử dụng một kiến trúc mạng nơ-ron duy nhất để dự đoán đồng thời các hộp giới hạn và xác suất của lớp cho các đối tượng trong ảnh đầu vào.[7] Tổng quan và các đặc điểm về kiến trúc mô hình YOLO:

- Phân chia đầu vào: Hình ảnh đầu vào được chia thành một lưới các ô. Mỗi ô có trách nhiệm dự đoán các khung giới hạn cho các đối tượng nằm trong ranh giới của nó.
- Trích xuất tính năng: Mô hình YOLO sử dụng mạng thần kinh tích chập (CNN) làm xương sống để trích xuất các tính năng từ hình ảnh đầu vào. Thông thường, các biến thể của kiến trúc Darknet, chẳng hạn như Darknet-19 hoặc Darknet-53, được sử dụng làm mạng cơ sở.
- Dự đoán ở nhiều tỷ lệ: Các bản đồ đặc trưng thu được từ mạng đường trực được xử lý bằng các lớp tích chập bổ sung để thu được các biểu diễn cấp cao hơn. Điều này cho phép mô hình đưa ra dự đoán ở nhiều tỷ lệ, thu thập các đối tượng có kích thước và độ phân giải khác nhau.



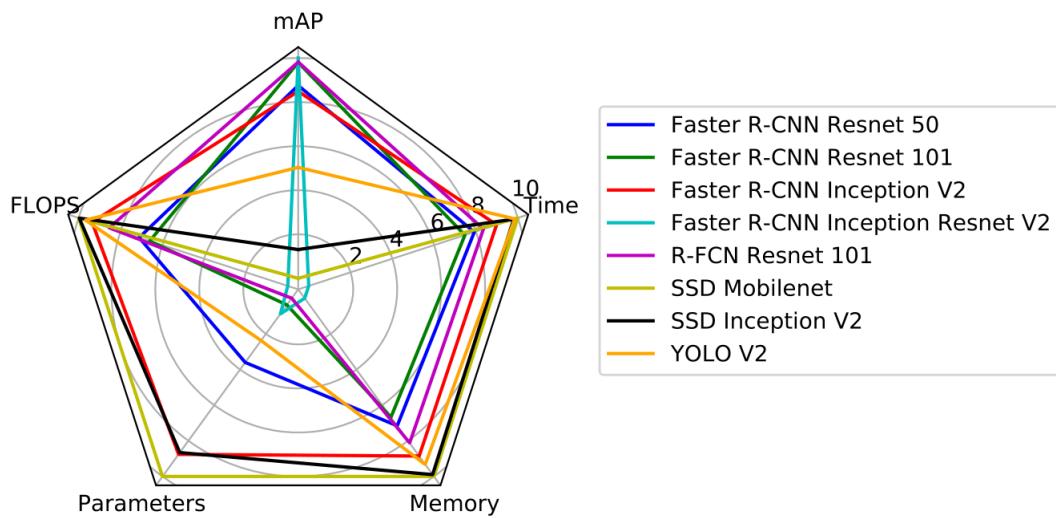
Hình 3.22: Kiến trúc cập nhật của YOLOv8

-
- Dự đoán hộp giới hạn: Mỗi ô lưới dự đoán một số hộp giới hạn cố định cùng với điểm tin cậy tương ứng của chúng. Đối với mỗi hộp giới hạn, mô hình xuất ra bốn giá trị: tọa độ góc trên bên trái của hộp (x, y) so với ô lưới, chiều rộng của hộp (w) và chiều cao (h).
 - Dự đoán lớp: Cùng với dự đoán hộp giới hạn, mô hình cũng dự đoán xác suất của từng lớp đối với các đối tượng có trong mỗi ô lưới. Số lượng xác suất của lớp được dự đoán phụ thuộc vào tập dữ liệu được sử dụng.
 - Loại bỏ không tối đa (Non-maximum Suppression): Sau khi có được các dự đoán về hộp giới hạn và xác suất của lớp, một bước xử lý hậu kỳ được gọi là loại bỏ không tối đa sẽ được áp dụng. NMS loại bỏ các dự đoán khung giới hạn dư thừa và chồng chéo bằng cách chỉ giữ lại những dự đoán chắc chắn nhất. Nó giúp giảm phát hiện trùng lặp và cải thiện đầu ra cuối cùng.

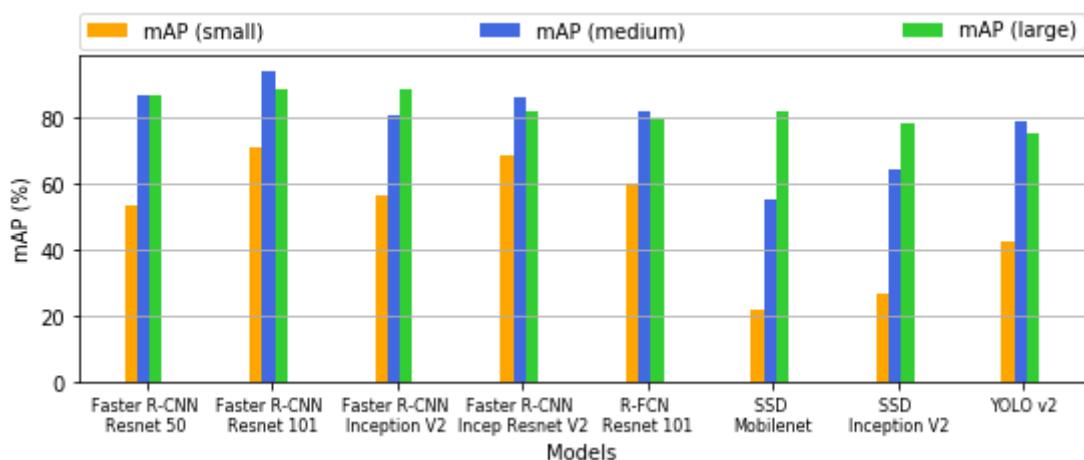
3.4.4 So sánh YOLO và một số mô hình CNN khác trong bài toán phát hiện vật thể

YOLO (You Only Look Once) có một số ưu điểm so với các thuật toán phát hiện đối tượng khác:[2]

- Tốc độ: YOLO được biết đến với tốc độ và hiệu quả, khiến nó trở nên lý tưởng cho các ứng dụng phát hiện đối tượng thời gian thực. Không giống như các thuật toán phát hiện đối tượng khác sử dụng phương pháp cửa sổ trượt, YOLO xử lý toàn bộ hình ảnh chỉ một lần, giúp giảm đáng kể thời gian tính toán.
- Tính đơn giản: YOLO có kiến trúc đơn giản bao gồm một mạng thần kinh duy nhất dự đoán các lớp đối tượng và hộp giới hạn. Sự đơn giản này giúp dễ dàng triển khai và tùy chỉnh cho các ứng dụng cụ thể.
- Độ chính xác: Mặc dù YOLO có thể không phải là thuật toán phát hiện đối tượng chính xác nhất, nhưng nó đã cho thấy đạt được hiệu suất cạnh tranh trên nhiều tiêu chuẩn khác nhau, đặc biệt là về tốc độ và hiệu quả.
- Tính linh hoạt: YOLO có khả năng tùy biến cao, cho phép người dùng đào tạo mô hình trên bộ dữ liệu của riêng họ và tinh chỉnh nó cho các ứng dụng cụ thể. Tính linh hoạt này làm cho YOLO trở thành một công cụ linh hoạt để phát hiện đối tượng trong nhiều lĩnh vực khác nhau, bao gồm rô-bốt, giám sát và lái xe tự động.
- Phát hiện và phân loại đối tượng trong một bước duy nhất: YOLO thực hiện cả phát hiện và phân loại đối tượng trong một bước duy nhất, giúp giảm độ phức tạp của thuật toán và làm cho nó hiệu quả hơn.



Hình 3.23: So sánh một số CNN



Hình 3.24: So sánh một số CNN với mAP là độ chính xác trung bình

3.5 Bộ dữ liệu dành cho huấn luyện model

3.5.1 GTSDB (German Traffic Sign Detection Benchmark)



Hình 3.25: Hình ảnh đường giao thông với các biển báo trong GTSDB

GTSDB là viết tắt của German Traffic Sign Detection Benchmark, đây là bộ dữ liệu điểm chuẩn được tiêu chuẩn hóa để thử nghiệm và đánh giá các thuật toán phát hiện biển báo giao thông. Bộ dữ liệu được tạo bởi Đại học Ulm và chứa các hình ảnh về cảnh giao thông từ đường cao tốc Autobahn của Đức.

Bộ dữ liệu GTSDB bao gồm hơn 39.000 hình ảnh được chú thích về cảnh giao thông, với tổng số 43 loại biển báo giao thông khác nhau. Các hình ảnh được chụp bằng camera có độ phân giải cao gắn trên ô tô đang lái dọc theo Autobahn và các biển báo giao thông được chú thích bởi các chuyên gia chú thích bằng cách sử dụng các hộp giới hạn.

Bộ dữ liệu GTSDB được sử dụng rộng rãi trong cộng đồng thị giác máy tính như một tiêu chuẩn chuẩn để thử nghiệm và đánh giá các thuật toán phát hiện biển báo giao thông. Bộ dữ liệu là một thách thức do sự thay đổi trong điều kiện ánh sáng, thời tiết và các yếu tố khác có thể ảnh hưởng đến hình thức của các biển báo giao thông.

Ngoài bộ dữ liệu điểm chuẩn tiêu chuẩn, GTSDB cũng bao gồm một bộ số liệu đánh giá để đo lường hiệu suất của các thuật toán phát hiện biển báo giao thông. Các số liệu này bao gồm độ chính xác trung bình trung bình (mAP), đo lường độ chính xác của thuật toán phát hiện và tỷ lệ phát hiện, đo lường phần trăm biển báo giao thông được phát hiện chính xác.

GTSDB là một nguồn tài nguyên quý giá dành cho các nhà nghiên cứu và nhà phát triển đang làm việc trên các thuật toán phát hiện biển báo giao thông và đã đóng một vai trò quan trọng trong việc thúc đẩy công nghệ tiên tiến nhất trong lĩnh vực này.



Hình 3.26: Hình ảnh biển báo trong GTSRB

3.5.2 GTSRB (German Traffic Sign Recognition Benchmark)

GTSRB (German Traffic Sign Recognition Benchmark) là một bộ dữ liệu chuẩn để nhận dạng biển báo giao thông. Bộ dữ liệu bao gồm hơn 50.000 hình ảnh biển báo giao thông, được chụp trong các điều kiện thời tiết và ánh sáng khác nhau, bao gồm 43 loại biển báo giao thông khác nhau.

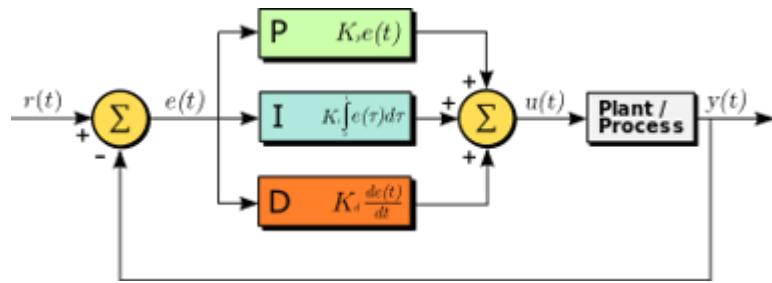
Bộ dữ liệu GTSRB được sử dụng rộng rãi làm chuẩn để đánh giá các thuật toán nhận dạng biển báo giao thông. Các nhà nghiên cứu sử dụng bộ dữ liệu để huấn luyện và kiểm tra các thuật toán của họ, đồng thời so sánh hiệu suất của chúng với các phương pháp tiên tiến khác.

Bộ dữ liệu GTSRB cũng bao gồm một tập hợp các phân tách xác thực và đào tạo được xác định trước, đảm bảo rằng tất cả các phương pháp được đánh giá trên cùng một dữ liệu và cho phép so sánh công bằng giữa các thuật toán khác nhau.

Bộ dữ liệu GTSRB đã được sử dụng để đánh giá các thuật toán nhận dạng biển báo giao thông khác nhau, bao gồm các phương pháp tiếp cận thị giác máy tính truyền thống và các phương pháp dựa trên học sâu. Điểm chuẩn cũng đã được sử dụng để đánh giá tác động của các yếu tố khác nhau đối với hiệu suất nhận dạng biển báo giao thông, chẳng hạn như ảnh hưởng của độ phân giải hình ảnh, không gian màu và kỹ thuật tăng cường dữ liệu.

3.6 PID Controller cho module điều khiển

Bộ điều khiển PID (Proportional-Integral-Derivative) là thuật toán điều khiển phản hồi được sử dụng rộng rãi nhằm điều chỉnh đầu ra của hệ thống bằng cách liên tục điều chỉnh đầu vào dựa



Hình 3.27: Bộ điều khiển PID

trên sai số giữa điểm đặt mong muốn và đầu ra thực tế. Hoạt động của bộ điều khiển bao gồm ba thành phần chính: tỷ lệ, tích phân và đạo hàm. Số hạng tỷ lệ trực tiếp chia tỷ lệ điều chỉnh đầu vào dựa trên sai số hiện tại, số hạng tích phân tích lũy các sai số trong quá khứ để chống lại các sai số ở trạng thái ổn định và số hạng đạo hàm dự đoán xu hướng sai số trong tương lai để đưa ra hành động khắc phục. Bằng cách cân bằng ba thành phần này, bộ điều khiển PID có thể giảm thiểu sai số một cách hiệu quả và duy trì sự ổn định trong các hệ thống điều khiển khác nhau, từ điều khiển nhiệt độ đơn giản đến các quy trình công nghiệp phức tạp.

