ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

CHUYÊN NGÀNH: AN TOÀN THÔNG TIN

ĐỀ TÀI:

THIẾT KẾ XE TỰ HÀNH ĐA HƯỚNG ỨNG DỤNG CÔNG NGHỆ AI

Sinh viên thực hiện:

NGUYỄN TRUNG ĐỨC **MSSV: 102200206**

PHẠM VĂN TRỌNG **MSSV: 102200237**

Lớp**:** **20TCLC-DT5**

Người hướng dẫn: **PGS. TS. NGUYỄN TẤN KHÔI**

THS. LÊ VĂN KHANH

Đà Nẵng, 6/2024

NGUYỄN TRUNG ĐỨC, PHẠM VĂN TRỌNG THIẾT KẾ XE TỰ HÀNH ĐA HƯỚNG ỨNG DỤNG CÔNG NGHỆ AI 2024

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

CHUYÊN NGÀNH: AN TOÀN THÔNG TIN

ĐỀ TÀI:

THIẾT KẾ XE TỰ HÀNH ĐA HƯỚNG ỨNG DỤNG CÔNG NGHỆ AI

Sinh viên thực hiện:

NGUYỄN TRUNG ĐỨC **MSSV: 102200206**

PHẠM VĂN TRỌNG **MSSV: 102200237**

Lớp**:** **20TCLC-DT5**

Người hướng dẫn: **PGS. TS. NGUYỄN TẤN KHÔI**

THS. LÊ VĂN KHANH

Đà Nẵng, 6/2024

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHẬN XÉT ĐỒ ÁN TỐT NGHIỆP

1. Thông tin chung:

1. Họ và tên sinh viên: Nguyễn Trung Đức MSSV: 102200206
   Họ và tên sinh viên: Phạm Văn Trọng MSSV: 102200237
2. Lớp: 20TCLC-DT5
3. Tên đề tài: Thiết Kế Xe Tự Hành Đa Hướng Ứng Dụng Công Nghệ AI
4. Người hướng dẫn: PGS. TS. Nguyễn Tấn Khôi Học hàm/ học vị:

II. Nhận xét, đánh giá đồ án tốt nghiệp:

1. Về tính cấp thiết, tính mới, khả năng ứng dụng của đề tài: (điểm tối đa là 2đ)

……..

……..

2. Về kết quả giải quyết các nội dung nhiệm vụ yêu cầu của đồ án: (điểm tối đa là 4đ)

……..

……..

3. Về hình thức, cấu trúc, bố cục của đồ án tốt nghiệp: (điểm tối đa là 2đ)

……..

……..

4. Đề tài có giá trị khoa học/ có bài báo/ giải quyết vấn đề đặt ra của doanh nghiệp hoặc nhà trường: (điểm tối đa là 1đ)

……..

……..

5. Các tồn tại, thiếu sót cần bổ sung, chỉnh sửa:

……..

……..

III. Tinh thần, thái độ làm việc của sinh viên: (điểm tối đa 1đ)

……..

IV. Đánh giá:

1. Điểm đánh giá: ……../10 (lấy đến 1 số lẻ thập phân)
2. Đề nghị: ☐ Được bảo vệ đồ án ☐ Bổ sung để bảo vệ ☐ Không được bảo vệ

Đà Nẵng, ngày tháng năm 2024

TÓM TẮT

Tên đề tài: THIẾT KẾ XE TỰ HÀNH ĐA HƯỚNG ỨNG DỤNG CÔNG NGHỆ AI
Sinh viên thực hiện:
NGUYỄN TRUNG ĐỨC MSSV: 102200206
PHẠM VĂN TRỌNG MSSV: 102200237
Lớp: 20TCLC-DT5

Đề tài "Thiết kế xe tự hành đa hướng ứng dụng công nghệ AI"với mong muốn phát triển một hệ thống xe tự lái thông minh. Sử dụng AI (Artificial intelligence), IoT (Internet of Things), đề tài nhằm xây dựng một phần mềm điều khiển xe tự động có khả năng nhận diện môi trường, phân tích thông tin và đưa ra quyết định để điều khiển xe an toàn và hiệu quả.

Đề tài sử dụng các cảm biến siêu âm và camera để thu thập thông tin về môi trường xung quanh. Dữ liệu thu thập được sau đó được xử lý bởi các thuật toán học máy và mô hình học sâu để nhận dạng vật cản, đường đi và các yếu tố khác trên đường.

Để đào tạo và đánh giá hệ thống, chúng tôi sử dụng một tập dữ liệu lớn chứa thông tin về các tình huống giao thông thực tế. Từ đó, chúng tôi xây dựng các mô hình nhận diện kết hợp thuật toán để đưa ra quyết định điều khiển xe dựa trên dữ liệu từ các cảm biến và đặc trưng của môi trường.

Mục tiêu của đồ án là tạo ra một hệ thống xe có tính năng tự hành ổn định và đáng tin cậy. Chúng tôi tiếp tục nghiên cứu và phát triển các thuật toán cải tiến để tăng cường khả năng phân loại và dự đoán của hệ thống. Chúng tôi cũng thực hiện các thử nghiệm trên môi trường thực tế để đảm bảo rằng hệ thống hoạt động hiệu quả và đáp ứng được các yêu cầu an toàn giao thông.

Đồ án của tôi hy vọng có thể đóng góp vào sự phát triển của công nghệ xe tự hành. Bằng cách tăng cường an toàn giao thông, giảm tải sức lao động cho người lái và tạo ra một môi trường di chuyển thông minh và bền vững.

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: NGUYỄN TRUNG ĐỨC Số thẻ sinh viên: 102200206

Họ tên sinh viên: PHẠM VĂN TRỌNG Số thẻ sinh viên: 102200237

Lớp: 20TCLC-DT5 Khoa: Công Nghệ Thông Tin

Ngành: Công Nghệ Thông Tin Đặc Thù- Hợp tác Doanh Nghiệp

1. *Tên đề tài đồ án:*

   THIẾT KẾ XE TỰ HÀNH ĐA HƯỚNG ỨNG DỤNG CÔNG NGHỆ AI

2. *Đề tài thuộc diện: ☐ Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*
3. *Các số liệu và dữ liệu ban đầu: Không có*
4. *Nội dung các phần thuyết minh và tính toán:*

   Mở đầu: Giới thiệu mục đích, mục tiêu đề tài, phạm vi, đối tượng nghiên cứu và bố cục đồ án
   Chương 1: Cơ sở lý thuyết về xe tự hành
   Chương 2: Phân tích và thiết kế hệ thống
   Chương 3: Triển khai và kết quả thực nghiệm
   Chương 4: Kết luận
5. *Họ tên người hướng dẫn: PGS. TS. Nguyễn Tấn Khôi*
6. *Ngày giao nhiệm vụ đồ án: 1/4/2024*
7. *Ngày hoàn thành đồ án: 20/6/2024*

Đà Nẵng, ngày tháng năm 2024

Trưởng Bộ môn: PGS.TS. Nguyễn Tấn Khôi **Người hướng dẫn**

LỜI NÓI ĐẦU

Trong thời đại công nghệ thông tin ngày nay, xe tự hành thông minh là một trong những ứng dụng nổi bật của trí tuệ nhân tạo và công nghệ ô tô. Được thiết kế để tự động hóa quá trình lái xe, xe tự hành thông minh đang trở thành xu hướng phát triển tiên tiến trong ngành công nghiệp ô tô.

Trong đề tài này, chúng tôi tập trung vào việc nghiên cứu và phát triển các công nghệ để xe tự hành có khả năng nhận biết và tương tác với môi trường xung quanh một cách tự động. Đặc biệt, chúng tôi quan tâm đến khả năng xe tự nhận biết làn đường và các vật cản trong quá trình di chuyển trên đường.

Xe tự hành thông minh là đứa con tinh thần và toàn bộ tâm sức chúng tôi đã dành trọn vào nhằm tạo ra một sản phẩm có lợi ích cả về mặt kinh tế và về mặt lợi ích xã hội. Đây là một đề tài mà chúng tôi đặc biệt quan tâm và đam mê và chúng tôi rất vui được chia sẻ với quý vị thầy cô về những khám phá và kết quả đáng chú ý mà mình đã đạt được.

Để có được một đồ án chỉnh chu và đạt được kết quả tốt đẹp, chúng tôi thực sự biết ơn sự giúp đỡ, hỗ trợ rất nhiều từ thầy hướng dẫn của mình - thầy PGS. TS. Nguyễn Tấn Khôi cùng với sự giúp đỡ của Ths. Lê Văn Khanh. Chúng tôi xin được gửi lời cảm ơn trân trọng và trân quý nhất đến thầy vì đã tạo điều kiện giúp đỡ trong quá trình nghiên cứu và phát triển dự án.

Với điều kiện về thời gian cũng như lượng kiến thức về đề tài rất rộng mà kinh nghiệm còn hạn chế khi còn là một sinh viên, đồ án này không thể tránh được những thiếu sót. Chúng tôi rất mong nhận được sự chỉ bảo, đóng góp ý kiến của các thầy cô để có điều kiện bổ sung, nâng cao kiến thức của chính mình, phục vụ tốt hơn công tác thực tế sau này.

Một lần nữa chân thành cảm ơn quý thầy cô!

CAM ĐOAN

Chúng tôi cam đoan rằng trong quá trình phát triển dự án xe tự hành, chúng tôi không sử dụng hoặc ăn cắp bất kỳ tài liệu nào mà không được phép. Chúng tôi thực hiện đề tài này dưới sự hướng dẫn trực tiếp của PGS.TS Nguyễn Tấn Khôi. Chúng tôi cam kết tuân thủ các quy định và luật pháp liên quan đến bản quyền và sở hữu trí tuệ.

Tất cả các công trình nghiên cứu, tài liệu tham khảo và nguồn thông tin khác được sử dụng trong đồ án đều được trích dẫn và tham khảo đúng quy định. Chúng tôi đảm bảo tính trung thực và đạo đức trong việc tiếp cận, sử dụng và trình bày thông tin từ các nguồn tài liệu có liên quan.

Chúng tôi cam kết tuân thủ và đảm bảo tính trung thực và đạo đức trong mọi hoạt động nghiên cứu và phát triển, và sẵn sàng chịu trách nhiệm về bất kỳ vi phạm nào liên quan đến sở hữu trí tuệ và tài liệu tham khảo.

Sinh viên thực hiện

MỤC LỤC

DANH SÁCH CÁC BẢNG, HÌNH VẼ

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

Từ	Viết tắt của	Diễn giải
HTTP	Hypertext Transfer Protocol	Giao thức Truyền tải Siêu Văn Bản
GPU	Graphics Processing Unit	Đơn vị xử lý đồ họa
CPU	Central Processing Unit	Bộ xử lý trung tâm
AI	artificial intelligence	Trí tuệ nhân tạo
IoT	Internet of Thing	Internet kết nối vạn vật
SSH	Secure Shell	Giao thức hỗ trợ người dùng quản lý, điều chỉnh Server từ xa
I2C	Inter – Integrated Circuit	Giao thức giao tiếp nối tiếp đồng bộ
SPI	Serial Peripheral Interface	Giao diện ngoại vi nối tiếp
UART	Universal Asynchronous Receiver-Transmitter	Bộ truyền nhận dữ liệu nối tiếp bất đồng bộ
GPS	Global Positioning System	Hệ thống định vị toàn cầu
WPA/ WPA2	Wi-Fi Protected Access	Giao thức an ninh trên những mạng không dây
LAN	Local Area Network	Mạng máy tính nội bộ
WAN	Wide Area Network	Mạng diện rộng
RAM	Random Access Memory	Bộ nhớ tạm thời của máy tính

CMOS	Complementary Metal-Oxide-Semiconductor	Một loại công nghệ dùng để chế tạo mạch tích hợp.
RTSP	Real time streaming protocol	Giao thức truyền tin thời gian thực
SD	Secure Digital	Định dạng thẻ nhớ bộ nhớ không bay hơi

MỞ ĐẦU

1. Mục đích thực hiện đề tài

Mục đích chính của đề tài xe tự hành là nghiên cứu và phát triển một hệ thống xe tự động có khả năng di chuyển đa hướng trong môi trường bất kỳ mà không cần sự can thiệp của người lái. Đề tài nhằm tạo ra một hệ thống thông minh và an toàn, có khả năng nhận diện môi trường và đưa ra quyết định điều khiển phù hợp để xe tự động hoạt động.

- Đề tài có mục tiêu nghiên cứu các công nghệ và phương pháp phát triển hệ thống xe tự động.
- Phân tích và xây dựng các thuật toán và mô hình để xe tự động nhận diện môi trường và đưa ra quyết định điều khiển.
- Phát triển và triển khai phần cứng và phần mềm cho hệ thống xe tự động.
- Kiểm tra và đánh giá hiệu suất và độ tin cậy của hệ thống xe tự động.

2. Phạm vi và đối tượng nghiên cứu

- Phạm vi đề tài tập trung vào phát triển một hệ thống xe tự động di chuyển được đa hướng.
- Đối tượng nghiên cứu là các công nghệ và phương pháp liên quan đến điều khiển xe tự động, bao gồm nhận diện môi trường, xử lý hình ảnh, trí tuệ nhân tạo và hệ thống nhúng.

3. Phương pháp nghiên cứu

Nghiên cứu đề tài xe tự hành thường sử dụng phương pháp nghiên cứu kỹ thuật và thí nghiệm. Phương pháp nghiên cứu bao gồm việc tìm hiểu các công nghệ và phương pháp hiện có, thiết kế và xây dựng hệ thống, triển khai và đánh giá hiệu suất và độ tin cậy của hệ thống.

4. Cấu trúc của đồ án

Mở đầu, phân tích và thiết kế hệ thống, triển khai và kiểm thử, đánh giá kết quả và kết luận.

- Mở đầu giới thiệu vấn đề, mục tiêu, phạm vi và đối tượng nghiên cứu của đề tài.
- Phần phân tích và thiết kế hệ thống mô tả kiến trúc tổng quan của hệ thống, bao gồm phần cứng và phần mềm.
- Phần triển khai và kiểm thử trình bày quá trình xây dựng và triển khai hệ thống, kèm theo các bước kiểm thử và kết quả.

- Đánh giá kết quả và kết luận tổng kết lại các thành quả đạt được trong đề tài.
- Ngoài ra, đồ án tốt nghiệp còn có thể bao gồm các phần như tài liệu tham khảo, phụ lục (nếu có), và được trình bày theo định dạng và yêu cầu của trường/đơn vị đào tạo.

Chương 1: CƠ SỞ LÝ THUYẾT VỀ XE TỰ HÀNH

Hiện nay hệ thống xe tự động đang dần phát triển mạnh mẽ và trở thành mục tiêu quan trọng trong ngành công nghệ thông tin và công nghệ ô tô, với mong muốn đem lại 1 hệ thống lái xe an toàn và giảm thiểu sức lao động của con người chúng tôi đã tìm hiểu và phát triển thành công đồ án về xe tự hành di chuyển đa hướng. Đồ án đã giải quyết được vấn đề tự động nhận diện các đối tượng đưa ra các quyết định di chuyển an toàn và có khả năng di chuyển đa hướng, phục vụ trong các tình huống môi trường thực tế di chuyển khó khăn mà những ô tô hiện nay chưa làm được…

1.1. Tổng quan về đề tài

1.1.1. Các vấn đề cần giải quyết

- Cần thiết kế mô hình phần cứng xe tự hành
- Thu thập dữ liệu và xử lý dữ liệu
- Phát hiện và nhận diện đối tượng, làn đường
- Đưa ra giải pháp quyết định và đường đi trong lúc di chuyển.