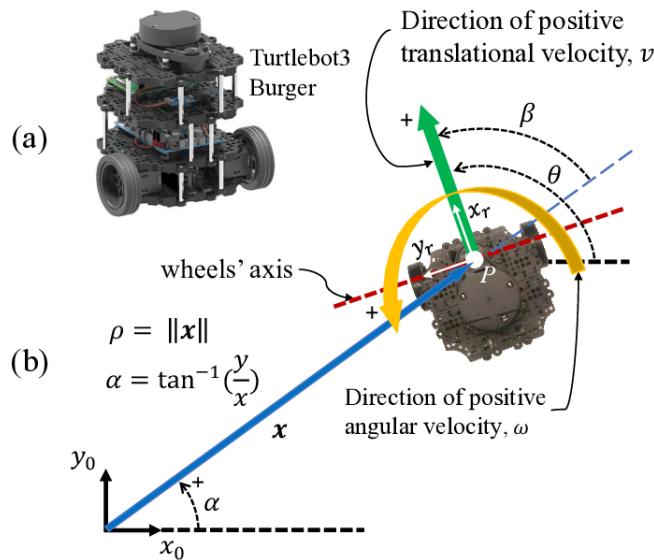
Bộ điều khiển PID xác định việc điều chỉnh phù hợp dựa trên sai số bằng cách kết hợp các số hạng tỷ lệ, tích phân và đạo hàm. Phân tích về cách mỗi thuật ngữ đóng góp vào việc điều chỉnh:

- **Số hạng tỷ lệ (P):** Số hạng tỷ lệ tính toán sự điều chỉnh bằng cách nhân sai số với mức tăng tỷ lệ (K_p). Nó trực tiếp điều chỉnh dựa trên lỗi hiện tại. Mức tăng tỷ lệ cao hơn dẫn đến mức điều chỉnh lớn hơn đối với một sai số nhất định. Tuy nhiên, mức tăng quá cao có thể dẫn đến điều chỉnh quá lố hoặc mất ổn định, trong khi mức tăng quá thấp có thể dẫn đến phản hồi chậm.
- **Số hạng tích phân (I):** Số hạng tích phân tính toán mức điều chỉnh bằng cách lấy tích phân sai số tích lũy theo thời gian và nhân nó với mức tăng tích phân (K_i). Nó giúp giải quyết các sai số ở trạng thái ổn định và loại bỏ sai lệch trong hệ thống. Số hạng tích phân liên tục điều chỉnh đầu vào dựa trên lịch sử sai số. Mức tăng tích phân cao hơn làm tăng khả năng điều chỉnh các sai số thường xuyên. Tuy nhiên, mức tăng quá mức có thể dẫn đến dao động hoặc mất ổn định.
- **Số hạng đạo hàm (D):** Số hạng đạo hàm xác định mức điều chỉnh bằng cách tính tỷ lệ thay đổi của sai số và nhân với mức tăng đạo hàm (K_d). Nó đưa ra hành động khắc phục dựa trên dự đoán về xu hướng sai số trong tương lai. Số hạng đạo hàm giúp làm giảm phản ứng và cải thiện độ ổn định. Nó phản ứng với những thay đổi đột ngột của sai số, giảm độ điều chỉnh quá lố. Tương tự như các số hạng khác, mức tăng đạo hàm cao hơn sẽ khuếch đại sự điều chỉnh, nhưng mức tăng quá lớn có thể gây ra nhiễu hoặc khuếch đại nhiễu đo lường.

Điều chỉnh cuối cùng được áp dụng cho hệ thống là tổng của các đóng góp riêng lẻ từ các số hạng tỷ lệ, tích phân và đạo hàm. Bộ điều khiển PID liên tục tính toán và cập nhật sự điều

chỉnh này dựa trên sai số hiện tại, tích hợp các sai số trong quá khứ và dự đoán xu hướng sai số trong tương lai. Bằng cách điều chỉnh mức tăng một cách thích hợp, bộ điều khiển PID có thể đạt được sự cân bằng giữa khả năng đáp ứng, độ ổn định và độ chính xác trong việc kiểm soát đầu ra của hệ thống.

3.7 Áp dụng PID để điều khiển Turtlebot3



Hình 3.28: TurtleBot3 và các vận tốc

Để áp dụng bộ điều khiển PID vào việc điều khiển robot trong hệ thống nhận diện làn đường, cần xác định một thông số phù hợp cho việc điều khiển. Đối với việc mô phỏng robot Turtlebot3, có hai thông số chính trong việc điều khiển robot được nhà sản xuất cung cấp là vận tốc tuyến tính (linear velocity) và vận tốc góc (angular velocity).

Vận tốc tuyến tính, trong bối cảnh TurtleBot3 hoặc bất kỳ robot di động nào, đề cập đến tốc độ robot di chuyển theo đường thẳng dọc theo hướng hiện tại của nó. Nó thể hiện tốc độ robot di chuyển tiến hoặc lùi mà không thay đổi hướng của nó.

Vận tốc tuyến tính thường được biểu thị bằng mét trên giây (m/s) và có thể dương (di chuyển về phía trước) hoặc âm (di chuyển về phía sau). Độ lớn của vận tốc tuyến tính xác định tốc độ của robot, trong khi dấu chỉ hướng chuyển động.

Trong TurtleBot3, vận tốc tuyến tính được điều khiển bằng cách điều chỉnh tốc độ động cơ hoặc vận tốc của bánh xe. Bằng cách tăng hoặc giảm tốc độ động cơ bằng nhau, robot sẽ di chuyển tiến hoặc lùi với tốc độ mong muốn. Vận tốc tuyến tính có thể được đặt thành các giá trị cụ thể để đạt được các tốc độ khác nhau, cho phép robot điều hướng ở các vận tốc khác nhau.

Vận tốc góc, trong bối cảnh của TurtleBot3 hoặc bất kỳ robot di động nào, đề cập đến tốc độ quay hoặc thay đổi hướng của robot xung quanh trục thẳng đứng của nó. Nó đại diện cho tốc độ mà robot có thể quay hoặc lái.

Vận tốc góc thường được đo bằng radian trên giây (rad/s) và có thể dương hoặc âm, biểu thị hướng và độ lớn của chuyển động quay. Vận tốc góc dương biểu thị sự quay ngược chiều kim

đồng hồ, trong khi vận tốc góc âm biểu thị sự quay theo chiều kim đồng hồ.

Trong TurtleBot3, vận tốc góc được kiểm soát bằng cách điều chỉnh sự khác biệt về tốc độ động cơ hoặc vận tốc góc của bánh xe. Bằng cách tăng tốc độ động cơ ở một bên và giảm tốc độ ở phía bên kia, robot bắt đầu quay theo hướng mong muốn. Độ lớn của vận tốc góc quyết định tốc độ quay.

Đối với hệ thống nhận diện làn đường, robot sẽ được mô phỏng ở điều kiện lý tưởng là giữ mức vận tốc tuyến tính cố định và điều chỉnh vận tốc góc thông qua kết quả tính toán của bộ điều khiển PID. Việc này nhằm giảm thiểu các vấn đề có thể xảy ra cho các module nhận diện làn đường và biển báo vì khi liên tục thay đổi vận tốc tuyến tính có thể tăng thêm độ phức tạp hoặc sai lệch kết quả tính toán để xác định làn đường và biển báo.

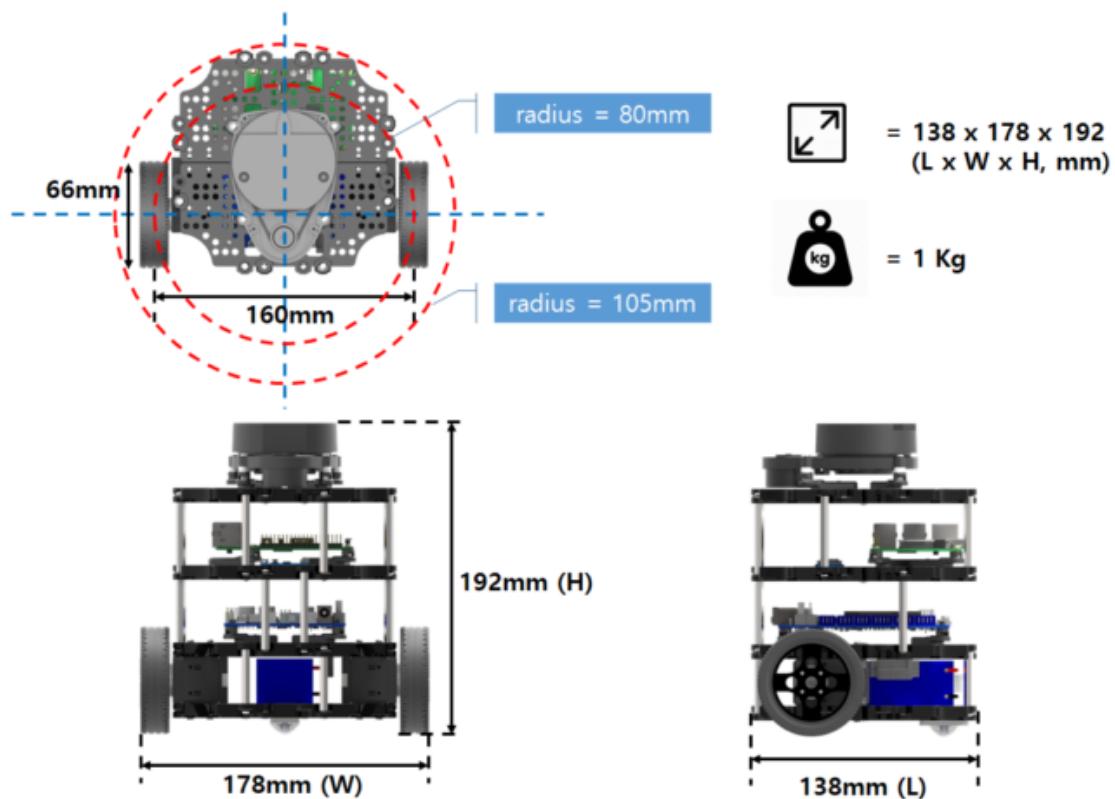
Đối với vận tốc góc, bộ điều khiển PID cần được cung cấp thông tin về độ lệch của vị trí hiện tại của robot so với tâm làn đường để tính toán vận tốc cần cung cấp cho robot để đảm bảo robot luôn nằm giữa làn đường trong quá trình mô phỏng,

Chương 4

Hiện thực hệ thống

4.1 TurtleBot3

TurtleBot3 Burger



Hình 4.1: TurtleBot3

TurtleBot3 là một nền tảng robot di động được thiết kế cho giáo dục, nghiên cứu và tạo mẫu. Đây là một nền tảng mã nguồn mở dựa trên Hệ điều hành rô-bốt (ROS) phổ biến và cung cấp một cách linh hoạt và giá cả phải chăng để thử nghiệm rô-bốt.[10]

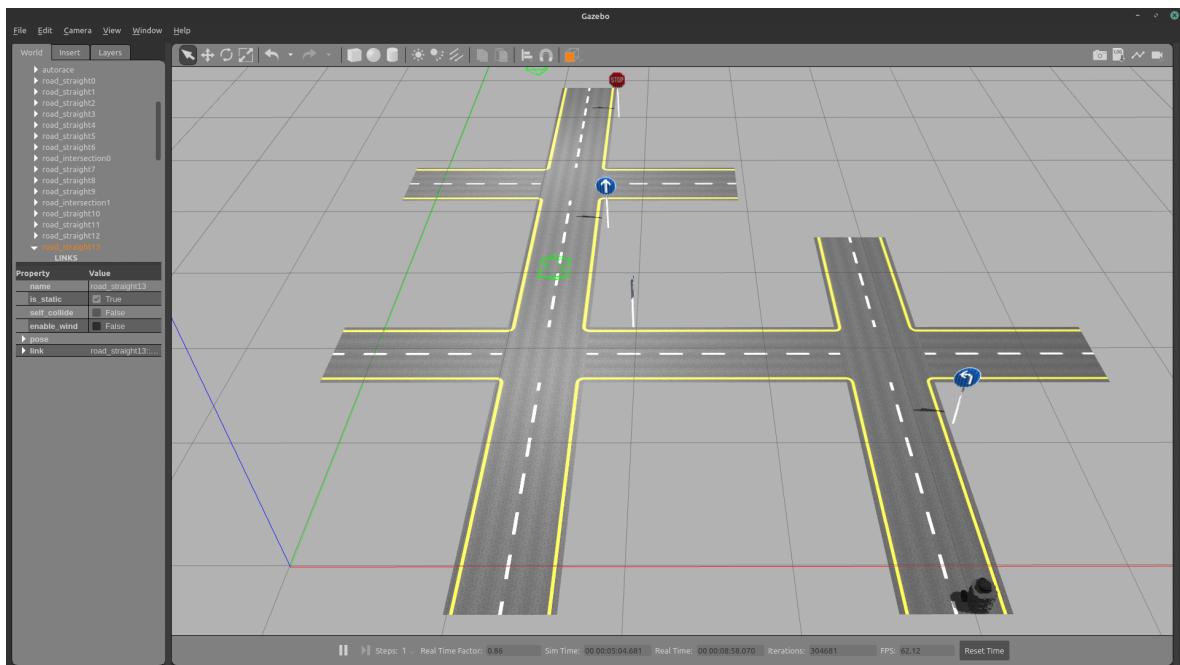
TurtleBot3 có nhiều biến thể khác nhau, bao gồm Burger, Waffle và Waffle Pi. Mỗi biến thể có bộ tính năng và khả năng riêng, nhưng tất cả đều được thiết kế để dễ sử dụng và tùy chỉnh.