1.2. Cơ sở lý thuyết

1.2.1. IOT(Internet of thing)

Internet of Things (IoT) là một khái niệm mô tả mạng lưới các đối tượng vật lý được kết nối với nhau và với Internet, cho phép chúng trao đổi dữ liệu và tương tác với nhau thông qua các giao thức mạng. Đối tượng vật lý trong hệ thống IoT có thể là các thiết bị điện tử, cảm biến, máy móc, phương tiện giao thông, các đối tượng thông minh trong gia đình (smart home), và nhiều hơn nữa.

Các thiết bị IoT thường được trang bị các cảm biến và chức năng kết nối mạng để thu thập và chuyển dữ liệu. Dữ liệu này có thể là thông tin về môi trường xung quanh, tình trạng hoạt động của thiết bị, thông tin vị trí, hoặc bất kỳ thông tin nào có liên quan. Dữ liệu này sau đó được chuyển đến các hệ thống phân tích và xử lý để tạo ra thông tin hữu ích, từ đó hỗ trợ việc đưa ra quyết định thông minh và cải thiện hiệu quả hoạt động.

Hình 1.1: Tổng quan về IoT

1.2.2. Ngôn ngữ lập trình Python

Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ

Hình 1.2: Logo Python

Ngôn ngữ lập trình Python có rất nhiều thư viện và framework lớn thuận lợi cho việc viết code và phát triển khoa Học máy tính. Python vốn là ngôn ngữ nổi tiếng về sự đơn giản không cầu kỳ, code dễ học, dễ đọc, cú pháp logic và ngắn gọn, còn Học máy (Machine Learning) liên quan đến các thuật toán cực kỳ phức tạp và quy trình làm việc nhiều giai đoạn nên ở đây, sự logic ngắn gọn và dễ dàng của Python đóng vai trò quan trọng trong việc tiết kiệm thời gian của các nhà phát triển.

Mặt khác, khi nói đến Khoa học dữ liệu (Data Science), Python cũng có các gói đặc biệt dành cho các công việc lĩnh vực này như SciPy, NumPy hay Pandas tạo điều kiện cho việc phân tích dữ liệu và có thể dễ dàng tích hợp với các ứng dụng web.

Tôi sử dụng ngôn ngữ lập trình Python để cài đặt Server với xây dựng mô hình để nhận diện môi trường và điều khiển xe.

1.2.3. Raspberry OS

Raspberry Pi OS (trước đây được gọi là Raspbian) là một hệ điều hành mã nguồn mở được phát triển dành riêng cho Raspberry Pi - một dòng máy tính nhúng giá rẻ và phổ biến. Raspberry Pi OS được xây dựng dựa trên hệ điều hành Linux Debian và tối ưu hóa để hoạt động tốt trên các bo mạch Raspberry Pi.

Raspberry Pi OS nhằm cung cấp một môi trường hoạt động ổn định và dễ sử dụng trên Raspberry Pi, để tạo điều kiện cho việc phát triển và triển khai các ứng dụng và dự án nhúng. Raspberry Pi OS sử dụng công cụ quản lý gói APT (Advanced Package Tool) để cài đặt và quản lý phần mềm. Nó cũng hỗ trợ nhiều ngôn ngữ lập trình như Python, C/C++, Java và nhiều ngôn ngữ khác, giúp người dùng phát triển ứng dụng và dự án trên Raspberry Pi.

Hình 1.3: Raspberry Pi

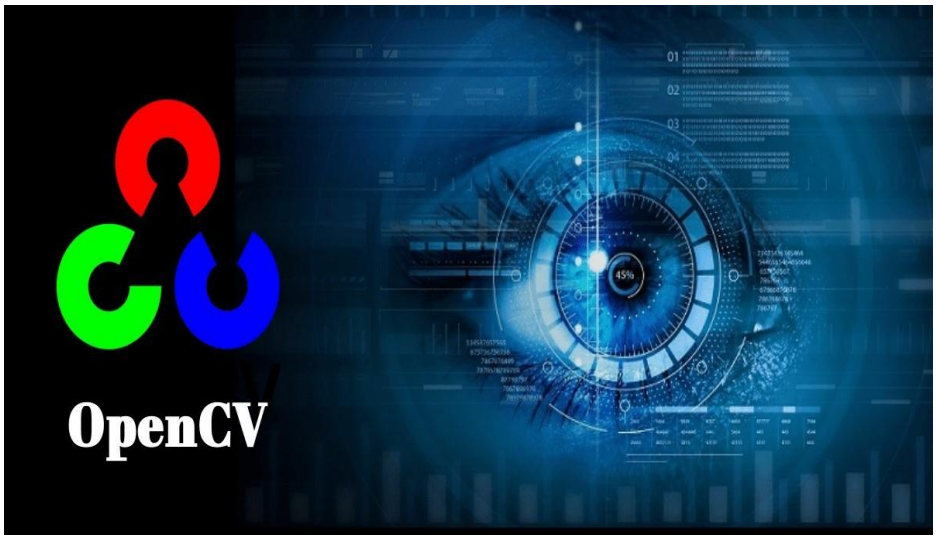
1.2.4. Thư viện hỗ trợ xử lý

1.2.4.1. OpenCV

OpenCV (Open Source Computer Vision) là một thư viện mã nguồn mở và miễn phí cho xử lý ảnh và thị giác máy tính. Nó được phát triển ban đầu bởi Intel và hiện được duy trì bởi các tác giả khác nhau trên toàn cầu. OpenCV cung cấp một loạt các thuật toán và công cụ để phân tích, xử lý và trích xuất thông tin từ ảnh và video.

Ưu điểm:

- Đa nền tảng: OpenCV hỗ trợ trên nhiều nền tảng khác nhau bao gồm Windows, Linux
- Xử lý ảnh và video: OpenCV cung cấp các hàm và công cụ mạnh mẽ cho xử lý ảnh và video, bao gồm chuyển đổi, lọc nhiễu, cân bằng histogram, biến đổi hình học, phát hiện cạnh, phát hiện đối tượng, và nhiều thao tác khác.
- Phân tích và nhận dạng: OpenCV hỗ trợ các công cụ cho phân tích và nhận dạng, bao gồm phát hiện khuôn mặt, nhận dạng đối tượng, theo dõi chuyển động, đọc mã vạch và mã QR, và nhiều công nghệ khác liên quan đến nhận dạng và phân loại.
- Machine learning: OpenCV tích hợp tính năng machine learning và deep learning thông qua việc hỗ trợ các thư viện như TensorFlow và PyTorch. Điều này cho phép ứng dụng các thuật toán học máy và mạng nơ-ron trong các ứng dụng thị giác máy tính.
- Tích hợp với ngôn ngữ lập trình: OpenCV hỗ trợ nhiều ngôn ngữ lập trình như C++, Python, Java, và MATLAB



Hình 1.4: Open CV

1.2.4.2. PyTorch

Pytorch là một framework mã nguồn mở hỗ trợ cho Deep Learning được phát triển bởi Facebook. Cùng với Tensorflow và Keras, đây là một trong những framework phổ biến nhất được sử dụng trong các bài toán về trí tuệ nhân tạo.

Trong nghiên cứu các ứng dụng trí tuệ nhân tạo, Pytorch thường được ưu tiên sử dụng vì có thể giúp triển khai các bài toán một cách dễ dàng. Pytorch hỗ trợ đắc lực cho

các nhà phát triển thực hiện Debug (gỡ lỗi) và Data Visualize (trực quan hóa dữ liệu) với cơ chế Dynamic Computation Graph.

a) Những lợi ích khi sử dụng PyTorch:

- Mã nguồn mở giúp PyTorch xây dựng một cộng đồng lớn mạnh với nguồn tài nguyên chất lượng.
- Khả năng xử lý đồ họa mạnh mẽ giúp kiểm soát CPU & GPU rõ ràng.
- Tập hợp nhiều Pythonic tự nhiên.
- Dễ dàng xử lý code khi gặp bug.
- Có TouchScript giúp triển khai các ứng dụng vào quy mô sản xuất để mở rộng quy mô.
- Các hàm, cú pháp cơ bản trong Pytorch giúp xử lý các bài toán về AI một cách nhanh chóng.

b) Các đặc điểm cơ bản của PyTorch

- Tensor: Tensor có thể là một vectơ hoặc ma trận đa chiều và đại diện cho các loại dữ liệu. Tất cả các giá trị trong một tensor sẽ có kiểu dữ liệu giống hệt nhau. Hình dạng của dữ liệu cũng là kích thước các mảng hoặc ma trận.
- Tensor cũng có thể được xử lý bởi CPU hoặc GPU để giúp hoạt động nhanh hơn. Có nhiều loại Tensor khác nhau như Float Tensor, Double Tensor, Half Tensor, Int Tensor và Long Tensor. PyTorch sử dụng Float Tensor 32-bit làm mặc định.
- Các hoạt động toán học: Code để thực hiện các phép toán trong PyTorch cũng giống như trong Numpy (một thư viện toán học phổ biến của Python). Người dùng cần khởi tạo 2 Tensor và thực hiện các phép toán như cộng, trừ, nhân và chia với chúng.
- Dynamic Computation Graph: Các đồ thị tính toán trong PyTorch cho phép framework tính toán các giá trị gradient cho các Neural Network được xây dựng. PyTorch sử dụng Dynamic Computation Graph cho phép người dùng có thể xây dựng xen kẽ và định giá đồ thị. Ngoài ra, dạng đồ thị này còn thân thiện với Debug vì cho phép thực thi code từng dòng.
- Tóm lại, Dynamic Computation Graph là một tính năng quan trọng khiến PyTorch trở thành lựa chọn ưu tiên trong ngành.

- Dataset và DataLoader: Làm việc với bộ dữ liệu lớn yêu cầu tải tất cả dữ liệu vào bộ nhớ trong một lần duy nhất để tiết kiệm thời gian. Điều này gây ra tình trạng đầy bộ nhớ và các chương trình chạy chậm.
- PyTorch cung cấp hai dữ liệu ban đầu là DataLoader và Dataset cho phép người dùng sử dụng dữ liệu của riêng họ cũng như các tập dữ liệu được tải trước.
- Khởi tạo ma trận: Để khởi tạo ma trận với các số ngẫu nhiên trong PyTorch, ta sẽ sử dụng hàm randn () và cung cấp một tensor chứa đầy các số ngẫu nhiên. Các phép toán ma trận cơ bản và phép toán chuyển vị trong PyTorch cũng tương tự như NumPy.

c) Các module phổ biến trong PyTorch

- Autograd: Autograd là module phân biệt tự động của PyTorch. Module này tạo ra một đồ thị xoay chiều có hướng với một Tensor đầu vào và một tensor đầu ra.
- Optim: Optim là một package với các thuật toán được viết sẵn cho các trình tối ưu hóa, có thể được sử dụng để xây dựng Neural Network (Mạng nơ-ron nhân tạo trong AI)
- Nn: nn bao gồm các class khác nhau giúp xây dựng các mô hình Neural Network. Tất cả các module trong PyTorch đều là các subclass của nn.

Hình 1.5: Pytorch

1.2.4.3. Sklearn

Scikit-learn (Sklearn) là thư viện mạnh mẽ nhất dành cho các thuật toán học máy được viết trên ngôn ngữ Python. Thư viện cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering, và dimensionality reduction.

Thư viện được cấp phép bản quyền chuẩn FreeBSD và chạy được trên nhiều nền tảng Linux. Scikit-learn được sử dụng như một tài liệu để học tập.

Để cài đặt scikit-learn trước tiên phải cài thư viện SciPy (Scientific Python). Những thành phần gồm:

Numpy	Gói thư viện xử lý dãy số và ma trận nhiều chiều
SciPy	Gói các hàm tính toán logic khoa học
Matplotlib	Biểu diễn dữ liệu dưới dạng đồ thị 2 chiều, 3 chiều
Python	Notebook dùng để tương tác trực quan với Python
SymPy	Gói thư viện các kí tự toán học
Pandas	Phân tích dữ liệu dưới dạng bảng

Những thư viện mở rộng của SciPy thường được đặt tên dạng SciKits. Như thư viện này là gói các lớp, hàm sử dụng trong thuật toán học máy thì được đặt tên là scikit-learn.

Scikit-learn hỗ trợ mạnh mẽ trong việc xây dựng các sản phẩm. Nghĩa là thư viện này tập trung sâu trong việc xây dựng các yếu tố: dễ sử dụng, dễ code, dễ tham khảo, dễ làm việc, hiệu quả cao.

Mặc dù được viết cho Python nhưng thực ra các thư viện nền tảng của scikit-learn lại được viết dưới các thư viện của C để tăng hiệu suất làm việc. Ví dụ như: Numpy(Tính toán ma trận), LAPACK, LibSVM và Cython.

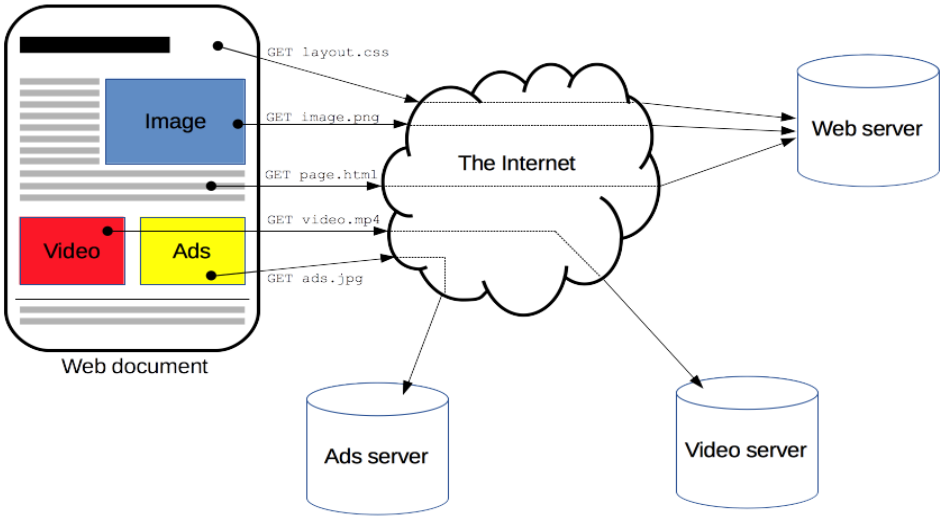
1.2.5. Giao thức HTTP

HTTP là viết tắt của HyperText Transfer Protocol, là một giao thức truyền tải siêu văn bản, giúp cho các máy tính có thể giao tiếp với nhau qua mạng. HTTP là nền tảng của World Wide Web kết nối giữa máy chủ (server) và máy khách (client) trong cùng một hệ thống mạng.

Với khả năng mở rộng đa dạng, HTTP không chỉ được sử dụng để tải các tài liệu siêu văn bản, mà còn để truyền tải hình ảnh, video và thậm chí để đăng tải nội dung lên

máy chủ, chẳng hạn như kết quả của các biểu mẫu HTML. Ngoài ra, HTTP cũng có thể sử dụng để tải lên các phần của trang web, giúp cập nhật nội dung theo yêu cầu.

HTTP được sử dụng rộng rãi trong việc truy cập các trang web trên Internet. Khi bạn nhập một địa chỉ web vào trình duyệt, trình duyệt sẽ gửi một yêu cầu HTTP đến máy chủ web, và máy chủ web sẽ trả về một phản hồi HTTP chứa mã HTML của trang web đó. Trình duyệt sẽ đọc mã HTML và hiển thị trang web cho bạn xem. Bạn có thể thấy các yêu cầu và phản hồi HTTP bằng cách sử dụng công cụ kiểm tra mạng (network inspector) của trình duyệt.



Hình 1.6: Giao thức HTTP

1.2.6. Giao thức RTSP

RTSP- Real Time Streaming Protocol hay còn gọi là giao thức truyền tin theo thời gian thực được sử dụng để truyền phát các dữ liệu truyền trực tiếp hoặc các luồng đa phương tiện(media stream) thông qua mạng. Giao thức này thường được sử dụng trong các ứng dụng truyền phát video và âm thanh trực tiếp trên Internet, video hội thảo trực tuyến và ứng dụng giám sát.

RTSP cho phép người dùng điều khiển việc truyền phát phương tiện bằng cách thực hiện các hành động như play (phát), pause (tạm dừng), stop (dừng), và seek (tìm kiếm) trên luồng phương tiện. Nó cung cấp một giao diện giữa máy chủ phát và máy khách truy cập, cho phép máy khách yêu cầu các phương tiện từ máy chủ, điều khiển phát và nhận luồng phương tiện.

Giao thức RTSP sử dụng cơ chế TCP/IP để truyền phát dữ liệu truyền trực tiếp và sử dụng các phương thức như OPTIONS, DESCRIBE, SETUP, PLAY và TEARDOWN

để thiết lập và điều khiển các luồng phương tiện. Nó cũng hỗ trợ các phương thức bổ sung như PAUSE và GET_PARAMETER.

RTSP có thể làm việc cùng với các giao thức phương tiện khác như RTSP (Real-time Transport Protocol) để truyền phát các luồng phương tiện trực tiếp.

1.2.7. Mit App Inventor

App Inventor là một ứng dụng web mã nguồn mở được cung cấp bởi Google từ tháng 7 năm 2010. Sau này, App Inventor được quản lý bởi Viện Công nghệ Massachusetts hay còn gọi là MIT hay còn gọi là Mit App Inventor.

App Inventor sẽ hoạt động dựa trên nền tảng di động Android. Tức là các thành phẩm được tạo ra từ App Inventor sẽ chỉ hoạt động được trên Android. Giao diện của App Inventor bao gồm các khối hộp, bên trong là các đoạn mã. Khi sử dụng, người dùng sẽ kéo thả các khối này vào bảng mã để tiến hành lắp ghép thành một ứng dụng hoàn chỉnh. Nhìn chung, cách sử dụng App Inventor rất đơn giản, tất cả chỉ xoay quanh thao tác kéo và thả.

Môi trường lập trình App Inventor có cách tiếp cận tương tự như Scratch, câu lệnh đều là những khối lệnh trực quan được kéo thả để tạo ra các ứng dụng từ đơn giản tới phức tạp. Các ứng dụng có thể chạy trực tiếp trên các thiết bị di động như điện thoại và máy tính bảng.

Hình 1.7: Mit App Inventor

Chương 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

Trong đồ án này, chúng tôi sẽ tiến hành phân tích và thiết kế hệ thống cho xe tự hành đa hướng ứng dụng công nghệ AI. Phân tích và thiết kế là các bước then chốt trong quá trình phát triển hệ thống, giúp đảm bảo rằng tất cả các yêu cầu kỹ thuật và chức năng đều được xác định rõ ràng và được đáp ứng một cách hiệu quả. Trước hết, chúng tôi sẽ tiến hành phân tích các yêu cầu của hệ thống, bao gồm cả yêu cầu từ phía người dùng và các yêu cầu kỹ thuật chi tiết. Tiếp theo, chúng tôi sẽ thiết kế hệ thống, từ cấu trúc phần cứng đến các thuật toán AI và phần mềm điều khiển, nhằm đảm bảo rằng hệ thống xe tự hành có thể hoạt động mượt mà, hiệu quả và an toàn. Mục tiêu của chương này là cung cấp một nền tảng vững chắc cho việc triển khai và phát triển hệ thống xe tự hành trong các chương tiếp theo.

2.1. Phân tích vấn đề

2.1.1. Nhận diện và nhận biết môi trường

Để xe có thể di chuyển trong môi trường thực tế, chúng ta cần xác định được môi trường xung quanh. Dữ liệu thu thập được trong môi trường hiện tại là dữ kiện giúp hệ thống đưa ra quyết định cho xe di chuyển.

a) Camera

- Được sử dụng để quan sát hình ảnh phía trước vật thể. Điều này cho phép hệ thống biết không gian phía trước của xe gồm những gì, có vật cản hay không.
- Camera được gắn ở vị trí phù hợp để camera có góc nhìn đủ rộng vị trí quan sát tốt nhất mà không ảnh hưởng tới kết cấu và sự ổn định của mô hình.
- Ngoài ra camera phải đảm bảo chất lượng hình ảnh, độ phân giải thích hợp, khả năng nhìn trong điều kiện ánh sáng yếu, tốc độ đường truyền nhanh để phù hợp với yêu cầu hoạt động trong thời gian thực.

b) Cảm biến siêu âm

Vì góc nhìn của camera bị giới hạn và luôn cố định 1 hướng về phía trước xe. Trong khi thực tế ta cần nhiều hơn thế. Tưởng tượng như khi ta lái xe thì mắt tập trung quan sát phía trước là chưa đủ. Ta cần đồng thời quan sát gương chiếu hậu để quan sát 2 bên và phía sau xe để tránh xe vượt lên.

Để giải quyết vấn đề này cảm biến siêu âm là một lựa chọn tốt bởi các tính năng:

- Đo khoảng cách từ xe đến các vật thể xung quanh.
- Cảm biến siêu âm phát ra sóng siêu âm và đo thời gian phản xạ để tính toán khoảng cách.
- Có thể sử dụng nhiều cảm biến siêu âm được đặt ở các vị trí chiến lược trên xe để đo khoảng cách từ nhiều hướng.

c) Kết hợp 2 công nghệ

Sử dụng kết hợp 2 công nghệ này giúp hệ thống xây dựng, mô hình hóa không gian xung quanh. Điều này yêu cầu hệ thống phải có 1 phương thức kết hợp 2 dữ liệu đầu vào này với nhau hiệu quả nhằm nâng cao hiệu suất của hệ thống.

2.1.2. Điều khiển xe

Sau khi đã biết được môi trường xung quanh, cần đi hướng nào thì bây giờ là ra quyết định điều khiển xe.

a) Kết hợp dữ liệu đầu vào

- Kết hợp các dữ liệu đầu vào thu được từ các cảm biến và camera, giá trị của 4 cảm biến siêu âm, hình ảnh từ camera làm dữ liệu đầu vào cho việc ra quyết định
- Đưa dữ liệu về định dạng chính xác mà thuật toán mong muốn, cần kiểm tra các điều kiện để bảo đảm dự tính toàn vẹn và đúng đắn của dữ liệu. Điều này giúp loại bỏ các ký tự đặc biệt, chuyển đổi đơn vị đo lường, định dạng cũng như chuyển đổi kiểu dữ liệu
- Trước khi áp dụng các thuật toán hoặc phân tích dữ liệu, ta cần thực hiện các bước tiền xử lý dữ liệu như xử lý dữ liệu thiếu, xử lý nhiễu, chuẩn hóa, chuẩn hoá hoặc giảm kích thước dữ liệu, tách dữ liệu thành các đặc trưng,...

b) Ra quyết định

Sau khi đã có dữ liệu đầu vào, dữ liệu đã được huấn luyện thì ta cần giải thuật để đưa ra quyết định. Giải thuật này phải đáp ứng được những yêu cầu sau:

- Giải thuật cần đáp ứng yêu cầu về thời gian phản hồi nhanh. Nó phải xử lý dữ liệu và đưa ra kết quả trong khoảng thời gian cụ thể, thường chỉ trong vài mili giây hoặc thậm chí nano giây. Tỉ lệ nhận dạng có tỉ lệ nhận dạng chính xác đủ lớn

- Giải thuật cần có thời gian thực hiện dự đoán hoặc xử lý dữ liệu rất nhanh. Điều này đòi hỏi sử dụng các thuật toán hiệu quả và tối ưu, tránh sự trễ và quá tải trong quá trình tính toán
- Giải thuật phải đảm bảo tính tin cậy trong việc xử lý dữ liệu. Nó cần xử lý dữ liệu một cách chính xác và đáng tin cậy, tránh lỗi và nhiễu.
- Nó phải đảm bảo sử dụng tài nguyên như bộ nhớ, CPU và băng thông mạng một cách tối ưu để đáp ứng yêu cầu về thời gian phản hồi.
- Cần xử lý đồng thời các tác vụ hoặc luồng dữ liệu. Nó phải đảm bảo khả năng xử lý đa luồng và xử lý dữ liệu đồng thời một cách hiệu quả

c) Ảnh hưởng từ môi trường thực tế

Khi xe chạy trong môi trường thực tế sẽ có rất nhiều yếu tố bên ngoài tác động vào khả năng hoạt động của xe có thể cả chủ quan và khách quan. Chủ quan như tính ổn định của xe, độ rung lắc của camera... Khách quan như không gian thực tế như độ nghiêng mặt đường, độ cao lề đường, tốc độ đường truyền,... cũng ảnh hưởng rất nhiều đến khả năng ra quyết định chính xác của giải thuật. Điều này đòi hỏi ta phải có những giải pháp để giảm thiểu tối đa những ảnh hưởng này. Như điều chỉnh dữ liệu, các trọng số, thiết kế phần cứng tốt nhất.

2.2. Kiến trúc phần cứng của xe

2.2.1. Yêu cầu hệ thống

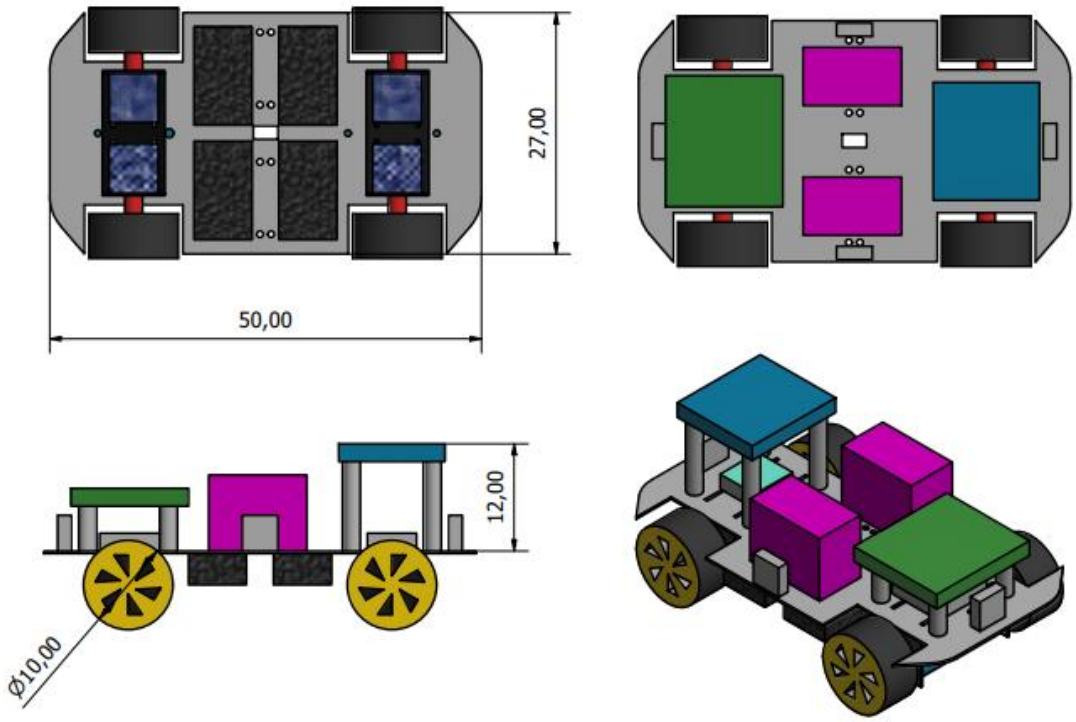
Xử lý trung tâm	Hệ thống cần 1 bộ xử lý trung tâm mạnh mẽ để tính toán dữ liệu hình ảnh kết hợp với các dữ liệu cảm biến, đưa ra kết quả một cách nhanh chóng và chính xác
Bộ nhớ	Hệ thống xe tự hành cần một không gian lưu trữ đủ lớn để có thể lưu trữ dữ liệu mô hình và tiếp tục thu thập thêm dữ liệu trong quá trình di chuyển, tạo 1 tập dữ liệu đa dạng giúp mô hình có thể học sâu và đưa ra các quyết định một cách chính xác

Giao tiếp	Giao tiếp giữa bộ xử lý trung tâm và động cơ cần phải liên kết và truyền kết quả 1 cách nhanh chóng, đáp ứng nhu cầu thực tế, tránh xảy ra trường hợp đứt gãy liên kết, mất tín hiệu khiến hệ thống xe rơi vào tình trạng mất điều khiển và xảy ra những sự cố ngoài ý muốn
Cảm biến	Hệ thống cảm biến của mô hình phải đưa ra được kết quả giống với thực tế nhất, giảm sai số trong tính toán và có được các quyết định 1 cách chính xác hơn
Thành phần điều khiển	Hệ thống xe cần thêm phần điều khiển từ app bên ngoài để có thể điều khiển hướng đi của xe giúp xử lý những trường hợp khẩn cấp như mất kết nối hay hệ thống AI đưa ra kết quả không chính xác do môi trường chưa thuận lợi.
Nguồn điện	Nguồn điện phải đáp ứng được cho động cơ trong khoảng thời gian dài để có thể di chuyển xa và đảm bảo tính ổn định, cũng như cung cấp nguồn điện cho các thiết bị có thể hoạt động đúng công suất
Thiết kế vật lý	Mô hình cần phải thiết kế cân đối, phù hợp cho việc di chuyển và vị trí lắp các linh kiện trên xe để có thể thu thập dữ liệu cho mô hình AI một cách chính xác.