TurtleBot3 được trang bị nhiều loại cảm biến, bao gồm máy ảnh, nắp đậy và đơn vị đo lường quán tính (IMU), cho phép nó nhận biết môi trường và điều hướng tự động. Nó cũng có nhiều cổng và giao diện mở rộng, cho phép người dùng thêm các cảm biến và bộ truyền động tùy chỉnh của riêng họ vào nền tảng.

TurtleBot3 được thiết kế theo mô-đun và có thể tùy chỉnh cao, đồng thời người dùng có thể dễ dàng hoán đổi các thành phần và thêm các tính năng mới khi cần. Nó cũng được thiết kế để có thể xách tay và dễ vận chuyển, khiến nó trở thành lựa chọn phổ biến cho các chương trình tiếp cận và giáo dục người máy.

Nhìn chung, TurtleBot3 là một nền tảng linh hoạt và mạnh mẽ, cung cấp một cách tuyệt vời để tìm hiểu về người máy, thử nghiệm các công nghệ mới và thử nghiệm các ứng dụng mới. Thiết kế mã nguồn mở và cộng đồng người dùng và nhà phát triển tích cực khiến nó trở thành lựa chọn lý tưởng cho bất kỳ ai quan tâm đến người máy.[5]

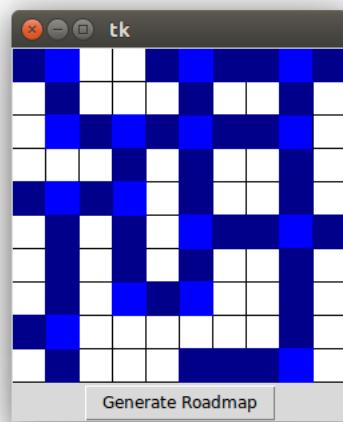
4.2 Mô phỏng



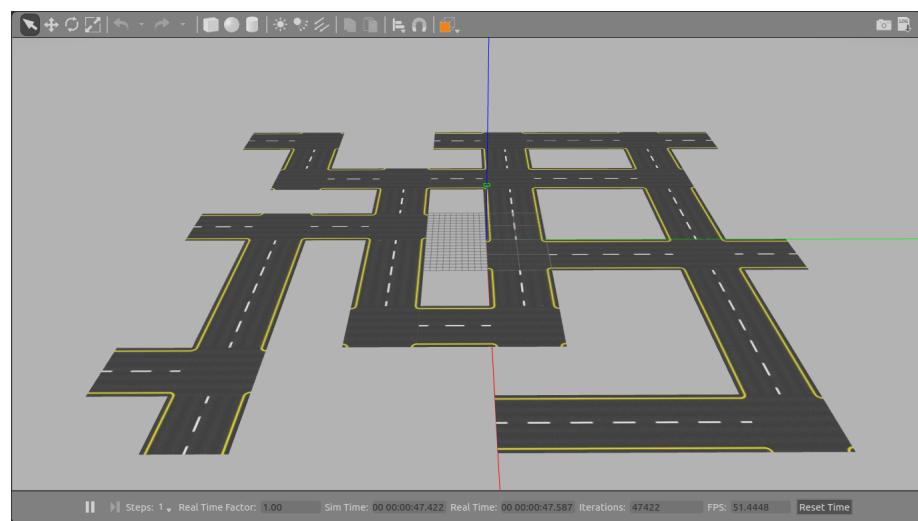
Hình 4.2: Bản đồ mô phỏng làn đường được sử dụng với Turtlebot3 và các biển báo

Bản đồ mô phỏng được xây dựng trong Gazebo với sự hỗ trợ của công cụ Roadmap Generator do Sarathkrishnan Ramesh phát triển giúp nhanh chóng tạo ra một bản đồ với làn đường và các giao lộ gần giống với bản đồ đường giao thông trong thực tế. Đường giao thông mô phỏng gồm 2 làn đường, mục tiêu của robot là di chuyển trong phạm vi 1 làn đường giới hạn bởi vạch đứt ở giữa và vạch liền bên phải.

Robot mô phỏng Turtlebot3 được cung cấp bởi ROBOTIS - nhà sản xuất Turtlebot với đầy đủ chức năng di chuyển tương đương với robot model Burger trong thực tế. Robot mô phỏng còn được trang bị sẵn module camera giúp quan sát được hình ảnh trong thế giới mô phỏng trong Gazebo.

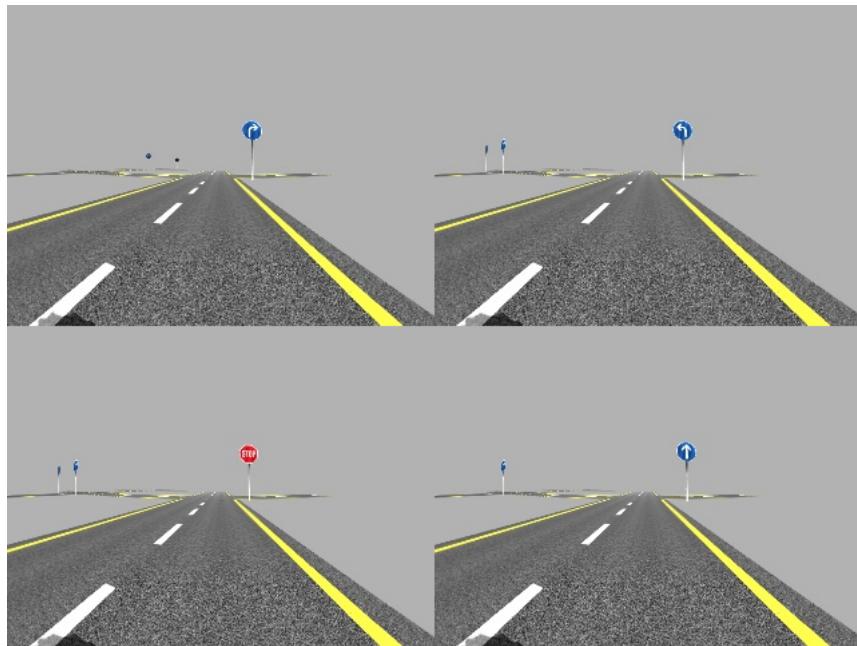


Hình 4.3: Công cụ tạo bản đồ làn đường



Hình 4.4: Bản đồ làn đường được tạo

4.3 Module nhận diện làn đường



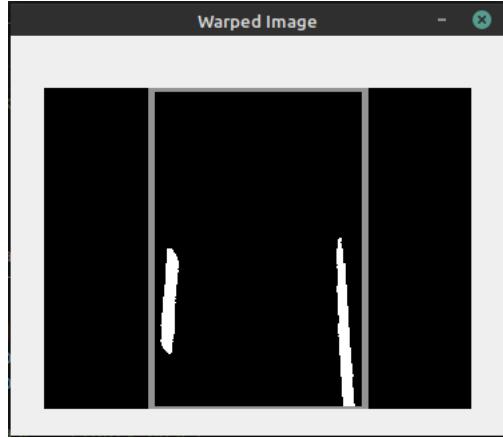
Hình 4.5: Góc nhìn từ camera của robot

Khi thực hiện binary thresholding, cần thử nghiệm với các giá trị threshold khác nhau để đạt được khả năng nhận diện được các làn đường tốt nhất và hạn chế sự ảnh hưởng của nhiễu từ môi trường xung quanh. Ngoài ra, còn phải xem xét điều kiện ánh sáng của môi trường ảnh hưởng tới khả năng nhận diện. [12]

Trong môi trường mô phỏng được kiểm soát, ít có hình ảnh gây nhiễu, điều kiện chiếu sáng tốt giúp khả năng nhận diện được tối ưu, với mức threshold 190, có thể nhận diện rõ ràng được 2 vạch kẻ làn đường được đặt giá trị pixel thành 255 màu trắng. Tất cả các vật thể khác trong môi trường không thuộc 2 làn đường được đặt thành pixel đen. Nếu chọn mức threshold thấp hơn thì chưa thể nhận diện chính xác làn đường do các thành phần khác của môi trường có đủ độ sáng cũng được đặt thành pixel 255. Sau đó histogram được tính toán, tìm điểm đặt sliding window và tìm được vị trí các điểm thuộc làn đường.



Hình 4.6: Binary thresholding ở mức 120



Hình 4.7: Binary thresholding ở mức 190

4.4 Module phát hiện biển báo

Trong khuôn khổ đề tài, một phần bộ dữ liệu GTSDB được sử dụng để train model nhận diện biển báo với 1123 hình ảnh đường giao thông trên thực tế có sự xuất hiện của các biển báo. Quá trình train model được thực hiện trên Google Colab, tập dữ liệu được lựa chọn được tải lên Roboflow, model được sử dụng để train là yolov8s với 100 epoch.

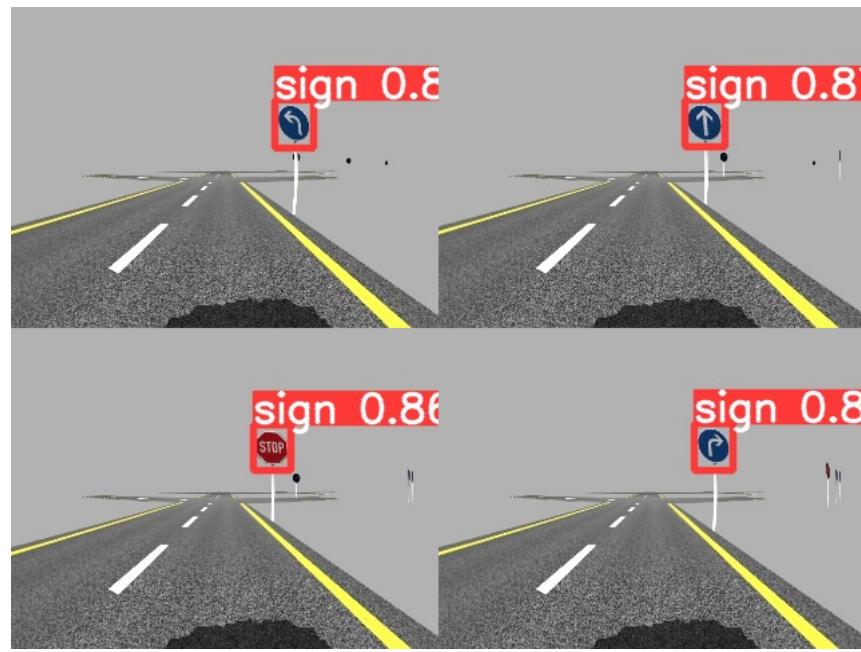
```
100 epochs completed in 0.494 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 22.5MB
Optimizer stripped from runs/detect/train/weights/best.pt, 22.5MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.0.20 Python-3.10.6 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11125971 parameters, 0 gradients, 28.4 GFLOPs
    Class      Images   Instances     Box(P)      R      mAP50      mAP50-95: 100% 7/7 [00:05<00:00,  1.28it/s]
        all       218      350      0.88      0.786      0.847      0.428
Speed: 0.5ms pre-process, 3.1ms inference, 0.0ms loss, 2.7ms post-process per image
Results saved to runs/detect/train
```

Hình 4.8: Model phát hiện biển báo

Model được train để phát hiện một class duy nhất là biển báo (sign), kết quả nhận được là bounding box chứa biển báo với độ chính xác cao. Hình ảnh gốc từ camera được crop theo toạ độ của bounding box để được hình ảnh chỉ chứa biển báo giao thông. Sau đó hình ảnh này được tiếp tục đưa sang module phân loại biển báo để tiến hành phân loại cụ thể loại biển báo xuất hiện trong ảnh. Việc tách riêng giữa module phát hiện và nhận diện biển báo giúp cho độ chính xác của các module đạt được tốt hơn khi mỗi module được huấn luyện để thực hiện một nhiệm vụ.

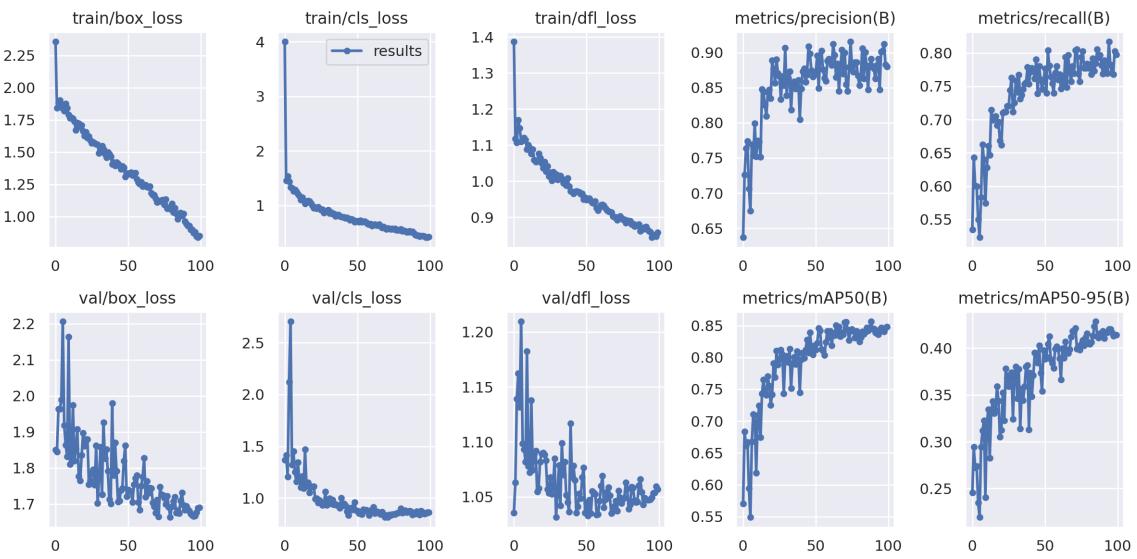
Ngoài ra, từ toạ độ của bounding box, có thể tính được diện tích bounding box nhằm xác định khoảng cách tương đối giữa robot và biển báo. Do các biển báo được đặt ở khoảng cách cố định trước các giao lộ nên khi xác định diện tích của biển báo đủ lớn, tức khoảng cách tới giao lộ đủ gần thì module điều khiển robot có thể thực hiện quá trình rẽ trái phải hay đi thẳng qua giao lộ hoặc dừng lại theo biển báo được phát hiện. [6]



Hình 4.9: Phát hiện biển báo



Hình 4.10: Kiểm tra sau khi train trên bộ dữ liệu GTSDB



Hình 4.11: Một số thông số đánh giá model

4.4.1 Đánh giá model

Khi đánh giá một mô hình phát hiện đối tượng, một số số liệu thường được sử dụng để đánh giá hiệu suất của nó. Dưới đây là một số số liệu chính:

Độ chính xác: Độ chính xác đo lường độ chính tổng thể của các dự đoán của mô hình. Nó tính toán tỷ lệ các đối tượng được phát hiện chính xác trên tổng số đối tượng. Tuy nhiên, chỉ độ chính xác có thể không đủ để phát hiện đối tượng vì nó có thể bị ảnh hưởng nặng nề bởi sự mất cân bằng lớp hoặc sự hiện diện của nhiều đối tượng trong một ảnh.

Độ chính xác: Độ chính xác đo tỷ lệ các đối tượng được phát hiện chính xác trong số tất cả các đối tượng được dự đoán là positive. Nó tập trung vào tính đúng đắn của những dự đoán tích cực. Độ chính xác cao cho thấy tỷ lệ positive giả thấp, điều đó có nghĩa là mô hình có khả năng tránh phát hiện sai.

Thu hồi (Độ nhạy): Thu hồi, còn được gọi là độ nhạy hoặc tỷ lệ positive thực, đo tỷ lệ các đối tượng được phát hiện chính xác trong số tất cả các đối tượng positive với sự thật trên cơ sở. Nó tập trung vào khả năng của mô hình trong việc tìm ra tất cả các trường hợp tích cực. Tỷ lệ thu hồi cao cho thấy tỷ lệ negative giả thấp, nghĩa là mô hình có khả năng phát hiện tốt các đối tượng liên quan.

Giao trên Hợp (IoU): IoU tính toán sự chồng chéo giữa các hộp giới hạn được dự đoán và các hộp giới hạn thực tế. Nó xác định mức độ các hộp giới hạn được dự đoán sẽ căn chỉnh với các đối tượng thực sự như thế nào. IoU thường được sử dụng để xác định xem phát hiện được coi là positive thật hay positive giả. Các ngưỡng phổ biến cho IoU bao gồm 0,5 và 0,75.

Độ chính xác trung bình (AP): Độ chính xác trung bình tính toán độ chính xác trung bình trên các ngưỡng IoU khác nhau. Nó đánh giá sự cân bằng giữa việc thu hồi độ chính xác ở các mức độ tin cậy phát hiện đối tượng khác nhau. AP tóm tắt hiệu suất tổng thể của mô hình qua các mức ngưỡng phát hiện đối tượng khác nhau.

Độ chính xác trung bình trung bình (mAP): mAP là giá trị trung bình của điểm chính xác

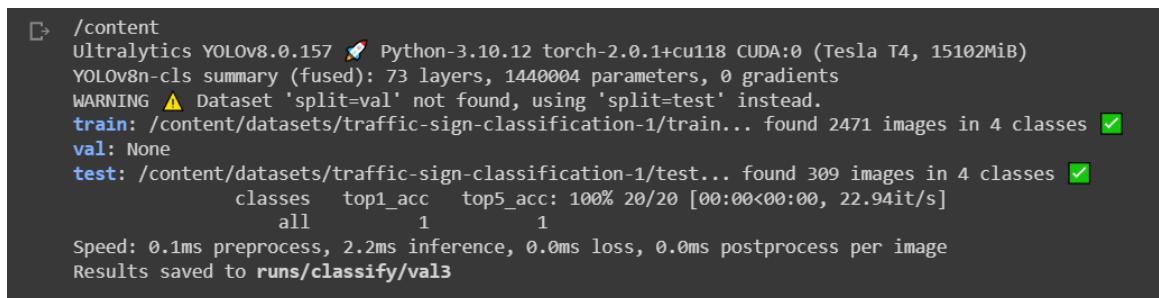
trung bình trên các lớp đối tượng khác nhau. Nó cung cấp thước đo tổng thể về hiệu suất của mô hình trên nhiều loại đối tượng. mAP thường được sử dụng để so sánh và xếp hạng các mô hình phát hiện đối tượng khác nhau.

Điểm F1: Điểm F1 kết hợp độ chính xác và khả năng thu hồi thành một số liệu duy nhất. Nó là giá trị trung bình hài hòa của độ chính xác và khả năng thu hồi, cung cấp thước đo cân bằng về hiệu suất của mô hình. Điểm F1 rất hữu ích khi có sự mất cân bằng giữa số lượng trường hợp tích cực và tiêu cực.

Các số liệu này cùng nhau cung cấp thông tin chi tiết về hiệu suất của mô hình phát hiện đối tượng bằng cách đánh giá độ chính xác, độ chính xác, thu hồi, IoU và khả năng phát hiện tổng thể của nó. Điều quan trọng là phải xem xét kết hợp các số liệu này để có được sự hiểu biết toàn diện về điểm mạnh và điểm yếu của mô hình.

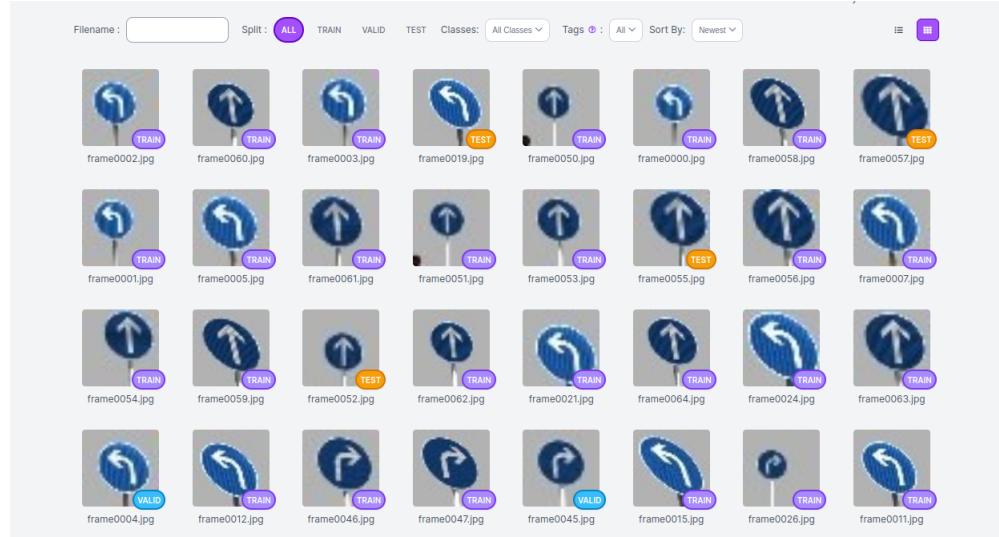
4.5 Module phân loại biển báo

Trong khuôn khổ đề tài, một phần bộ dữ liệu GTSRB được sử dụng để train model phân loại biển báo với 3089 hình ảnh biển báo giao thông. Ngoài ra, hình ảnh của các biển báo trong môi trường mô phỏng cũng được thu thập đưa vào bộ dữ liệu. Số lượng hình ảnh của mỗi biển báo được thu thập gần bằng nhau. Model được sử dụng để train là YOLOv8n-cls với 100 epochs. Sau khi nhận được hình ảnh chỉ chứa biển báo từ module phát hiện biển báo, module phân loại tiến hành đánh giá hình ảnh biển báo thuộc loại nào trong số 4 loại biển báo Dừng, Đi thẳng, Rẽ trái, Rẽ phải.



```
↳ /content
ultralytics YOLOv8.0.157 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8n-cls summary (fused): 73 layers, 1440004 parameters, 0 gradients
WARNING ▲ Dataset 'split=val' not found, using 'split=test' instead.
train: /content/datasets/traffic-sign-classification-1/train... found 2471 images in 4 classes ✓
val: None
test: /content/datasets/traffic-sign-classification-1/test... found 309 images in 4 classes ✓
classes top1_acc top5_acc: 100% 20/20 [00:00<00:00, 22.94it/s]
all 1 1
Speed: 0.1ms preprocess, 2.2ms inference, 0.0ms loss, 0.0ms postprocess per image
Results saved to runs/classify/val3
```

Hình 4.12: Model phân loại biển báo

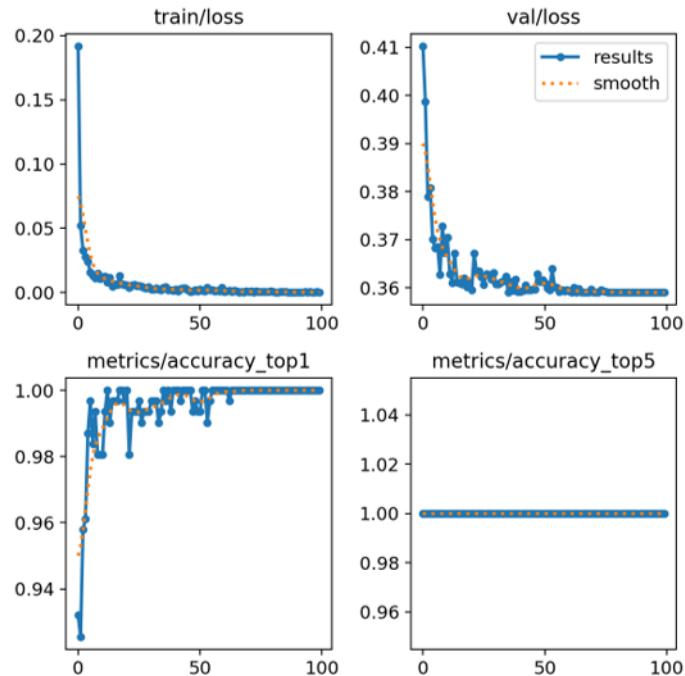


Hình 4.13: Bổ sung các biển báo trong mô phỏng vào bộ dữ liệu

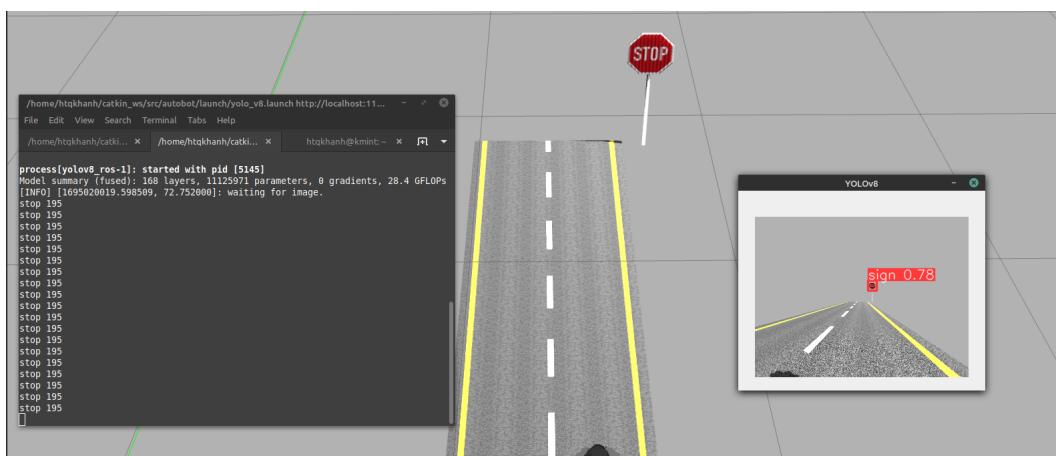


Hình 4.14: Kiểm tra sau khi train trên bộ dữ liệu GTSRB

Đối với model phân loại biển báo, thông số đánh giá quan trọng nhất là độ chính xác của model. Nó được tính dựa trên số lượng mẫu được phân loại chính xác trên tổng số mẫu. Tuy nhiên, nó có những hạn chế, đặc biệt là khi xử lý các tập dữ liệu mất cân bằng. Tập dữ liệu mất cân bằng là tập dữ liệu có số lượng mẫu trong mỗi lớp khác nhau đáng kể. Trong những trường hợp như vậy, điểm có độ chính xác cao có thể phản ánh hiệu suất thực sự của mô hình vì nó có thể bị sai lệch thiên về nhóm đa số. Vì vậy cần đảm bảo trong bước thu thập dữ liệu, số lượng hình ảnh biển báo của mỗi loại được thu thập gần tương đương nhau, tránh trường hợp có quá nhiều hình ảnh của một số biển báo và quá ít hình ảnh của một số biển báo khác.