2.2.2. Thiết kế hệ thống

a) Thiết kế vật lý

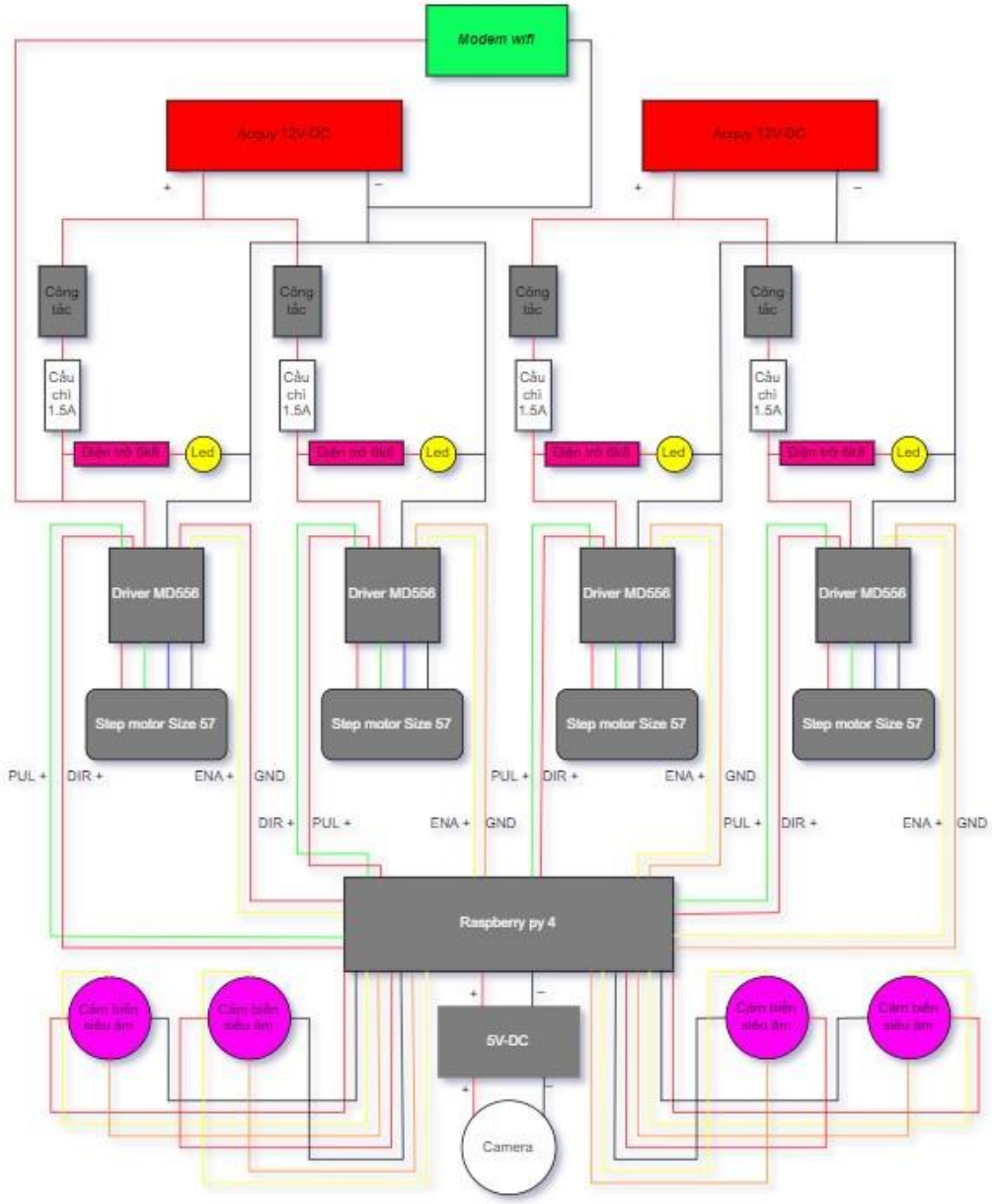
Mô hình hệ thống được thiết kế qua bản vẽ 3D và triển khai hệ thống mạch điện bằng draw.io, qua quá trình cập nhật đổi mới thiết kế để đưa ra mô hình phù hợp nhất, đáp ứng được các điều kiện yêu cầu đặt ra cho mô hình. Dưới đây là sơ đồ thiết kế khung xe:



Hình 2.1: Tổng thể thiết kế mô hình 3D cho xe tự hành

b) Thiết kế hệ thống mạch điện

Ở đây chúng tôi sử dụng Raspberry pi 4 Model B với 40 chân GPIO cho phép mô hình tích hợp với các thành phần khác như động cơ bước, cảm biến siêu âm và các bộ điều khiển động cơ. Dưới đây là sơ đồ tổng thể của các hệ thống:



Hình 2.2: Sơ đồ tổng thể mạch điện hệ thống

Nguồn cung cấp	- Hai ắc quy 12V DC (Acquy 12V DC) cung cấp nguồn cho hệ thống. - Các cầu chì (Cầu chì 1.5A) được sử dụng để bảo vệ mạch khỏi quá tải dòng điện. - Các công tắc (Công tắc) để bật/tắt nguồn cho các phần của mạch. - Đèn Led dùng để thông báo có dòng điện đi qua hay không.

Điện trở	- Điện trở 6k8Ω được sử dụng để giới hạn dòng điện.
Động cơ bước	- Bốn động cơ bước size 57 (Step motor Size 57) được điều khiển bởi các driver MD556.
Driver MD556	- Mỗi động cơ bước được điều khiển bởi một driver MD556. - Các kết nối từ driver đến động cơ bước bao gồm:A+,A-,B+,B-
Raspberry Pi 4	- Raspberry Pi 4 là bộ điều khiển trung tâm của hệ thống. - Các chân GPIO của Raspberry Pi được kết nối với các driver động cơ bước để điều khiển chúng
Cảm biến siêu âm	- Bốn cảm biến siêu âm (Cảm biến siêu âm) được sử dụng để phát hiện vật cản hoặc khoảng cách. - Các cảm biến này cũng được kết nối với Raspberry Pi.
Camera	- Một camera được kết nối với Server để ghi nhận hình ảnh hoặc video.
Nguồn 5V-DC	- Nguồn 5V DC để cung cấp nguồn cho Raspberry Pi và Camera
Router Wifi 4G	- Sử dụng router Wifi 4G để cung cấp Internet cho toàn bộ hệ thống hoạt động.

2.3. Kiến trúc phần mềm

2.3.1. Yêu cầu hệ thống

a) Hệ điều hành

- Hệ thống cần có một hệ điều hành (Operating System) để quản lý và điều phối các tác vụ và tài nguyên của hệ thống. Hệ điều hành cần hỗ trợ các tính năng cơ bản như quản lý bộ nhớ, quản lý nhiệm vụ, giao tiếp với phần cứng và quản lý lỗi.
- Hệ điều hành cần đủ nhẹ tương thích với phần cứng và các module phù hợp cho raspberry

b) Thuật toán điều khiển

Thuật toán phải đưa ra được độ chính xác cao và tính ổn định trong nhiều tình huống thực tế, giảm tối đa các phép toán có thể để tối ưu mô hình, có thể ra quyết định trong các tình huống phức tạp,...

c) Thuật toán xử lý hình ảnh

Hình ảnh đầu vào cần được tiền xử lý và đưa vào mô hình, phải trích xuất được các đặc trưng từ hình ảnh, đưa ra được vị trí các vật cản, dựa vào đó có thể xác định kích thước và vị trí vật cản để có thể đưa ra hướng đi tốt nhất

d) Quản lý dữ liệu

Dữ liệu phải được thu thập và lưu trữ vào bộ nhớ, phải được tổ chức một cách rõ ràng và đầy đủ, lượng dữ liệu này sẽ được sử dụng cho mô hình tiếp tục học và phát triển, cải thiện độ chính xác

e) Kết nối mạng

Kết nối mạng phải đảm bảo tín hiệu đường truyền, không giật lag, không mất mát dữ liệu, không gián đoạn trong quá trình truyền tải để mô hình có thể hoạt động một cách ổn định nhất

f) Tương thích và mở rộng

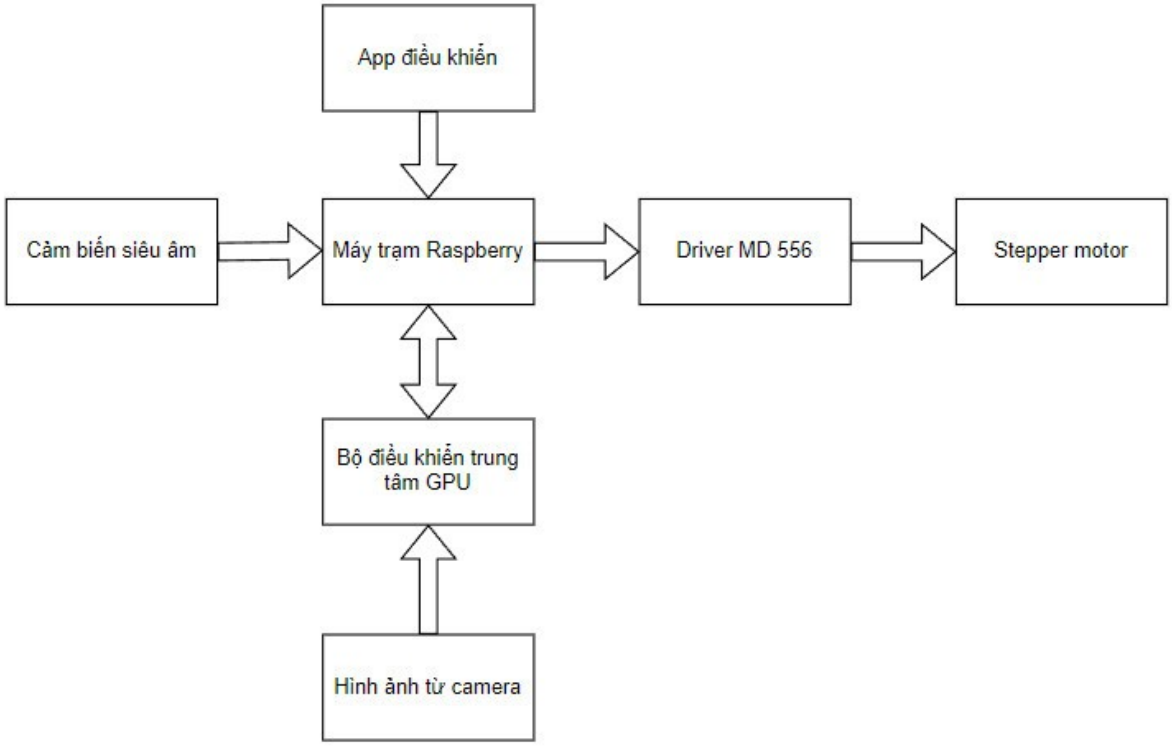
Hệ thống phải có sự tương thích cho nhiều loại mô hình, phù hợp cho sự phát triển trong thực tế, kết hợp nhiều mô hình lại với nhau để đưa ra một cái nhìn tổng quan về dữ liệu thực tế và có quyết định chính xác nhất

2.3.2. Thiết kế hệ thống

a) Tổng quan về hệ

Hệ thống là sự kết hợp giữa các mô hình AI, xử lý ảnh, các phép toán dựa trên các dữ liệu đầu vào từ camera và cảm biến để có thể đưa ra quyết định di chuyển, tránh vật cản, bám làn, có thêm hệ thống xử lý sự cố ngoài ý muốn như

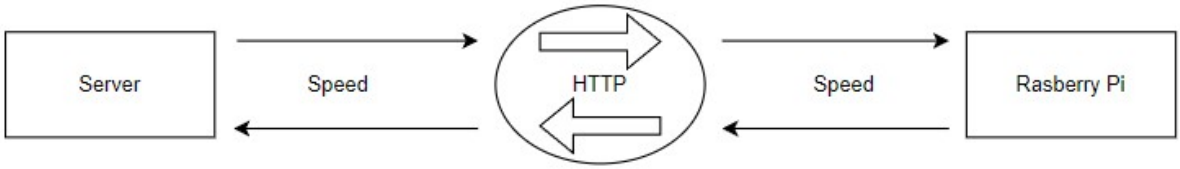
mất kết nối, trong tình trạng thực tế xe không di chuyển được để tránh xảy ra rủi ro ngoài ý muốn



Hình 2.3: Sơ đồ khối hệ thống tự hành

b) Giao tiếp HTTP

Dưới đây là sơ đồ mô tả cho cách hoạt động của giao thức HTTP về trao đổi dữ liệu giữa Server và Raspberry pi.



Hình 2.4: Hoạt động của giao thức HTTP trao đổi thông tin giữa Raspberry và Server

c) Xử lý đa luồng trên RaspberryPi

Vì hệ thống cần sự chính xác và tốc độ vậy nên việc xử lý đa luồng là rất cần thiết để hệ thống có thể tính toán 1 cách độc lập, nhận các tín hiệu điều khiển từ nhiều nguồn khác nhau, xử lý song song cho các tình huống,...

d) Ra quyết định

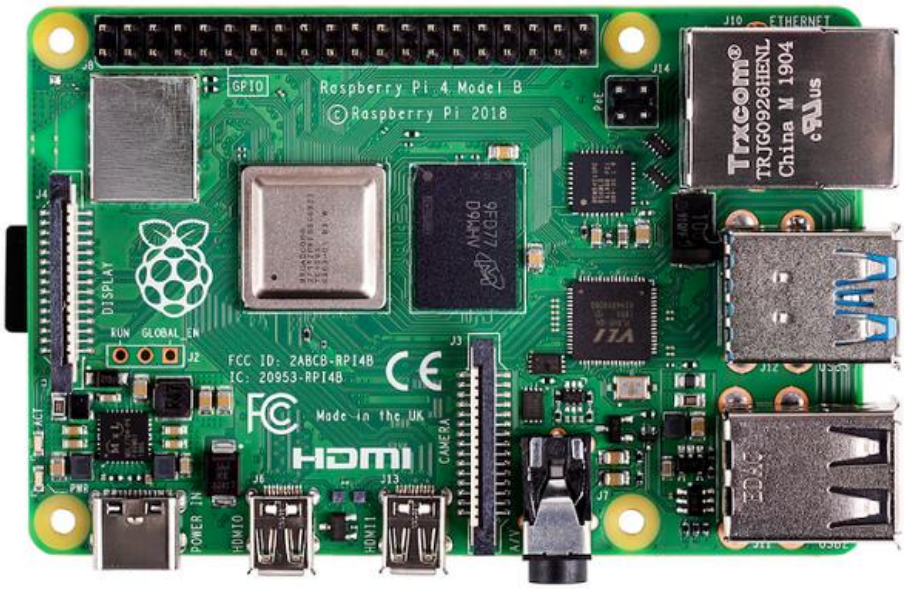
Sau khi hệ thống tính toán xong và đưa ra quyết định hướng đi, dữ liệu điều khiển sẽ được truyền từ server xuống cho raspberry để điều khiển động cơ di chuyển theo hướng mong muốn.

Chương 3 :TRIỂN KHAI VÀ KẾT QUẢ THỰC NGHIỆM

3.1. Lựa chọn phần cứng

3.1.1. Board mạch Raspberry Pi

Trong dự án này tôi sử dụng Raspberry Pi 4 Model B để làm bộ xử lý chính trên xe với nhiệm vụ là bộ xử lý chính cho hệ thống xe tự hành. Trên Raspberry là nơi nhận thông tin từ server, đồng thời là nơi thu nhận các giá trị từ cảm biến siêu âm và gửi về cho server và ra là nơi điều khiển quyết định di chuyển của xe.



Hình 3.1: Raspberry Pi 4 Model B- 4GB

a) Thông số kỹ thuật Raspberry Pi 4 Model B

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- RAM 4GB
- Wifi chuẩn 2.4 GHz và 5.0 GHz IEEE 802.11ac. Bluetooth 5.0, BLE
- Cổng mạng Gigabit Ethernet
- 2 cổng USB 3.0 và 2 cổng USB 2.0
- Chuẩn 40 chân GPIO, tương thích với các phiên bản trước
- Hỗ trợ 2 cổng ra màn hình chuẩn Micro HDMI với độ phân giải lên tới 4K
- Cổng MIPI DSI

- Cổng MIPI CSI
- Cổng AV 4 chân
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.0 graphics
- Khe cắm Micro-SD cho hệ điều hành và lưu trữ
- Nguồn điện DC 5V – 2A DC chuẩn USB-C
- 5V DC via GPIO header (minimum 3A*)
- Hỗ trợ Power over Ethernet (PoE) (yêu cầu có PoE HAT)

b) Thiết bị đi kèm

- Nguồn DC 5V=2A
- Thẻ SD 32GB

c) Lý do lựa chọn

- Raspberry pi 4 Model B có kích thước nhỏ gọn chỉ bằng 1 thẻ tín dụng cho phép người dùng dễ dàng tích hợp vào hệ thống
- Với 40 cổng GPIO cho phép Raspberry dễ dàng mở rộng, tích hợp với các module khác
- Hỗ trợ phần mềm Raspberry Pi chạy trên hệ điều hành Linux và có sẵn nhiều phiên bản hệ điều hành như Raspbian, Ubuntu Mate và RetroPie, có thể sử dụng Python, Java, C/C++ tạo điều kiện tốt để phát triển phần mềm
- Cộng đồng lớn và nhiệt tình, dễ dàng tìm kiếm tài liệu hướng dẫn

3.1.2. Camera ImouLife

Chúng tôi lựa chọn sử dụng Camera Cue2 – 1 dòng Ip camera được cung cấp bởi Dahua.