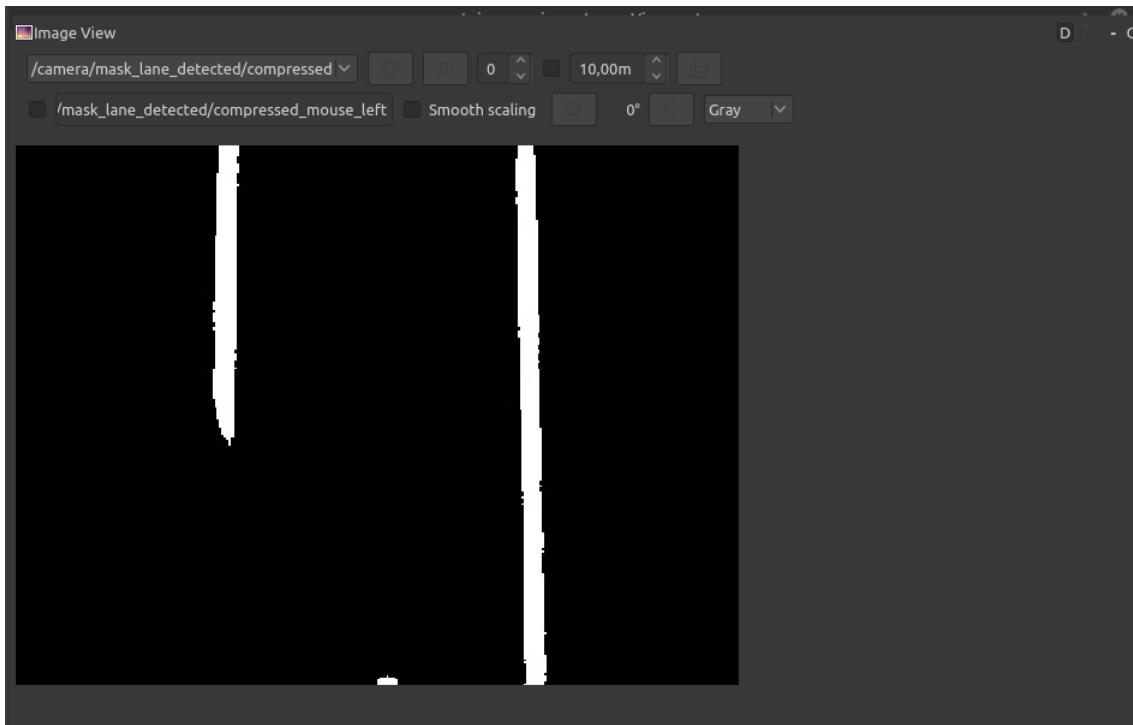


Hình 4.15: Một số thông số đánh giá model

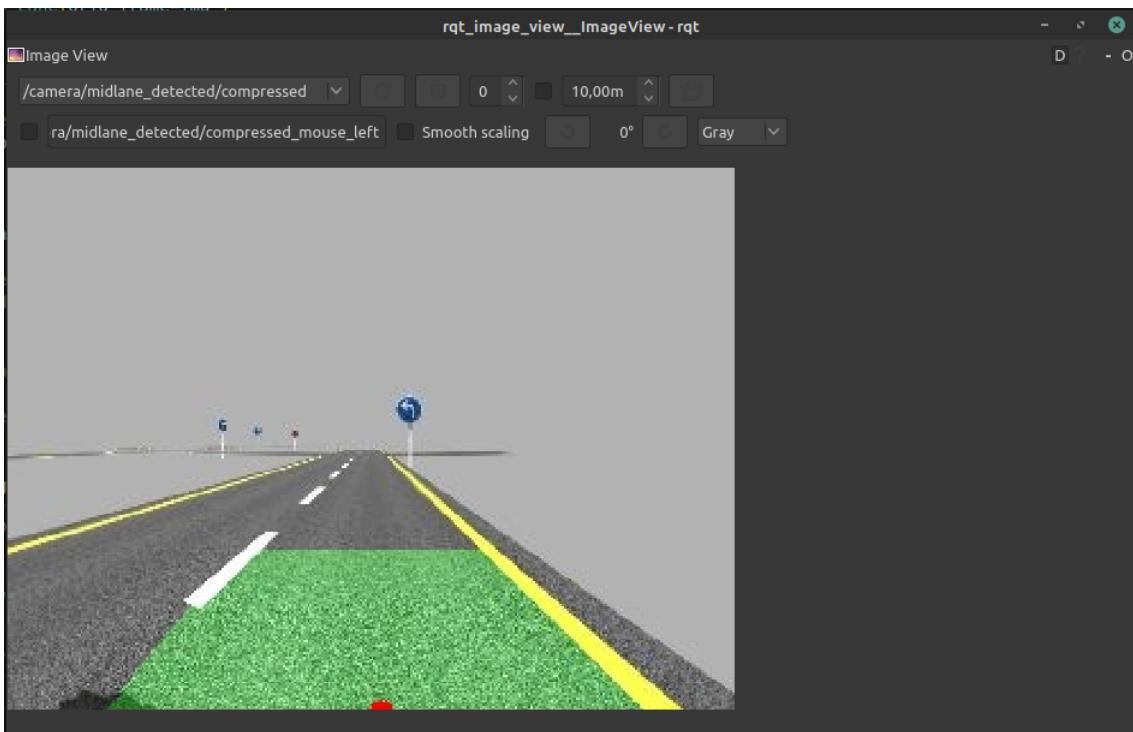


Hình 4.16: Biển báo được nhận diện và diện tích bounding box

4.6 Module điều khiển



Hình 4.17: Làn đường trong bird's eye view



Hình 4.18: Hình ảnh làn đường được phát hiện

Hình 4.19: Vị trí tâm làn đường

```
htqkhanh@kmint:~
```

File Edit View Search Terminal Tabs Help

/home/htqkh... x /home/htqkh... x htqkhanh@k... x htqkhanh@k... x

```
^Chtqkhanh@kmint:~$ rostopic echo /yolov8/area
data: "208"
...
data: "208"
```

Hình 4.20: Diện tích biển báo

Hình 4.21: Phân loại biển báo

```

        error = center - 160
        current_time = time.time()

        dt=current_time-self.prev_time
        self.integral= (self.integral + error)*dt

        self.angular_z += Kp * (error - self.lastError) + Kd * (error - 2 * self.prevError + self.lastError) +Ki *self.integral

        self.prevError = self.lastError
        self.lastError = error
    
```

Hình 4.22: Bộ điều khiển PID để tính vận tốc góc

```

htqkhanh@k... ~
File Edit View Search Terminal Tabs Help
htqkhanh@k... ✘ htqkhanh@k... ✘ htqkhanh@k... ✘ htqkhanh@k... ✘ htqkhanh@k... ✘
/
controller (autobot/controller.py)

ROS_MASTER_URI=http://localhost:11311

process[controller-1]: started with pid [47171]
[INFO] [1696281531.890683, 0.000000]: lane following controller is running
0.097013984298706055
0.017067119932396325
0.017110344741179768
0.01715549304729168
0.01722920030418572
0.017299525506749774
0.017340436718361135
0.017417216058609384
0.017445670254799255
0.017520385443675048
0.017564039742067695
0.017641875692632553
0.01770850578626884
0.017772010694797297
0.021385545401419567
0.02646526452050857

```

Hình 4.23: Vận tốc góc sau khi tính toán

Module điều khiển giữ cho robot giữ vị trí giữa làn đường được cung cấp bởi module phát hiện làn đường và điều chỉnh góc xoay khi robot di chuyển để không bị lệch ra khỏi tâm làn đường. Gia tốc tịnh tiến (twist.linear.x) được giữ cố định và chỉ gia tốc góc (twist.angular.y) được điều chỉnh. Các thông số được publish đến topic /cmd_vel để điều khiển robot. Module điều khiển nhận đầu vào từ module phát hiện làn đường, module phát hiện và module phân loại biển báo để bám theo làn đường di chuyển đến các ngã ba, ngã tư. Khi hình ảnh biển báo được phát hiện đủ gần, module điều khiển sẽ thực hiện rõ theo biển báo sang các làn đường tương ứng.

4.6.1 Tinh chỉnh PID

Khi điều chỉnh bộ điều khiển PID cho hệ thống ô tô tự lái với vận tốc tuyến tính và cố định, sẽ cần tập trung vào việc điều chỉnh bộ điều khiển PID để điều khiển vận tốc góc. Cần thực hiện các bước để điều chỉnh thủ công:

- Đặt mức tăng tích phân và đạo hàm về 0 ($Ki = Kd = 0$) và bắt đầu với giá trị mức tăng tỷ lệ thận trọng (Kp), chẳng hạn như $Kp = 1$.
- Quan sát phản ứng của hệ thống khi xe tự lái gặp sự thay đổi vận tốc góc mong muốn hoặc có sự xáo trộn.
- Nếu hệ thống có sai số ở trạng thái ổn định, hãy tăng dần hệ số khuếch đại tỷ lệ (Kp) để

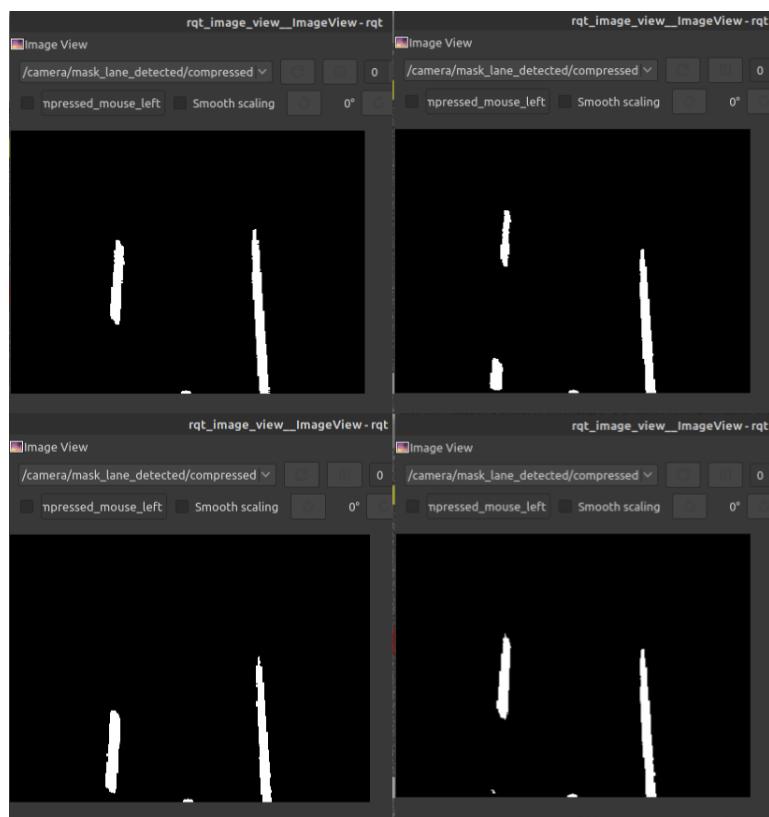
giảm sai số. Tuy nhiên, hãy thận trọng không tăng quá nhiều để tránh mất ổn định hoặc vượt quá mức.

- Khi mức tăng tỷ lệ được điều chỉnh để giảm thiểu sai số ở trạng thái ổn sai số, điều chỉnh mức tăng tích phân (Ki). Tăng Ki dần dần để giải quyết mọi lỗi ở trạng thái ổn định còn lại. Độ lợi tích phân cao hơn giúp loại bỏ các sai số nhỏ, dai dẳng theo thời gian.
- Cẩn trọng khi điều chỉnh mức tăng tích phân (Ki) vì giá trị lớn hơn có thể gây ra dao động hoặc mất ổn định. Nếu hệ thống bắt đầu dao động, giảm độ lợi tích phân và tìm sự cân bằng giúp giảm thiểu sai số trạng thái xác lập mà không làm mất đi tính ổn định.
- Sau khi tinh chỉnh mức tăng tích phân, tiến hành điều chỉnh mức tăng vi phân (Kd). Số hạng đạo hàm giúp làm giảm đáp ứng và giảm độ vọt lố và thời gian ổn định. Bắt đầu với mức tăng vi phân nhỏ và tăng dần để cải thiện phản hồi của hệ thống.
- Tương tự như mức tăng tích phân, khi điều chỉnh mức tăng vi phân vừa phải vì giá trị quá lớn có thể khuếch đại nhiễu trong hệ thống hoặc dẫn đến mất ổn định. Tìm một mức cân bằng mang lại phản hồi mượt mà hơn mà không tạo ra nhiễu hoặc dao động quá mức.
- Lặp lại quá trình điều chỉnh bằng cách thực hiện những điều chỉnh nhỏ đối với mức tăng, quan sát phản ứng của hệ thống và tinh chỉnh mức tăng cho phù hợp. Phải kiểm tra hệ thống trong các tình huống khác nhau, chẳng hạn như vận tốc góc, vòng quay và nhiễu loạn khác nhau để đảm bảo bộ điều khiển PID hoạt động tốt trong các điều kiện khác nhau.