Hình 3.2: Camera Imoulife Cue2

a) Thông số kỹ thuật

- Độ phân giải 2 MEGAPixel cảm biến CMOS kích thước 1/2.7”, 25/30fps@2.0M(1920×1080)
- Chuẩn nén H.265 – Độ nhạy sáng tối thiểu 1.96lux/F1.2(color), 0 lux/F1.2(IR on)
- Chế độ ngày đêm(ICR), chống ngược sáng DWDR, tự động cân bằng trắng (AWB), tự động bù sáng (AGC), chống ngược sáng(BLC), Chống nhiễu (3D-DNR)
- Tầm xa hồng ngoại 10m với công nghệ hồng ngoại thông minh
- Tích hợp còi báo động cùng tính năng Human detection
- Ống kính cố định 2,8mm cho góc nhìn 112°(H), 59°(V), 135°(D)
- Tích hợp mic và loa với chuẩn âm thanh G.711a/G.711u/PCM
- Đàm thoại hai chiều
- Hỗ trợ khe cắm thẻ nhớ Micro SD, Max 256GB
- Tích hợp Wi-Fi(IEEE802.11b/g/n)
- Hỗ trợ P2P, chuẩn tương thích ONVIF
- Điện áp DC5V 1A, công suất <2,5W

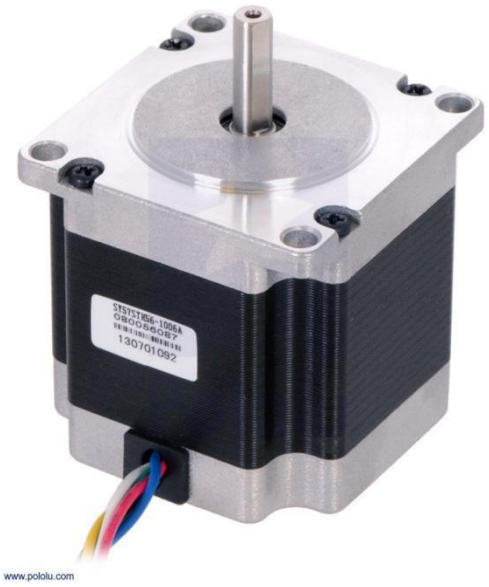
b) Lý do lựa chọn

- Kích thước nhỏ gọn nhẹ, dễ dàng kết nối thông qua RTSP

- Với những thông số kỹ thuật trên thì hình ảnh đầu ra đảm bảo chất lượng cho việc xử lý hình ảnh

3.1.3. Stepper Motor

Bộ Động Cơ Bước 57 57byg250b Mô Men Xoắn 1.2n. M Dài 56Mm + DM556 Bộ Điều Khiển 4.2A



Hình 3.3: Stepper Motor

a) Thông số kỹ thuật stepper Motor

- Kích thước mặt bích: 57x57 mm
- Chiều dài thân
- Dòng chịu tải 4A
- Góc bước 1,8°/step
- Momen xoắn trên trục: 1,2Nm
- Đường kính trục 8mm

b) Lý do lựa chọn

- Điều khiển: DM556 Bộ điều khiển 4.2A là một bộ điều khiển mạnh mẽ và đáng tin cậy cho bộ động cơ bước. Nó hỗ trợ dòng điện đến 4.2A, cho phép kiểm soát chính xác và ổn định các bước chuyển động của bộ động cơ.
- Tương thích: Bộ động cơ 57byg250b và DM556 Bộ điều khiển được thiết kế để tương thích với nhau, đảm bảo sự tương thích và khả năng làm việc

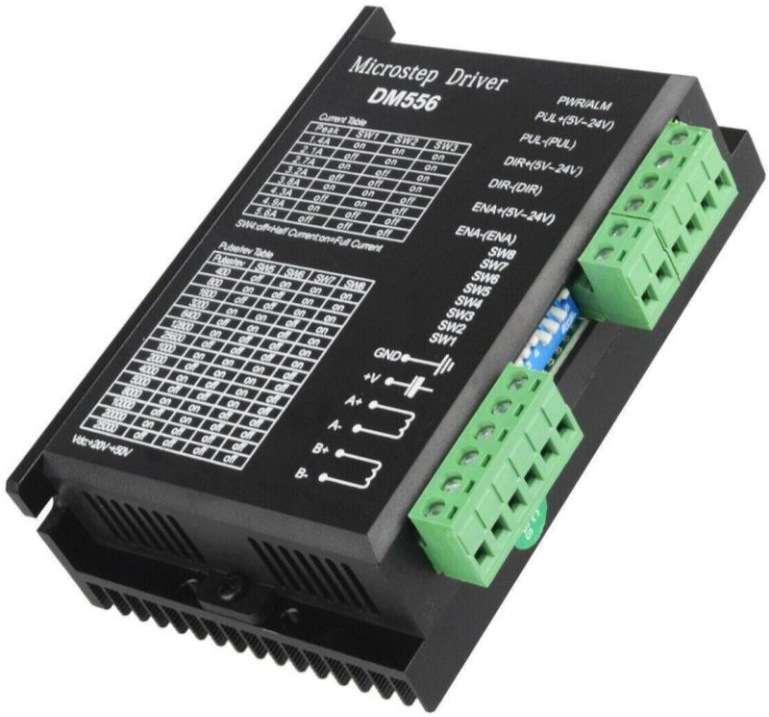
tốt với nhau. Điều này giảm đáng kể rủi ro lựa chọn các thiết bị không tương thích và đảm bảo tính ổn định của hệ thống.

- Hiệu suất: Kết hợp giữa bộ động cơ 57byg250b và DM556 Bộ điều khiển 4.2A cung cấp hiệu suất tốt. Ta có thể đạt được độ chính xác và đáng tin cậy trong việc điều khiển vòng quay và vị trí của bộ động cơ bước.

3.1.4. Driver DM556

Driver Điều Khiển Động Cơ Bước DM556 là phiên bản nâng cấp với IC và thuật toán xử lý DSP vượt trội cho khả năng chống nhiễu, độ ổn định và độ ồn thấp (thử nghiệm thực tế cho thấy tiếng ồn và độ rung của động cơ giảm đi rất nhiều lần, động cơ chạy êm, mượt), hình dạng, kích thước, cách sử dụng của DM556 hoàn toàn tương thích với phiên bản cũ.

Driver điều khiển động cơ bước DM556 được sử dụng để điều khiển động cơ bước 2 phase và 4 Phase, Driver có chất lượng tốt, độ bền và độ ổn định cao, là loại thường được sử dụng nhất trong máy Cắt Laser, máy CNC cỡ trung hiện nay.



Hình 3.4: Driver DM556

a) Thông số kỹ thuật

- Điện áp sử dụng: 20 ~ 50VDC.
- Dòng điện ngõ ra tối đa: Max 4.2A.

- Các tùy chỉnh vi bước: 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128; or 1/5, 1/10, 1/20, 1/25, 1/40, 1/50, 1/100, 1/125
- Các tùy chỉnh dòng điện: 1.0A, 1.46A, 1.91A, 2.37A, 2.84A, 3.31A, 3.76A, 4.20A
- Giảm độ nóng motor bằng công nghệ implementation of 3-state current control.
- Xung điều khiển: PULSE/DIRECTION & CW/CCW
- Tích hợp công nghệ tự động căn chỉnh phù hợp với động cơ.
- Tần số xung tối đa: 300 KHz
- Tương thích mức tín hiệu TTL 5VDC và tích hợp Opto cách ly giao tiếp với mạch điều khiển
- Automatic idle-current reduction.
- 15 selectable resolutions, up to 25,000 steps/rev
- Sử dụng cho động cơ 2 Phase và 4 Phase.
- Bảo vệ Ngắn mạch, Quá áp, dòng, nhiệt độ tích hợp.
- Kích thước: 4.65 X 2.97 X 1.30 inches; 10 oz
- CE, ROHS certified

b) Lý do lựa chọn

- Hiệu suất cao: Driver DM556 có khả năng cung cấp dòng điện cao và mô-men xoắn lớn cho động cơ bước. Điều này giúp đảm bảo hiệu suất cao và độ chính xác trong quá trình hoạt động.
- Bảo vệ quá tải: Driver DM556 có tính năng bảo vệ quá tải, giúp ngăn chặn việc động cơ bước bị hư hỏng do quá tải. Nếu dòng điện hoặc mô-men xoắn vượt quá giới hạn đã định trước, driver sẽ tự động tắt nguồn hoặc giảm dòng điện để bảo vệ động cơ.
- Chế độ chạy mượt mà: Driver DM556 có khả năng điều chỉnh tốc độ và gia tốc của động cơ bước. Điều này cho phép chạy mượt mà và tránh tình trạng rung, động cơ bước bị kẹt hoặc mất bước.
- Tính linh hoạt cao: Driver DM556 hỗ trợ nhiều chế độ điều khiển, bao gồm chế độ vị trí, chế độ vận tốc và chế độ tỷ lệ. Điều này cho phép ứng dụng trong nhiều lĩnh vực và kiểu điều khiển khác nhau.

- Dễ sử dụng: Driver DM556 có giao diện đơn giản và dễ sử dụng. Nó được thiết kế để dễ dàng cài đặt và kết nối với các thành phần khác trong hệ thống điều khiển.
- Độ tin cậy cao: Driver DM556 được xây dựng với các linh kiện chất lượng cao và tuân thủ các tiêu chuẩn sản xuất nghiêm ngặt. Điều này đảm bảo độ tin cậy cao và tuổi thọ dài cho driver.

3.1.5. 4G LTE Wi-fi Router APTEK- L300

4G WiFi Router APTEK L300 được thiết kế với một cổng Fast Ethernet WAN, RJ45, chuyên cung cấp tốc độ mạng tiêu chuẩn cho SOHO, hộ gia đình, văn phòng... Bên cạnh đó APTEK L300 còn được trang bị 4 cổng Fast Ethernet LAN, RJ45 cho phép kết nối thêm thiết bị hoặc phục vụ cho việc mở rộng mạng diễn ra dễ dàng.

Hình 3.5: 4G LTE Wi-fi Router APTEK- L300

a) Thông số kỹ thuật

- Router 4G/LTE 1 SIM slot tương thích với tất cả các nhà mạng, không cần cấu hình
- 1 cổng Fast Ethernet WAN
- 4 cổng Fast Ethernet LAN
- 2 anten băng tần 2.4GHz chuẩn 802.11n với tốc độ 300Mbps.
- 2 antenna LTE có thể tháo rời, độ nhạy cao.
- Hỗ trợ tính năng tự động thiết lập lại kết nối 3G/4G sau khi bị mất kết nối 3G/4G

- Chức năng giới hạn số lượng kết nối.
- Lập lịch tắt/mở WIFI, Lập lịch tự động Reboot.
- Vỏ sắt giúp thiết bị chịu được nhiệt độ cao, môi trường khắc nghiệt

b) Lý do lựa chọn

- Kết nối 4G/LTE: APTEK L300 hỗ trợ kết nối 4G/LTE, cho phép truy cập Internet với tốc độ cao và độ ổn định tốt.
- Tốc độ WiFi chuẩn N 300Mbps: Với tốc độ truyền dữ liệu lên đến 300Mbps, APTEK L300 cung cấp kết nối WiFi nhanh chóng và ổn định cho nhiều người dùng cùng lúc. Điều này rất hữu ích trong các môi trường có nhu cầu sử dụng Internet nhanh và độ trễ thấp, như trong việc truyền dữ liệu, truy cập trực tuyến và chia sẻ tài nguyên mạng.
- Tính linh hoạt và dễ dùng: APTEK L300 được thiết kế để dễ dàng cài đặt và cấu hình. Nó cung cấp giao diện quản lý web và hỗ trợ nhiều tính năng như quản lý băng thông, lọc địa chỉ MAC, tạo mạng khách và nhiều hơn nữa.
- Độ tin cậy và bảo mật: APTEK L300 có tính năng bảo mật cao, bao gồm mã hóa WPA/WPA2, hạn chế truy cập dựa trên địa chỉ MAC và tường lửa.

3.1.6. Bánh xe Omni

Sử dụng 4 bánh xe đa hướng Mecanum Omni, với 9 con lăn

Hình 3.6: Bánh Xe Mecanum 97mm

a) Thông số kỹ thuật

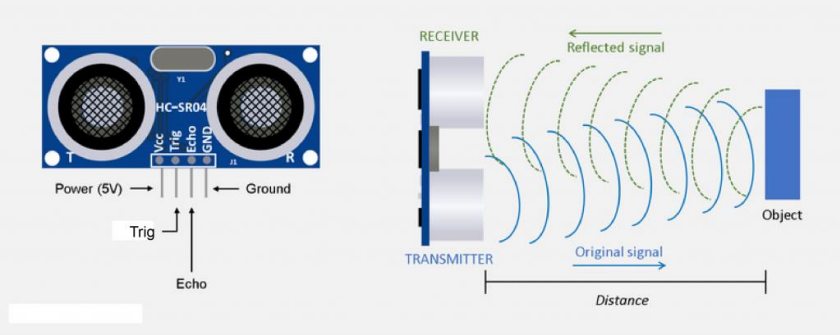
- Góc: 45 °
- Màu sắc: bạc, đen
- Chất liệu: nhựa + cao su silicon
- Kích thước:
    - Đường kính: 9.7cm
    - Độ dày: 4.6cm
- Trọng lượng : 250g

b) Nguyên lý sử dụng

- Rất linh hoạt trong việc di chuyển: Hệ thống bánh xe Mecanum Omni cho phép robot di chuyển theo nhiều hướng và quay theo chính xác 360 độ.
- Điều khiển chính xác: Với 9 con lăn trên 4 bánh xe, robot có khả năng di chuyển mượt mà và linh hoạt
- Tăng cường khả năng vượt chướng ngại vật: Với khả năng di chuyển theo nhiều hướng và quay linh hoạt, robot có thể vượt qua chướng ngại vật một cách dễ dàng hơn.

3.1.7. UltraSonic

Cảm biến siêu âm là một thiết bị có thể đo khoảng cách đến một vật thể bằng cách sử dụng sóng âm thanh.Cảm biến siêu âm đo khoảng cách bằng cách phát ra một sóng âm thanh ở một tần số cụ thể và lắng nghe sóng âm thanh đó dội lại. Bằng cách ghi lại khoảng thời gian trôi qua giữa sóng âm thanh được tạo ra và sóng âm thanh dội lại, có thể tính toán khoảng cách giữa cảm biến sonar và đối tượng.



Hình 3.7: UltraSonic và nguyên lý hoạt động

a) Thông số kỹ thuật

Số lượng sử dụng: 4

- Nguồn điện: +5V DC
- Dòng tĩnh: <2mA
- Dòng điện làm việc: 15mA
- Góc hiệu dụng: <15°
- Phạm vi khoảng cách: 2 cm – 400 cm/1″ – 13ft
- Độ phân giải: 0,3 cm
- Góc đo: 30 độ
- Độ rộng xung đầu vào kích hoạt: 10uS
- Kích thước: 45mm x 20mm x 15mm.

b) Sơ đồ thời gian

Biểu đồ thời gian của HC-SR04 được hiển thị. Để bắt đầu đo, Trig của SR04 phải nhận xung cao (5V) trong ít nhất 10us, điều này sẽ bắt đầu cảm biến sẽ phát ra 8 chu kỳ xung siêu âm ở 40kHz và đợi xung siêu âm phản xạ. Khi cảm biến phát hiện siêu âm từ máy thu, nó sẽ đặt chân Echo ở mức cao (5V) và trì hoãn trong một khoảng thời gian (chiều rộng) tỷ lệ thuận với khoảng cách. Để có được khoảng cách, hãy đo chiều rộng (Tấn) của chân Echo.

3.1.8. Nguồn cấp điện

Sử dụng 1 nguồn acquy loại 12V-5Ah

Hình 3.8: Nguồn 12V/5Ah

a) Thông số kỹ thuật

- Kích thước (Dài x Rộng x Cao): **120 x 60 x 130** (mm)
- Trọng lượng: **~ 2.2**kg

b) Lý do lựa chọn

- Có kích thước tương đối nhỏ gọn
- Điện dung phù hợp cho hoạt động của stepper motor
- Đủ điều kiện đáp ứng sử dụng làm nguồn cho, stepper motor và Router wifi

3.2. Triển khai mô hình nhận diện và đánh giá

3.2.1. Tổng quan về hệ thống nhận diện

Nhận diện môi trường của lĩnh vực xe tự hành có rất nhiều thứ nhưng tập trung chủ yếu nhất vào:

- Phát hiện làn đường
- Nhận diện đối tượng
- Đưa ra quyết định cho xe

Trong đồ án này tôi chúng tôi tập trung giải quyết ba bài toán trên. Đặc biệt là việc xử lý phát hiện đối tượng và ra quyết định cho xe ở trong mọi môi trường thực tế.

3.2.2. Phát hiện làn đường

3.2.2.1. Tổng quan bài toán

Phát hiện làn đường là một nhiệm vụ thị giác máy tính liên quan đến việc xác định ranh giới của các làn đường lái xe trong video hoặc hình ảnh về cảnh đường. Mục tiêu là định vị và theo dõi chính xác vạch kẻ đường trong thời gian thực, ngay cả trong những điều kiện khó khăn như ánh sáng kém, ánh sáng chói hoặc bố cục đường phức tạp.

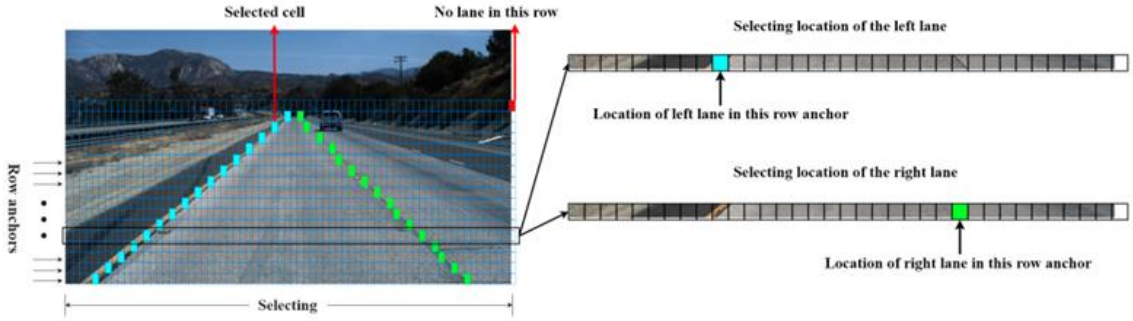
Phát hiện làn đường là một thành phần quan trọng nhất của hệ thống xe tự hành, vì nó cung cấp thông tin về bố cục đường và vị trí của xe trong làn đường, điều này rất quan trọng đối với việc điều hướng và đảm bảo an toàn.

3.2.2.2. Ultra Fast Lane Detection

a) Giới thiệu và kiến trúc mô hình

Mô hình Ultra Fast Lane Detection (UFLD) là một mô hình nhận dạng làn đường dựa trên deep learning, được đề xuất bởi nhóm tác giả Trung Quốc vào năm 2020. Mô hình này sử dụng mạng neural tích chập (CNN) để trích xuất đặc trưng của ảnh đầu vào,

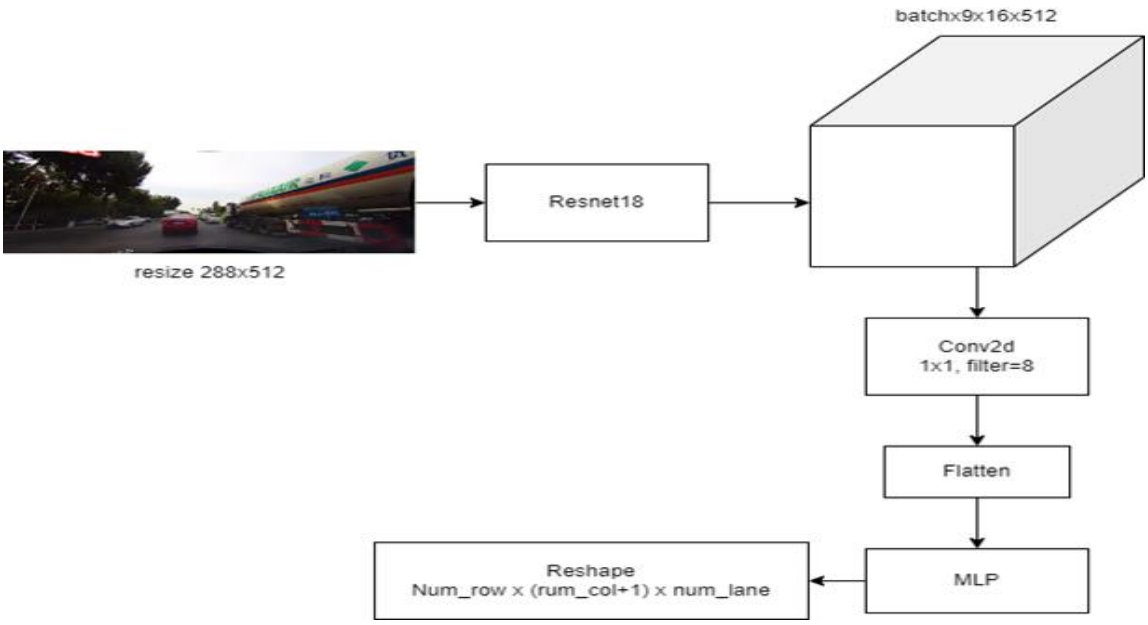
sau đó sử dụng một mô hình hồi quy để dự đoán các thông số của làn đường. Mô hình sử dụng phương pháp lựa chọn và phân loại vị trí làn đường dựa trên các hàng và chia hình ảnh thành các hàng và các cột để nhận diện để giảm thiểu tối đa thời gian nhận diện.



Hình 3.9: Phương pháp phân loại vị trí theo hàng của UFLD

Mô hình UFLD thay vì sử dụng điểm trung tâm ô thì các giao điểm của hàng và cột sẽ làm đại diện. Mô hình UFLD cũng chỉ quan tâm tới vùng ảnh có làn đường nhưng không cần cắt ảnh đưa vào mô hình như mô hình của tôi.

b) Kiến trúc mô hình



Hình 3.10 Kiến trúc mô hình UFLD

Mô hình UFLD đã sử dụng mô hình Resnet18 làm backbone rồi đưa qua lớp tích chập để trích xuất feature map nhằm mục đích trích xuất các đặc trưng cục bộ và tăng khả năng học cho mô hình. Sau đó qua các lớp fully connected để phân loại.

c) Hàm mất mát

Hàm mất mát phân loại vị trí của mô hình UFLD cũng sử dụng softmax loss.

Để giải quyết vấn đề tính ổn định của hình dạng đường và tính liên tục của đường thì mô hình UFLD đã đề xuất thêm hai hàm mất mát cho việc đó.

Hàm mất mát cho tính liên tục theo công thức (hình 3.11):

$$L_{sim} = \sum_{i=1}^{C} \sum_{j=1}^{h-1} \left\| P_{i,j,:} - P_{i,j+1,:} \right\|_1$$

Hình 3.11: Hàm mất mát tính liên tục UFLD

: Giá trị dự đoán vị trí của đường i hàng j

: Hàm smooth L1 loss

Mô hình UFLD tính vị trí của đường ở hàng i đường j bằng cách:

$$Prob_{i,j,:} = softmax(P_{i,j,1:w}), \qquad Loc_{i,j} = \sum_{k=1}^{w} k \cdot Prob_{i,j,k}$$

Hình 3.12: Tính vị trí của đường

: Xác suất của từng vị trí đường i hàng j

: Vị trí của đường i hàng j

Hàm mất mát cho hình dạng của đường:

$$L_{shp} = \sum_{i=1}^{C} \sum_{j=1}^{h-2} \| (Loc_{i,j} - Loc_{i,j+1}) - (Loc_{i,j+1} - Loc_{i,j+2}) \|_1,$$

Hình 3.13 Hàm mất mát cho hình dạng đường UFLD

Tổng mất mát:

$$L_{str} = L_{sim} + \lambda L_{shp} \quad L_{total} = L_{cls} + \alpha L_{str}$$

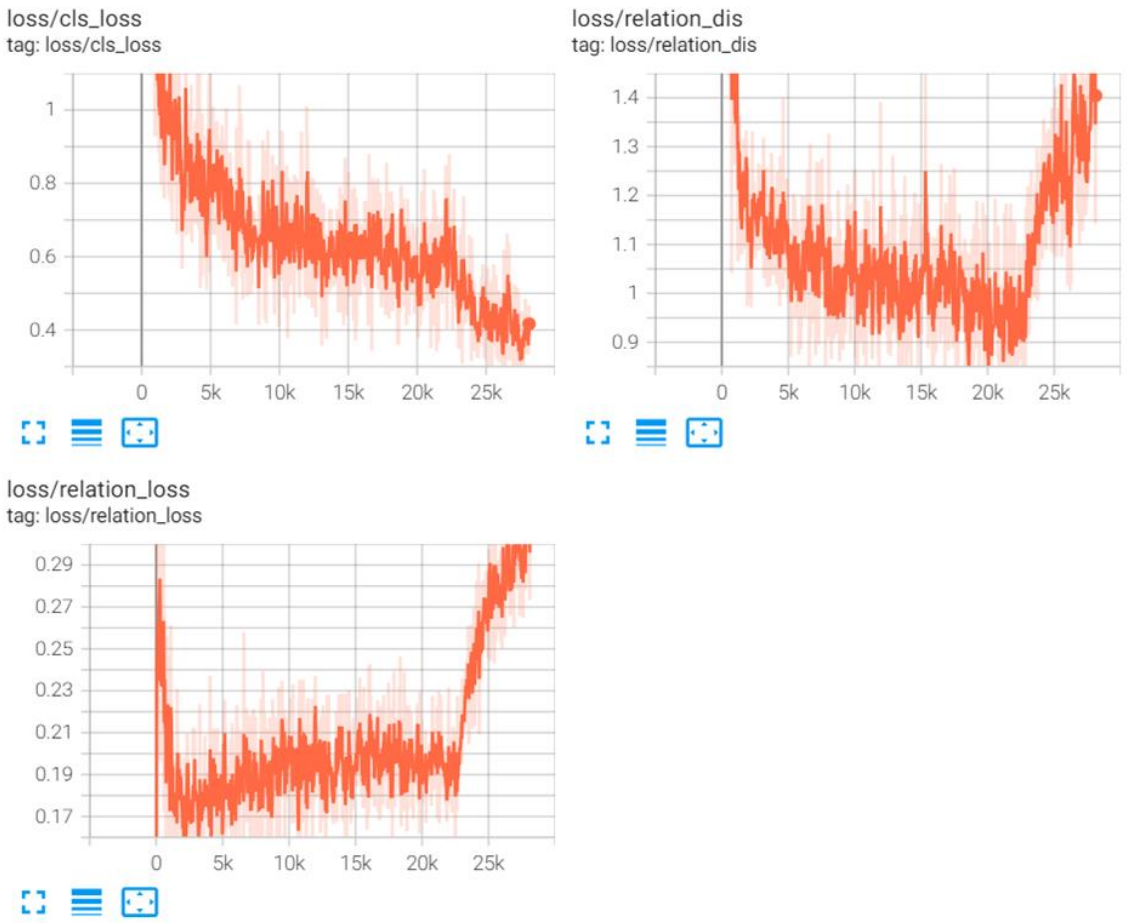
Hình 3.14: Tổng mất mát của UFLD

Hệ số giữa hàm mất mát liên tục và hàm mất mát hình dạng.

: Hệ số giữa hàm mất mát phân loại và tổng hai hàm mất mát trên

d) Đánh giá

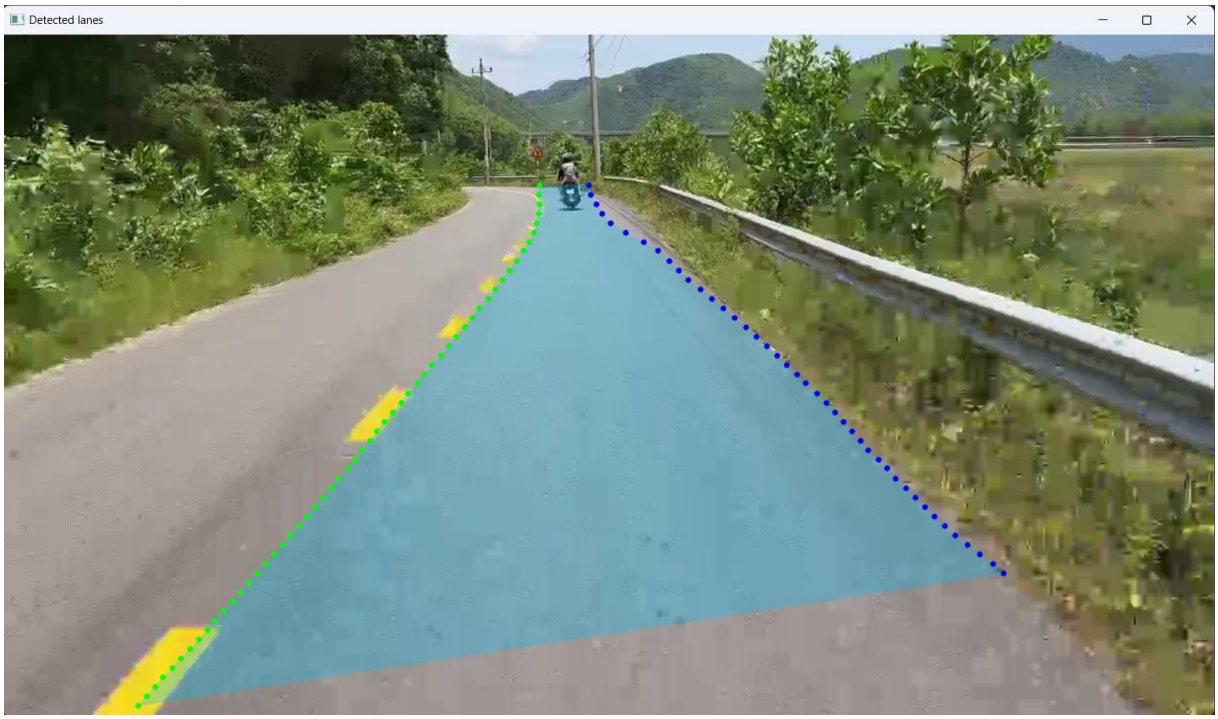
Sau khi huấn luyện, tôi có các đồ thị mất mát như sau (Hình 3.15)



Hình 3.15: Đồ thị mất mát

Hàm mất mát của relation_dis và relation_loss có xu hướng về cuối vì để nhận diện với các làn ngoài thì việc đánh giá vị trí của làn đường theo hàng sẽ không đúng nữa . Vì thế mặc dù hai hàm mất mát tăng nhưng hàm mất mát cho việc phân loại cũng cải thiện.

Độ chính xác của mô hình cho việc phân loại làn đường trên tập liệu test là 71.3% cho tập dữ liệu test. Độ chính xác được cải thiện và khá tốt chứng tỏ việc sử dụng lớp tích chập để tăng khả năng học và sử dụng hai hàm mất mát trên vào việc học hiệu quả. Mặc dù hai hàm mất mát tăng nhưng theo như kết quả nhận diện (hình 3.15) từ mô hình đã ổn định và giúp việc train nhanh hơn trong những epoch đầu tiên.