Chương 5

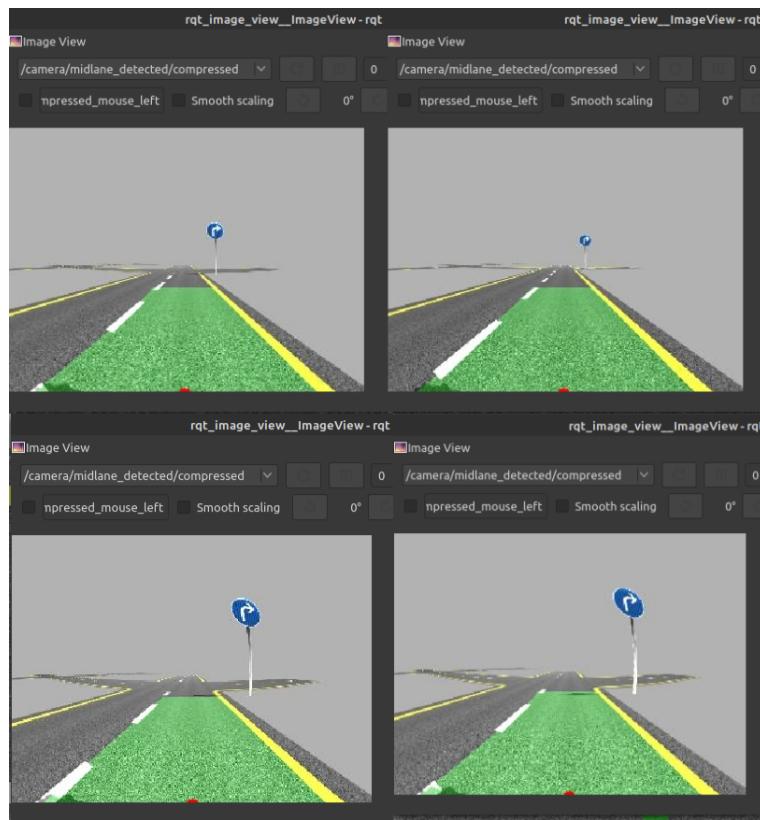
Kết quả thực nghiệm

5.1 Module nhận diện làn đường



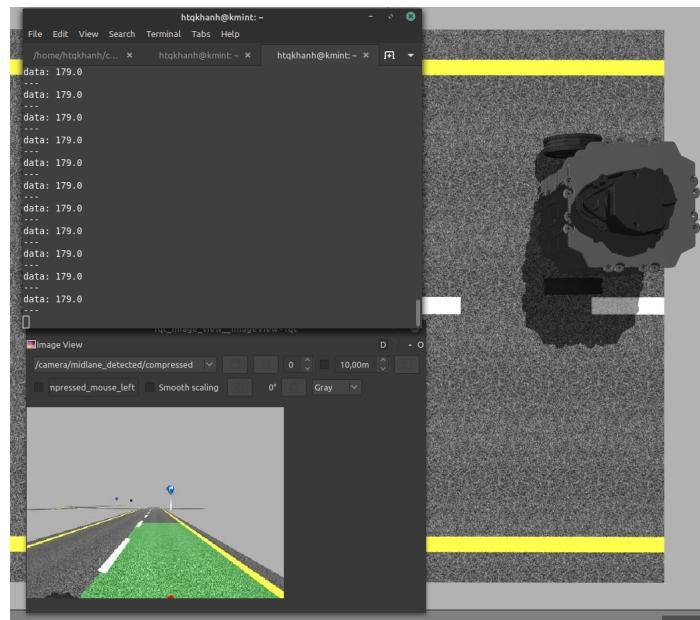
Hình 5.1: Làn đường nhận diện được

Module nhận diện làn đường lấy hình ảnh từ camera robot, trải qua các bước xử lý hình ảnh để phát hiện được làn đường và vị trí của robot so với tâm làn đường. Module hoạt động ổn định, phát hiện tốt vị trí làn đường trong điều kiện ánh sáng tốt của môi trường mô phỏng. Trong quá trình di chuyển, module vẫn phát hiện đúng vị trí làn đường và biểu diễn hình ảnh làn đường chính xác trong vùng quan tâm. Khi đến gần biển báo và giao lộ, robot tạm ngừng quan tâm đến thông tin từ module nhận diện làn đường do đến giao lộ không có làn đường để bám theo mà phải thực hiện rẽ theo quy luật được xác định, khi thực hiện rẽ xong robot lại tiếp tục phát hiện



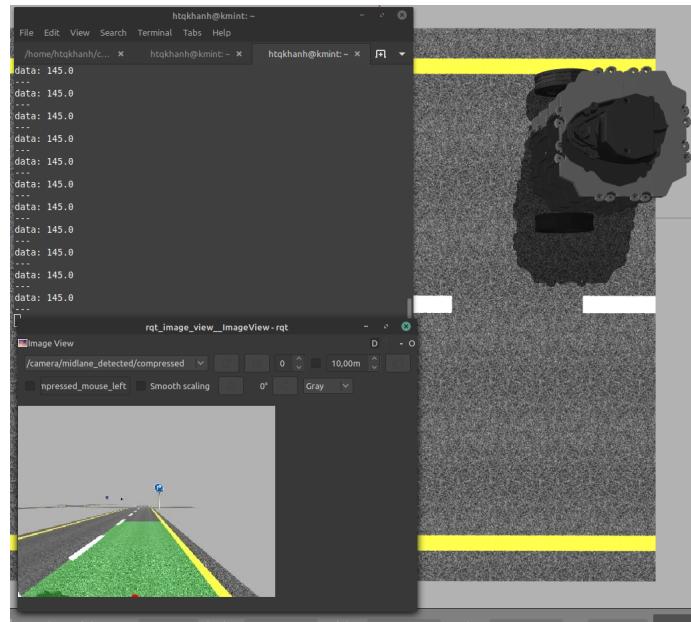
Hình 5.2: Biểu diễn làn đường nhận diện được

làn đường chính xác và di chuyển theo làn đường như bình thường.

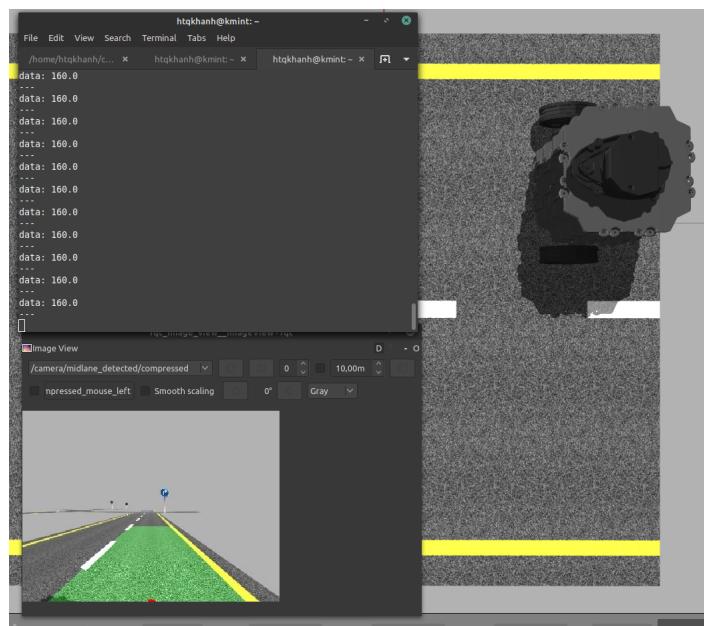


Hình 5.3: Vị trí tâm làn đường khi robot lệch trái

Trong quá trình di chuyển, robot luôn kiểm tra vị trí giữa làn đường trong hình ảnh từ camera, khi robot đang ở vị trí lệch trái so với tâm làn đường thì tâm làn đường sẽ có giá trị >160 tương đương với tâm làn đường nằm ở nửa bên phải trong khung hình từ camera còn nếu robot đang ở vị trí lệch phải so với tâm làn đường thì tâm làn đường sẽ có giá trị <160 tương đương với tâm



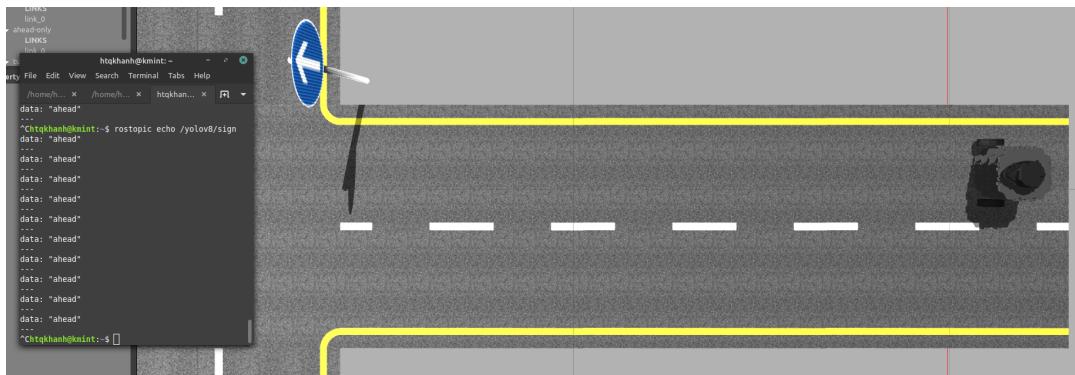
Hình 5.4: Vị trí tâm làn đường khi robot lệch phải



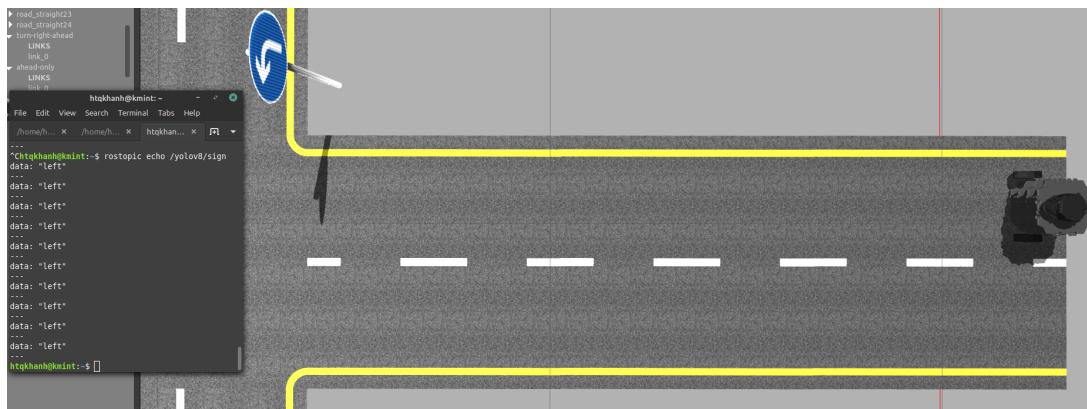
Hình 5.5: Vị trí tâm làn đường khi robot đang ở vị trí lý tưởng

làn đường nằm ở nửa bên trái khung hình. Khi robot di chuyển đúng ngay vị trí tâm làn đường thì tâm làn đường sẽ có giá trị 160. Vị trí tâm làn đường sẽ được cung cấp cho module điều khiển để tính toán vận tốc góc cần cung cấp cho robot để giúp robot đảm bảo vị trí càng gần tâm làn đường càng tốt.

5.2 Module phát hiện và phân loại biển báo giao thông



Hình 5.6: Nhận diện biển báo đi thẳng



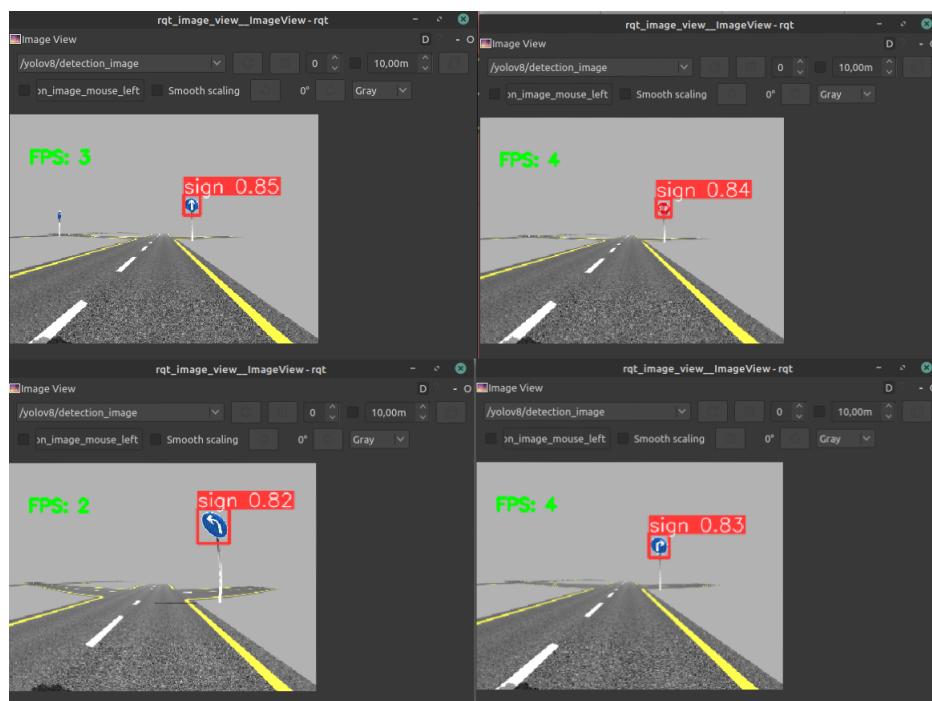
Hình 5.7: Nhận diện biển báo rẽ trái

Model phát hiện biển báo tìm được vị trí của các biển báo trong hình ảnh từ camera với độ chính xác cao, không xảy ra trường hợp trong hình ảnh không có biển báo mà model lại tìm thấy hình ảnh biển báo do nhầm lẫn hình ảnh bất kỳ trong môi trường với biển báo thật. Khoảng cách mà model có thể bắt đầu phát hiện thấy biển báo là khá xa do hình ảnh dùng để huấn luyện có bao gồm bộ dữ liệu GTSDB là hình ảnh được chụp trên đường phố thực tế với khoảng cách xa.

Model phân loại biển báo hoạt động chính xác nhất khi robot đứng yên, trong khi di chuyển thỉnh thoảng có các khung hình model nhầm lẫn giữa các biển báo. Vì vậy, để đạt độ chính xác cao nhất, hệ thống áp dụng cách xử lý dừng robot lại khi đến đủ gần với biển báo trong khoảng 2 giây để model thực hiện việc phân loại chính xác nhất. Ngoài ra, do khi đến gần sát biển báo camera của robot không còn nhìn thấy biển báo nên cần lưu lại thông tin biển báo đã phát hiện để thực hiện rẽ theo chỉ dẫn của biển báo.