Hình 3.16: Kết quả nhận diện của mô hình UFLD

3.2.3. Nhận diện đối tượng

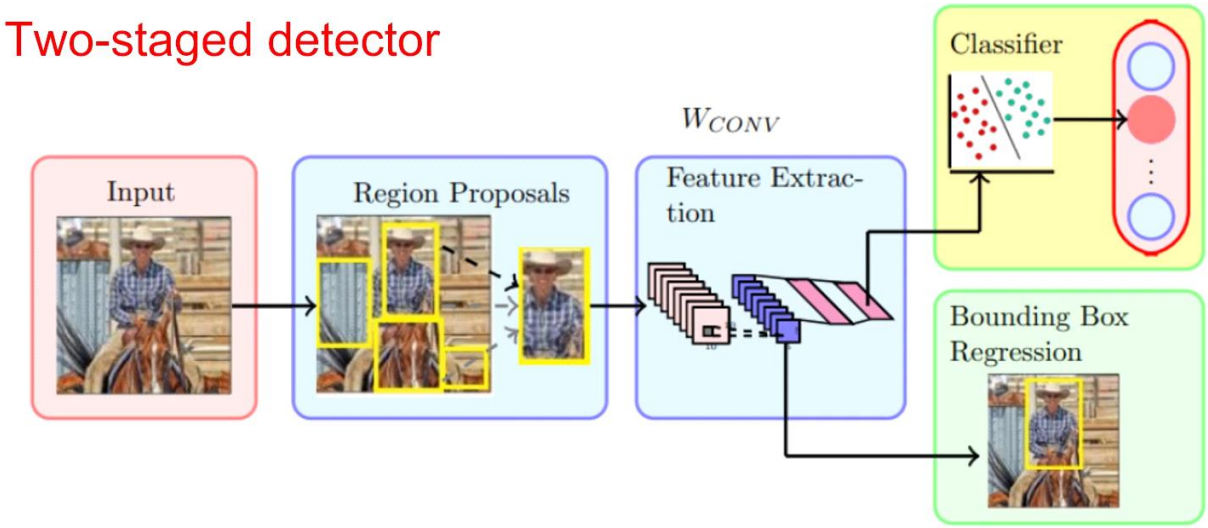
3.2.3.1. Tổng quan bài toán nhận diện đối tượng

Bài toán nhận diện đối tượng có hai loại mô hình để giải quyết vấn đề này:

- Two Staged Detector (R-CNN, Fast R-CNN, Faster R-CNN, …)
- One Staged Detector (SSD MobilenetV2, Yolo, ...)

- **Mô hình Two Staged Detector**

  Kiến trúc mô hình Two-staged detector



Hình 3.17: Mô hình Two-staged

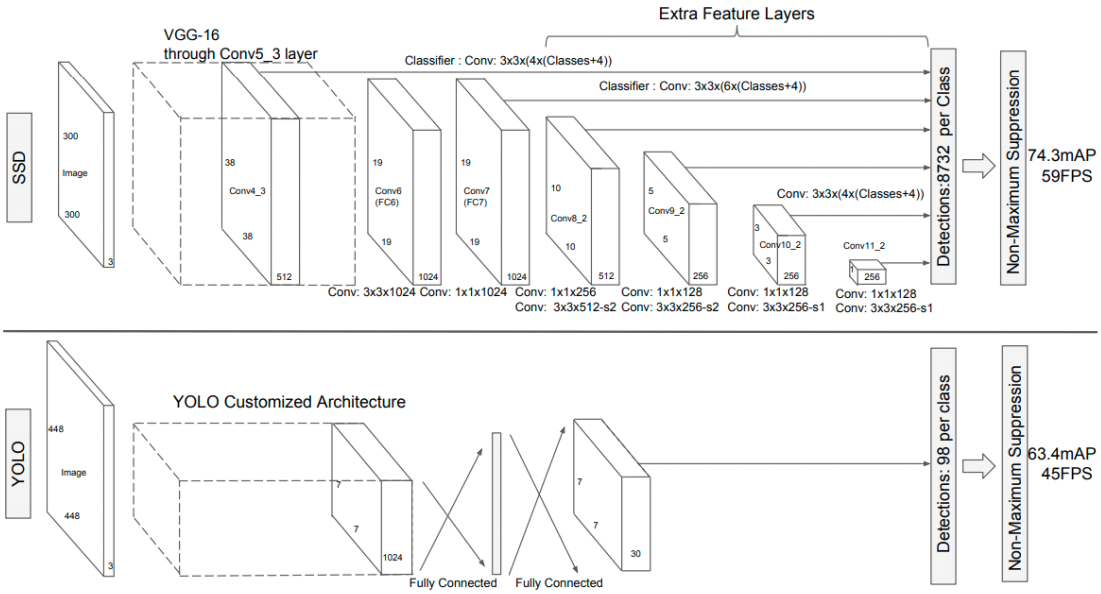
Dạng mô hình Two-Staged Detector có hai giai đoạn:

+ Đề xuất các hộp có thể chứa vật thể (Region Proposals)
+ Phân loại các hộp đó và sàng lọc đúng vị trí

Ở giai đoạn thứ nhất, mô hình R-CNN đã sử dụng thuật toán Selective Search để lấy được các hộp đề xuất trong hình ảnh input mà có khả năng chứa đối tượng. Tới giai đoạn thứ hai, ta đưa các hộp đề xuất vào backbone chẳng hạn như Resnet 50, VGG… để trích xuất đặc trưng các hộp đó để ra được các feature map và các lớp fully connected để encode các feature map để đưa ra các offset của các hộp đã được đề xuất ở giai đoạn một và phân loại các đối tượng đó.

- **Mô hình One-Staged Detector**

Kiến trúc mô hình One-Staged Detector ở hình 3.18:



Hình 3.18: Mô hình One-staged

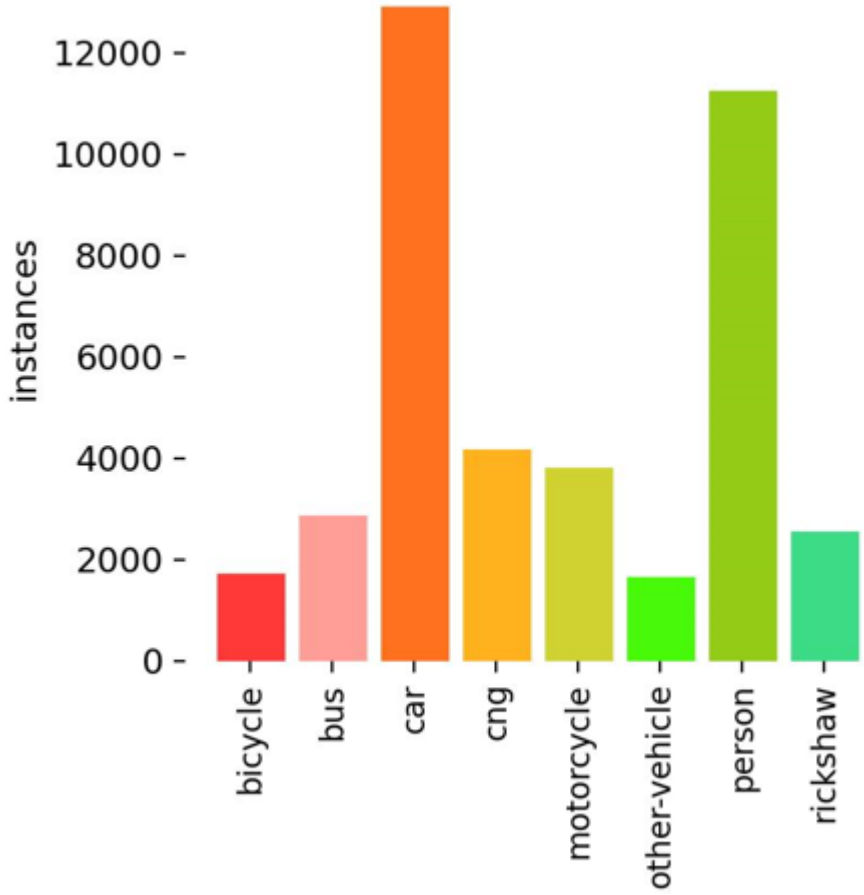
Dạng mô hình One-Staged Detector sẽ tính toán trong một lần truyền mạng liên tiếp. Trái ngược với các phương pháp Two-Staged, mô hình tạo ra các hộp mặc định có sẵn với các tỉ lệ khác nhau và việc còn lại chỉ cần sử dụng các lớp tích chập hoặc các lớp fully connected để tính toán các offset của các hộp có sẵn và phân loại các hộp đó [6]. Chính vì vậy nên dạng mô hình One-Staged Detector sẽ nhẹ hơn và xử lý nhanh hơn dạng mô hình Two-Staged nhưng độ chính xác thì sẽ không bằng các mô hình Two-Staged.

Kết luận

Dựa vào các ưu điểm và nhược điểm đã được nêu ra ở trên thì tôi chọn dạng One-Staged để sử dụng trong đồ án vì mô hình nhẹ rất phù hợp với máy tính nhúng nhỏ như jetson nano và tốc độ xử lý nhanh sẽ hiệu quả hơn với bài toán thời gian thực này.

3.2.3.2. Tập dữ liệu

P2 Dhaka Dataset Image Dataset là một bộ dữ liệu lớn về hình ảnh và video đường phố ở Hoa Kỳ được sử dụng để huấn luyện các mô hình trí tuệ nhân tạo trong lĩnh vực xe tự hành. Bộ dữ liệu này chứa hơn 100.000 hình ảnh và video đường phố được gắn nhãn, với các đối tượng như xe hơi, xe tải, xe buýt, người đi bộ , vv. Có một vài nhãn có rất ít ảnh ví dụ như tàu hỏa chỉ có 10 ảnh trong tập dữ liệu này nên tôi đã quyết định xóa các nhãn có số lượng ảnh ít. Sau đây là các nhãn còn lại sau khi được lọc. Và tôi đã thống kê các số lượng đối tượng trong tổng ảnh của từng class (hình 3.19)



Hình 3.19: Phân bố dữ liệu của các nhãn - images

Trong ảnh có nhiều đối tượng khác nhau, có đối tượng xuất hiện nhiều trong các ảnh, có đối tượng xuất hiện với tần suất ít hơn khoảng 5-10 ảnh mới có một đối tượng.

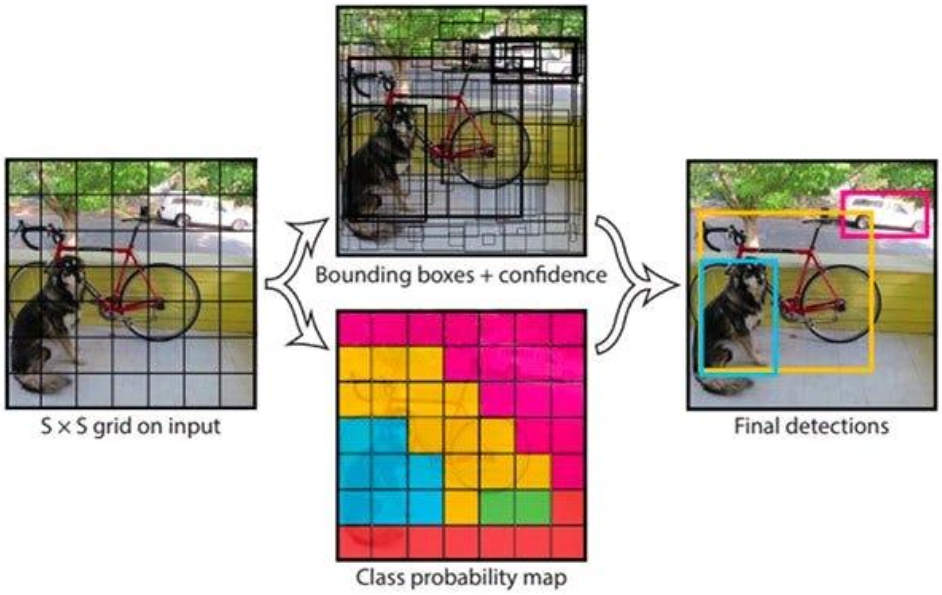
3.2.3.3. Mô hình Yolov8

a) Giới thiệu và kiến trúc mô hình

Yolo (You Only Look Once) là một mô hình nhận diện đối tượng (object detection) trong lĩnh vực thị giác máy tính (computer vision). Mô hình Yolo được phát triển bởi Joseph Redmon và các đồng nghiệp tại Đại học Washington.

Mô hình Yolo hoạt động bằng cách chia ảnh đầu vào thành một lưới ô vuông và dự đoán các hộp giới hạn (bounding boxes) cho các đối tượng trong mỗi ô vuông và đây là điểm khác biệt của Yolo với SSD. Thay vì SSD sẽ đánh giá từng hộp mặc định trong một ô thì Yolo sẽ đánh giá theo từng ô. Mỗi hộp giới hạn được dự đoán bởi một số lượng các thông số, bao gồm tọa độ của góc trái trên và góc phải dưới của hộp, độ tin cậy của dự đoán và xác suất của đối tượng được phát hiện trong hộp đó.

Quá trình nhận diện đối tượng của Yolo được biểu diễn theo hình 3.20

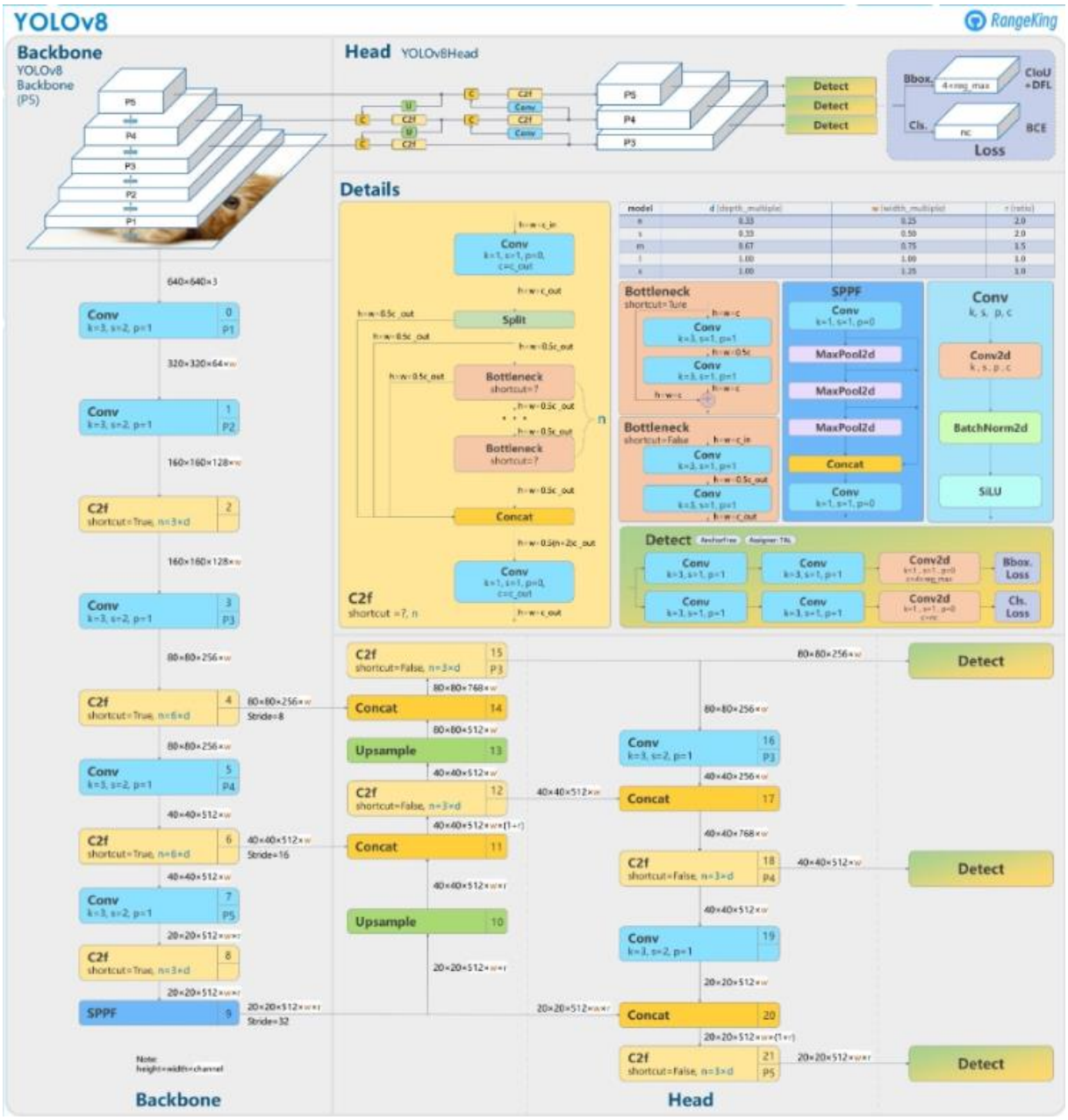


Hình 3.20: Quá trình nhận diện đối tượng của Yolov8

YOLOv8 là một thuật toán thị giác máy tính được sử dụng để phát hiện đối tượng. Mô hình này được phát triển dựa trên các phiên bản trước của YOLO (You Only Look Once – Nhận diện vật thể bằng một lần nhìn). Ngoài ra, YOLOv8 còn được thiết kế để phát hiện đối tượng trong thời gian thực và có thể xử lý hình ảnh trong vài mili giây. Hơn nữa Yolov8 còn hỗ trợ rất nhiều loại Backbone khác nhau chẳng hạn như EfficiencyNet, ResNet và CSPDarknet, giúp người dùng linh hoạt chọn mô hình tốt nhất cho trường hợp sử dụng cụ thể của họ và còn rất nhiều các tính năng đặc biệt khác. Vì vậy nên chúng em chọn Yolov8 để triển khai trong đồ án của mình.