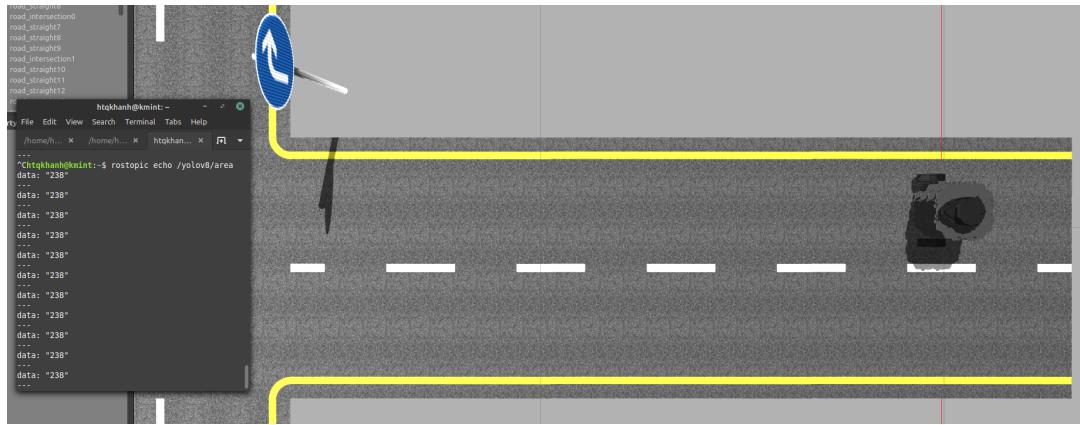
```
process[yolov8_ros-1]: started with pid [21312]
Model summary (fused): 168 layers, 11125971 parameters, 0 gradients, 28.4 GFLOPs
[INFO] [1697472686.496891, 2172.122000]: waiting for image.
0.413989782333374
0.37376880645751953
0.38018798828125
0.35974550247192383
0.3703420162200928
0.3630547523498535
0.3489038944244385
0.369107723236084
0.36816978454589844
0.3681049346923828
^C[yolov8_ros-1] killing on exit
shutting down processing monitor...
... shutting down processing monitor complete
done
```

Hình 5.8: Tốc độ xử lý của model phát hiện biển báo

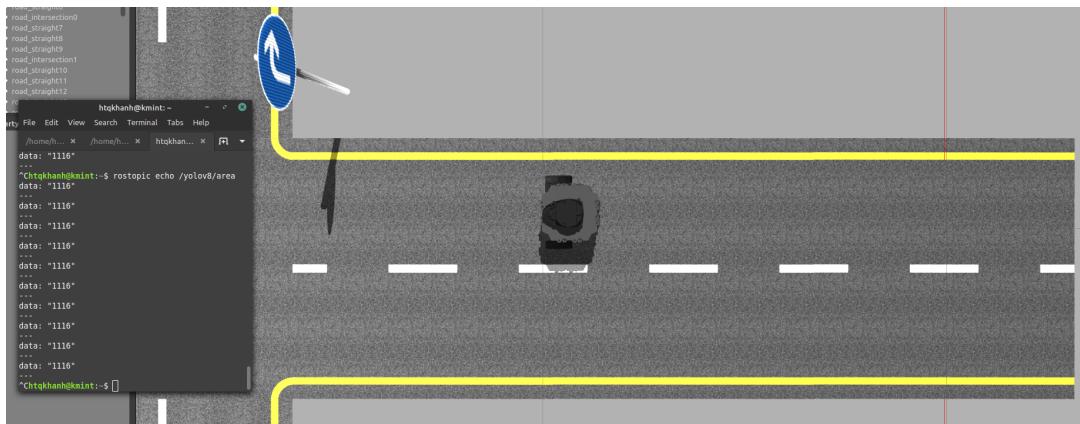


Hình 5.9: Tốc độ khung hình (FPS) của model phát hiện biển báo

Tốc độ xử lý của model phát hiện biển báo trung bình là 0.372s tương đương với khoảng 3-4 khung hình mỗi giây. Tốc độ xử lý này phụ thuộc vào phần cứng thực hiện tính toán cho model và độ phức tạp của model được chọn để thực hiện phát hiện biển báo. Với tốc độ xử lý này đảm bảo được khả năng phát hiện biển báo trong quá trình di chuyển của robot, tuy nhiên cần lựa chọn vận tốc di chuyển của robot phù hợp để đảm bảo khả năng xử lý của model phát hiện biển báo. Nếu robot di chuyển với tốc độ quá nhanh, có thể bỏ sót các khung hình chứa biển báo hoặc tiến đến quá gần ngã rẽ trước khi model kịp phân loại biển báo.



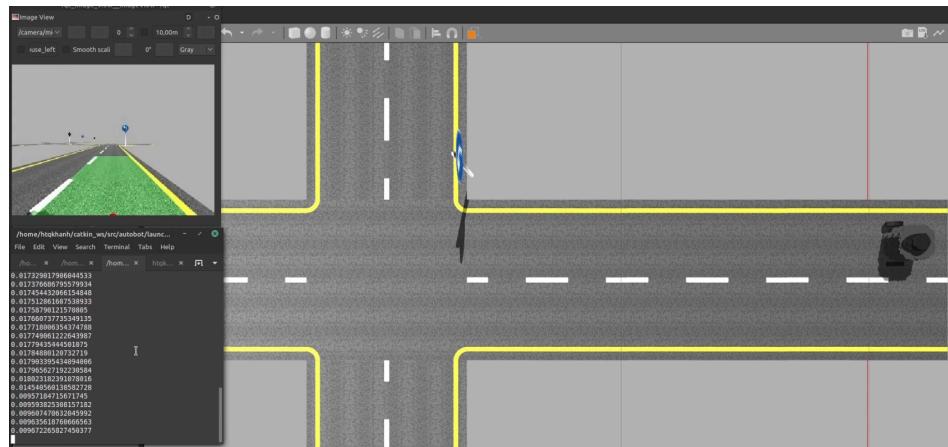
Hình 5.10: Diện tích biển báo ở khoảng cách xa



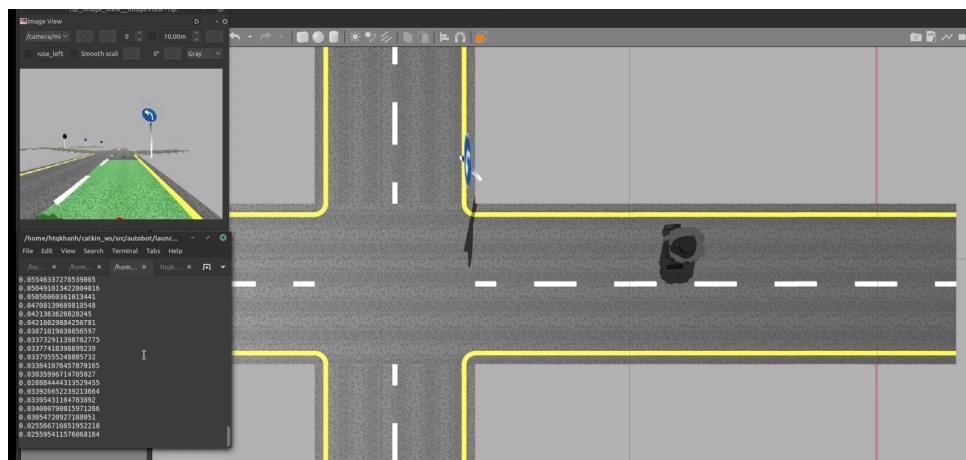
Hình 5.11: Diện tích biển báo ở khoảng cách gần

Diện tích biển báo được tính dựa trên tọa độ của bounding box từ model phát hiện biển báo. Những bounding box có độ tin cậy thấp ($confidence < 50\%$) sẽ được bỏ qua và chỉ giữ lại những bounding box có độ tin cậy cao, đồng thời trong những bounding box được giữ lại sẽ chỉ ưu tiên giữ lại bounding box có diện tích là lớn nhất (chứng tỏ gần nhất so với robot). Thông qua quá trình thử nghiệm, độ lớn diện tích bounding box được chọn là 1000, trong quá trình di chuyển, robot nhận thông tin về diện tích biển báo và khi đạt 1000 robot cần phải dừng lại và xác nhận loại biển báo vì khi di chuyển càng đến gần ngã rẽ và biển báo, sẽ có thời điểm robot không còn có thể nhìn thấy hình ảnh biển báo từ camera.

5.3 Module điều khiển

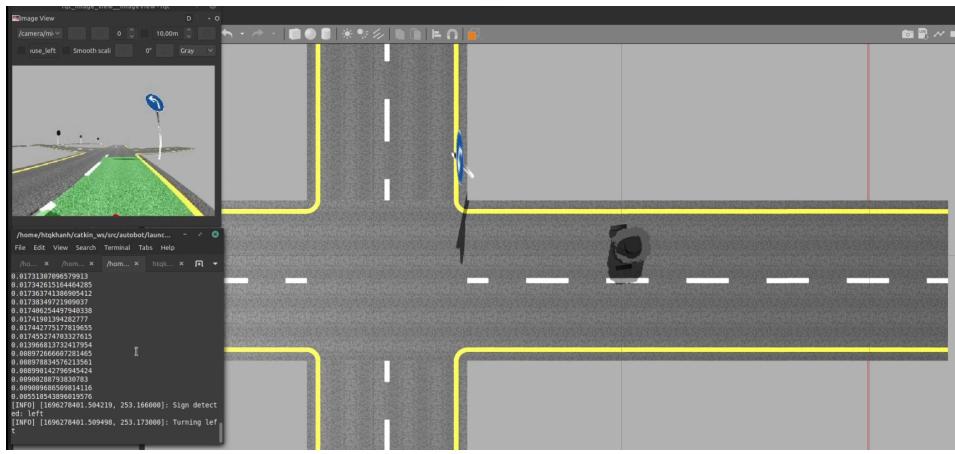


Hình 5.12: Vận tốc góc khi robot di chuyển

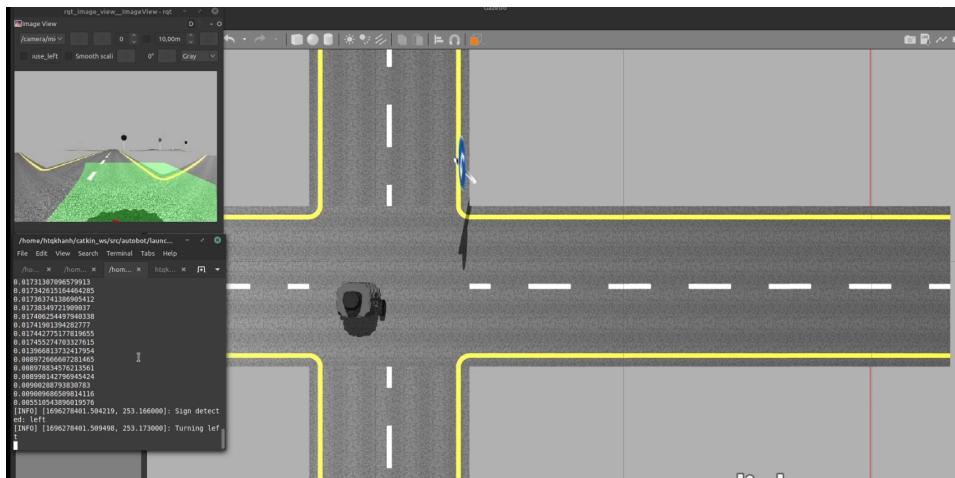


Hình 5.13: Vận tốc góc khi robot di chuyển

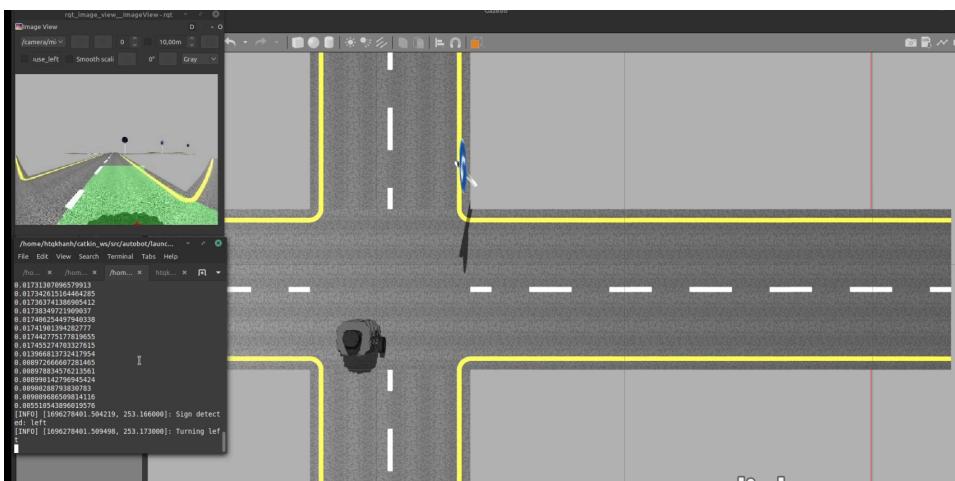
Module điều khiển nhận 3 thông tin từ các module khác về vị trí của robot so với tâm làn đường, diện tích biển báo và loại biển báo. Module giữ cố định vận tốc tuyến tính 0.08 m/s khi di chuyển và điều chỉnh vận tốc góc để không di chuyển lệch ra khỏi làn đường. Trong lúc di chuyển, module liên tục cập nhật vận tốc góc cho robot để điều chỉnh vị trí so với làn đường. Thông tin về diện tích biển báo giúp xác định vị trí tương đối gần xa giữa robot và biển báo. Khi đến gần biển báo tương đương với diện tích đủ lớn, robot ngay lập tức dừng lại để xác nhận thông tin phân loại biển báo, lưu giữ và thực hiện rẽ.



Hình 5.14: Robot phát hiện biển báo và bắt đầu quá trình rẽ trái



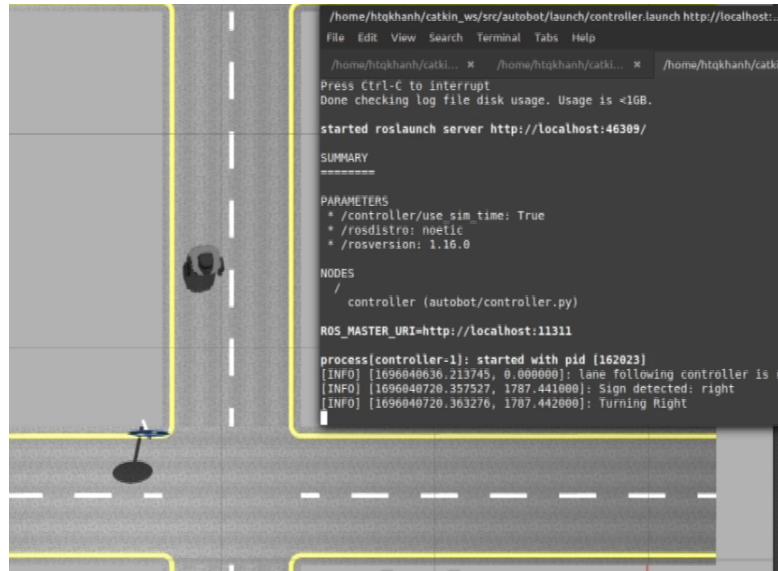
Hình 5.15: Robot đang rẽ trái



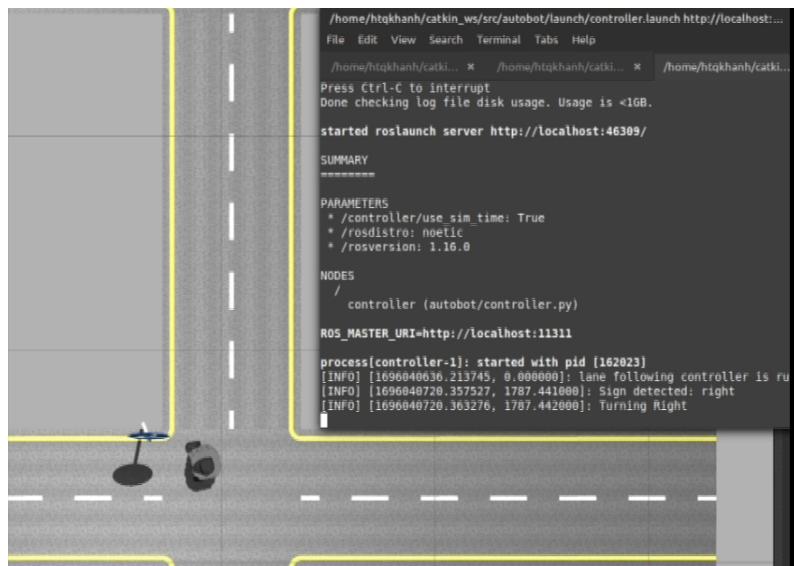
Hình 5.16: Robot sắp kết thúc quá trình rẽ và tiếp tục với module nhận diện làn đường

Khi phát hiện biển báo và thực hiện rẽ trái, robot bắt đầu di chuyển thẳng tới phía trước giao lộ với vận tốc tuyến tính cố định trong 13 giây, khi đến giữa giao lộ, robot xoay sang trái bằng cách giảm vận tốc tuyến tính về 0 và tăng vận tốc góc thành 0.3 trong 5 giây, sau đó tiến thẳng

về phía trước để ra khỏi giao lộ và tiếp tục thực hiện việc nhận diện làn đường để tiếp tục di chuyển.



Hình 5.17: Robot phát hiện biển báo và bắt đầu quá trình rẽ phải



Hình 5.18: Robot đang rẽ phải

Khi phát hiện biển báo và thực hiện rẽ phải, robot bắt đầu di chuyển thẳng tới phía trước giao lộ với vận tốc tuyến tính cố định trong 10 giây, khi đến giữa giao lộ, robot xoay sang phải bằng cách giảm vận tốc tuyến tính về 0 và tăng vận tốc góc thành -0.3 trong 5 giây, sau đó tiến thẳng về phía trước để ra khỏi giao lộ và tiếp tục thực hiện việc nhận diện làn đường để tiếp tục di chuyển.

Khi phát hiện biển báo và thực hiện đi thẳng, robot chỉ cần di chuyển thẳng về phía trước với vận tốc tuyến tính cố định trong 15 giây để qua khỏi giao lộ, sau đó tiếp tục thực hiện việc nhận diện làn đường để tiếp tục di chuyển.

Khi phát hiện biển báo dừng lại, robot sẽ đến đủ gần với biển báo và giảm cả vận tốc tuyến tính và vận tốc góc về 0 để đứng yên tại chỗ.

5.4 Một số thông số khi thực nghiệm tích hợp hệ thống

Module nhận diện làn đường:

- Kích thước hình ảnh từ camera robot: 240x320 pixel
- Số lượng khung cửa sổ trượt (sliding window): 10

Module phát hiện và phân loại biển báo:

- Độ tin cậy tối thiểu model phát hiện biển báo: 0.5
- Tốc độ khung hình trung bình: 3 khung hình mỗi giây
- Tốc độ xử lý trung bình: 0.372s
- Diện tích bounding box để robot thực hiện quá trình rẽ: 1000 đơn vị diện tích

Module điều khiển robot:

- Vận tốc tuyến tính cố định: 0.08 m/s
- Vận tốc góc tối đa: 0.5 rad/s
- Vận tốc góc tối thiểu: -0.5 rad/s
- PID: $K_p = 0.001$, $K_i=0.0001$, $K_d = 0.0025$

Chương 6

Tổng kết đề tài

6.1 Kết quả đạt được và hạn chế

6.1.1 Kết quả đạt được

Sau thời gian nỗ lực tìm hiểu và thực hiện, em đã đạt được một số kết quả chính sau đây:

- Tổng hợp được một số lý thuyết tổng quát về lĩnh vực phát hiện làn đường và nhận diện biển báo giao thông.
- Khảo sát được một số công trình nghiên cứu liên quan.
- Đưa ra được mô hình đề xuất trên cơ sở các kiến thức đã tìm hiểu được.
- Hiện thực được thuật toán phát hiện làn đường dùng camera và xử lý ảnh
- Hiện thực được thuật toán nhận diện biển báo giao thông dùng camera và machine learning.
- Hiện thực được module điều khiển robot với các thông tin được cung cấp từ các module khác, sử dụng bộ điều khiển PID để tính toán vận tốc cho robot.
- Tích hợp các thành phần của hệ thống thành hệ thống hoàn chỉnh, thử nghiệm trên môi trường mô phỏng.

6.1.2 Hạn chế

- Chưa đánh giá được các yếu tố ràng buộc về phần cứng để triển khai mô hình hệ thống đề xuất lên làn đường thực tế.
- Chưa tìm hiểu sâu và đưa ra được các tham số để tối ưu các mô hình học máy khi thực hiện huấn luyện giúp nâng cao độ chính xác.
- Một số thành phần của hệ thống còn phụ thuộc vào việc thử nghiệm để đưa ra thông số thay vì có phương pháp khoa học để giải quyết vấn đề.

-
- Quy mô đề tài còn hạn chế, đã giới hạn lại nhiều đặc điểm của bài toán để đảm bảo khả năng thực thi của hệ thống nên chưa hoàn toàn biểu diễn và giải quyết được các vấn đề có trong thực tế.

6.2 Thuận lợi và khó khăn khi thực hiện đề tài

6.2.1 Thuận lợi

- Em luôn nhận được sự hướng dẫn, giúp đỡ tận tình của giảng viên hướng dẫn, cùng với sự động viên, chia sẻ của người thân, bạn bè.
- Nguồn tài liệu phong phú với các kiến thức được cập nhật liên tục. Có nhiều phương pháp giải quyết vấn đề đã được nghiên cứu chi tiết, cho phép em được lựa chọn phương pháp phù hợp với khả năng hiện thực và quy mô của đề tài.
- Tính khả thi của đề tài: Đây là một đề tài với các vấn đề thiết thực, hữu ích trong thực tế, có sự hấp dẫn khi thực hiện nghiên cứu và hiện thực hóa hệ thống.

6.2.2 Khó khăn

- Đề tài liên quan đến nhiều lĩnh vực rộng, đòi hỏi kiến thức khó và chuyên sâu, cá nhân em chưa có kinh nghiệm về các lĩnh vực này.
- Đề tài được thực hiện bởi cá nhân nên không có sự hợp tác làm việc nhóm giúp nhanh chóng giải quyết khi có khó khăn xuất hiện và không tránh khỏi những thiếu sót trong quá trình nghiên cứu và hiện thực đề tài.

6.3 Hướng phát triển đề tài

Em đề xuất những gợi ý sau nhằm hoàn thiện, mở rộng đề tài:

- Tiến hành kiểm tra, đánh giá kỹ lưỡng để chọn ra được các giá trị thích hợp cho các tham số trong thuật toán .. Điều này nhằm giúp cho hệ thống có hiệu năng đủ tốt trong nhiều điều kiện hoạt động khác nhau. Thử nghiệm hệ thống với các thông số khác để tối ưu hóa hoạt động của hệ thống.
- Nghiên cứu, hiện thực và kiểm tra hiệu năng nhiều thuật toán phát hiện làn đường và nhận diện biển báo giao thông khác nhau để nâng cao khả năng hiện thực của hệ thống. Cần ưu tiên quan tâm đến độ chính xác trước so với thời gian xử lý.
- Nâng cấp module phát hiện làn đường với các phương pháp học sâu đã chứng tỏ sự thành công đáng kể trong các nhiệm vụ phát hiện làn đường bằng cách tận dụng sức mạnh của mạng lưới thần kinh để tự động tìm hiểu và trích xuất các đặc điểm của làn đường từ hình ảnh hoặc khung hình video.

Các CNN có thể được đào tạo bằng cách sử dụng bộ dữ liệu được gắn nhãn có hình ảnh và chú thích làn đường tương ứng. Mạng học cách nhận biết các mẫu, cạnh và kết cấu làn đường, cho phép chúng phát hiện và phân chia làn đường một cách hiệu quả trong thời gian thực. Sau đó, các mô hình đã được đào tạo có thể được triển khai trên nền tảng nhúng hoặc tích hợp vào các phương tiện tự lái để cung cấp khả năng phát hiện làn đường chính xác và mạnh mẽ.

Phương pháp học sâu mang lại một số lợi thế trong việc phát hiện làn đường. Chúng có thể xử lý các hình dạng làn đường phức tạp, các điều kiện ánh sáng khác nhau và các tình huống đường đầy thử thách. Ngoài ra, các mô hình deep learning có thể học các biểu diễn làn đường phức tạp, bao gồm các làn đường cong, đường đứt nét và nhiều làn đường. Tuy nhiên, các phương pháp này yêu cầu dữ liệu đào tạo được dán nhãn đáng kể và tài nguyên tính toán để đào tạo và suy luận, khiến chúng có cường độ tính toán cao hơn so với các kỹ thuật thị giác máy tính truyền thống.

- Tìm hiểu và ứng dụng Adaptive Thresholding thay cho phương pháp thresholding đơn giản. Do sự thay đổi trong điều kiện ánh sáng, bóng đổ,..., có thể một giá trị phân ngưỡng sẽ hoạt động đối với một phần nhất định của hình ảnh đầu vào nhưng sẽ hoàn toàn không hoạt động trên một phần khác của hình ảnh. Adaptive thresholding xem xét từng khu vực nhỏ trên hình ảnh và tìm giá trị phân ngưỡng tối ưu cho từng khu vực thay vì áp dụng một giá trị phân ngưỡng cho toàn bộ hình ảnh.[11]
- Mở rộng quy mô đề tài với nhiều vấn đề hơn khi tham gia giao thông cho xe tự hành như thực hiện né vật cản, bổ sung thêm các biển báo vào hệ thống nhận diện, áp dụng các quy luật giao thông phức tạp hơn,
- Xem xét tới việc phát triển hệ thống trên các phần cứng, robot khác.

Tài liệu tham khảo

- [1] Automatic Addison. Difference between histogram equalization and histogram matching. In <https://automaticaddison.com/difference-between-histogram-equalization-and-histogram-matching/>.
- [2] Richeng Cheng. A survey: Comparison between convolutional neural network and yolo in image identification. volume 1453, page 012139. IOP Publishing, jan 2020.
- [3] Vighnesh Devane, Ganesh Sahane, Hritish Khairmode, and Gaurav Datkhile. Lane detection techniques using image processing. In *ITM Web of Conferences*, volume 40, pages 1–4, Jan. 2021.
- [4] Ultralytics YOLOv8 Docs.
- [5] Jekyll and Minimal Mistakes. Robotic e-manual. In *computer-vision.zone*, 2020.
- [6] Miguel Maestre. Traffic sign recognition for autonomous cars. In <https://github.com/MichiMaestre/Traffic-Sign-Recognition-for-Autonomous-Cars>.
- [7] M.Hassan. Traffic signs classification using convolution neural networks cnn. In *computer-vision.zone*, February 2020.
- [8] Sandipann P. Narote. A review of recent advances in lane detection and departure warning system. 2018.
- [9] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.
- [10] ROBOTIS. Turtlebot3. In <https://emanual.robotis.com/docs/en/platform/turtlebot3/>.
- [11] Adrian Rosebrock. Adaptive thresholding with opencv (cv2.adaptiveThreshold). In <https://pyimagesearch.com/2021/05/12/adaptive-thresholding-with-opencv-cv2-adaptivethreshold/>, 2021.
- [12] Adrian Rosebrock. Opencv thresholding (cv2.threshold). In <https://pyimagesearch.com/2021/04/28/opencv-thresholding-cv2-threshold/>, 2021.
- [13] Wikipedia. In <https://en.wikipedia.org/wiki/Curvefitting>.