- **Kiến trúc của Yolov8**

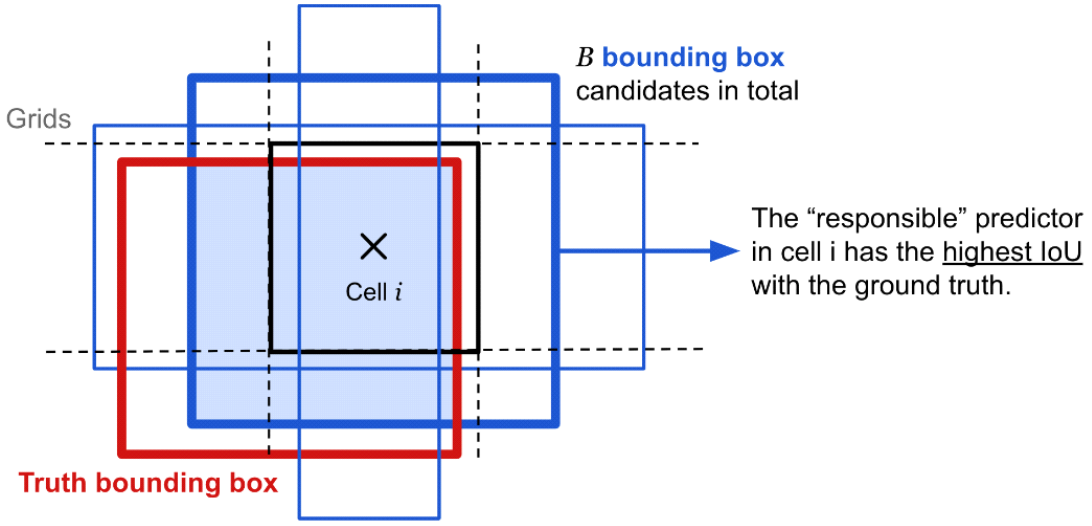
  YOLOv8 là một mô hình nhận dạng đối tượng dựa trên mạng convolutional neural network (CNN) được phát triển bởi Joseph Redmon và nhóm nghiên cứu của ông tại Đại học Washington.



Hình 3.21 : Kiến trúc mô hình Yolov8

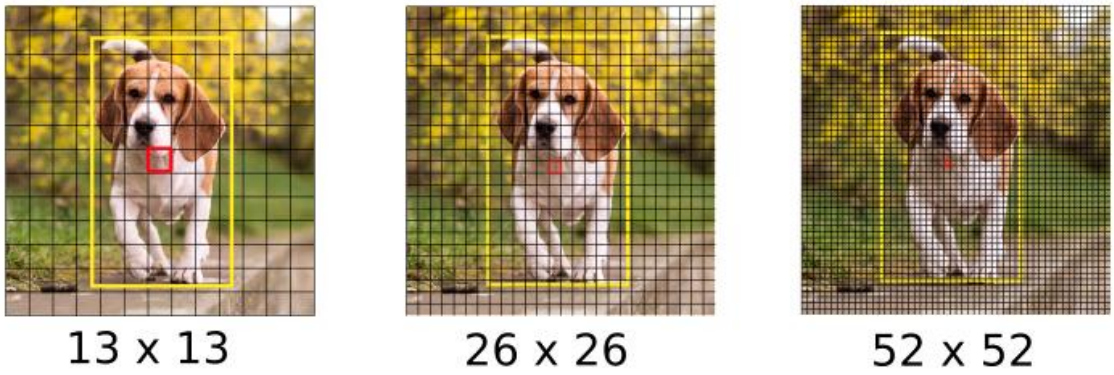
Ta cũng có thể tự hình dung kiến trúc YOLOv8 bằng cách chuyển đổi nó sang định dạng ONNX. ONNX viết tắt của Open Neural Network Exchange. Đây là định dạng mở được sử dụng để biểu thị các mô hình học máy. Nó có thể được xem như một đại diện chung cho các mô hình ML, bởi vì tính phổ biến cho bất kỳ mô hình nào, dù code mô hình được viết trong PyTorch, TensorFlow hoặc các framework nào khác.

Để tìm được bounding box cho vật thể, YOLO sẽ cần các anchor box làm cơ sở ước lượng tương tư các default boxes của SSD. Những anchor box này sẽ được xác định trước và sẽ bao quanh vật thể một cách tương đối chính xác. Sau này thuật toán regression bounding box sẽ tinh chỉnh lại anchor box để tạo ra bounding box dự đoán cho vật thể . Anchor box trong YOLO được thể hiện ở hình 3.22:



Hình 3.22: Anchor boxes trong Yolo

Yolo cũng dự báo trên nhiều feature map . Những feature map ban đầu có kích thước nhỏ giúp dự báo được các object kích thước lớn. Những feature map sau có kích thước lớn hơn trong khi anchor box được giữ cố định kích thước nên sẽ giúp dự báo các vật thể kích thước nhỏ được thể hiện ở hình 3.23.

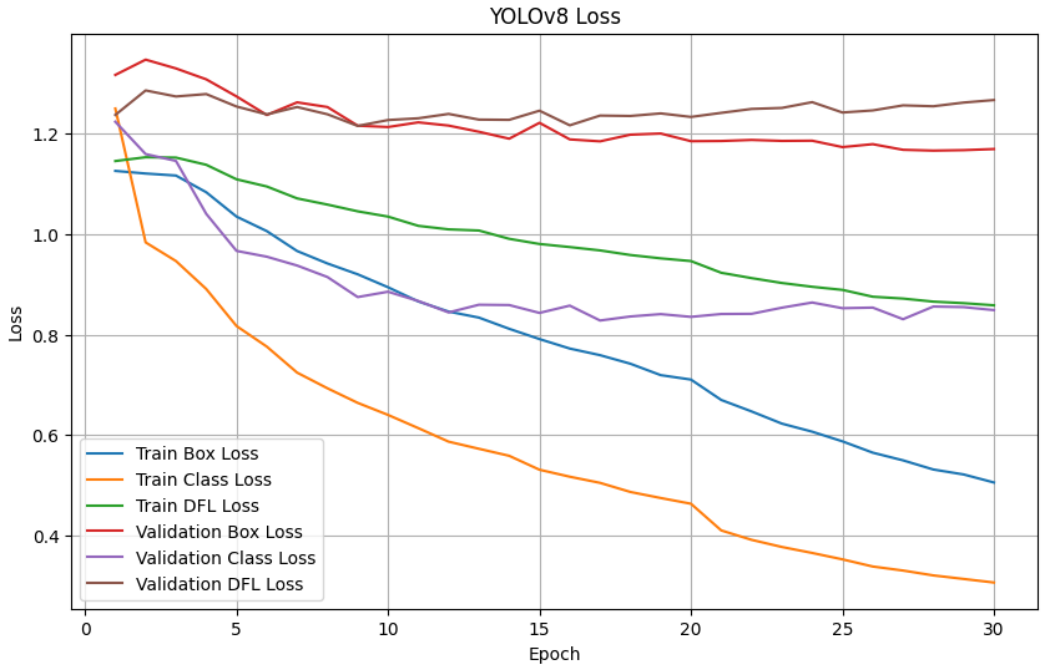


Hình 3.23: Dự báo trên nhiều feature map Yolo

Mô hình Yolov8 có 2 feature map nhằm mục đích xác định các vật thể với các tỉ lệ khác nhau có hình dạng lần lượt là [(1, 26, 26, num_anchor*(num_class+5)); (1, 13, 13, num_anchor*(num_class+5))]

Đánh giá mô hình

Sau khi huấn luyện, chúng tôi có đồ thị mất mát ở hình 3.25.



Hình 3.24: Đồ thị hàm mất mát

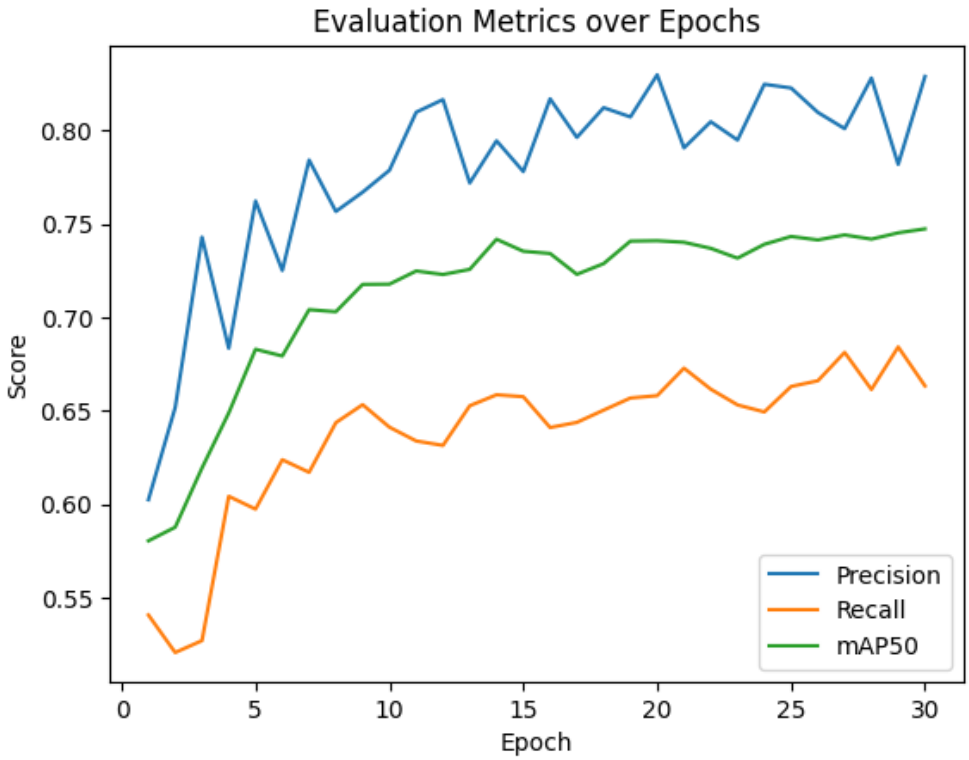
Tôi dựa vào metrics như precision, recall, và mAP từ mỗi epoch trong quá trình huấn luyện và đánh giá mô hình. Và kết quả tôi tính toán và đưa ra được ở hình 3.25. Và kết quả cho tôi nhận được đó là:

Mean Precision: 0.7790969999999999

Mean Recall: 0.636935

Mean mAP: 0.7122953333333335

Dựa vào kết quả cho ta thấy Mean Precision (Độ chính xác trung bình) có độ chính xác là khoảng 77.9% các bounding box được dự đoán là đúng trên toàn bộ dữ liệu kiểm tra. Mean Recall (Tỷ lệ nhớ trung bình) Giá trị trung bình của recall là khoảng 0.637, có nghĩa là khoảng 63.7% các bounding box thực tế đã được dự đoán đúng trên toàn bộ dữ liệu kiểm tra. Mean mAP (Mean Average Precision - Trung bình độ chính xác trên toàn bộ lớp) Giá trị trung bình của mAP50 là khoảng 0.712, điều này có nghĩa là trung bình khoảng 71.2% đối tượng trong các lớp đã được dự đoán chính xác trên toàn bộ dữ liệu kiểm tra với ngưỡng IoU là 0.5.



Hình 3.25: Đồ thị đánh giá mô hình Yolov8

3.2.3.4. Kết Luận

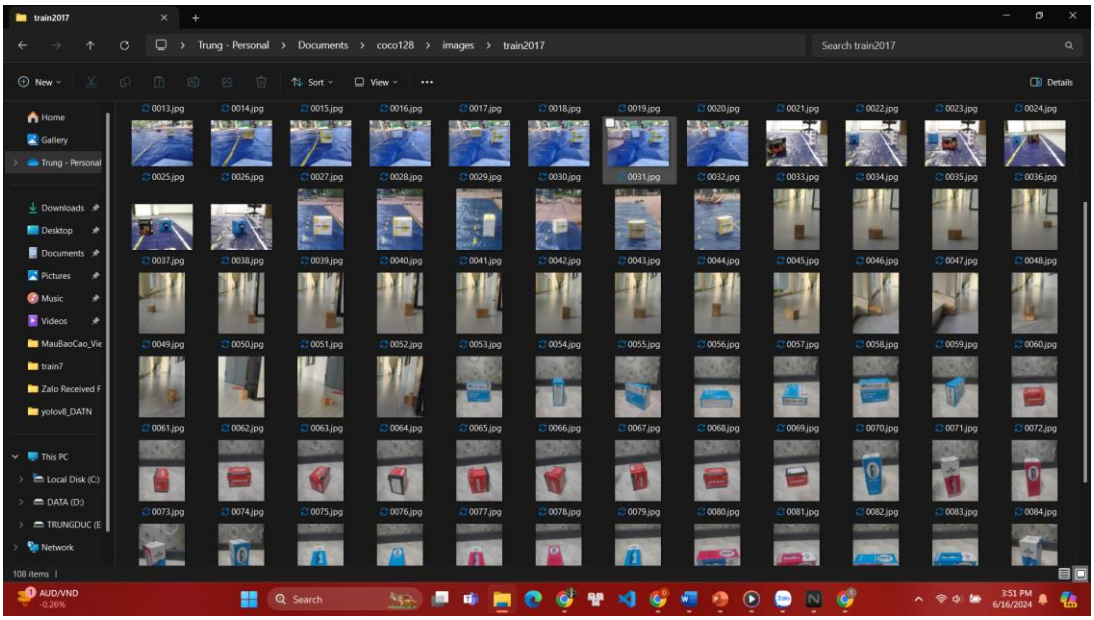
Yolov8 là một mô hình nhận diện đối tượng dựa trên mạng nơ-ron tích chập (CNN) và sử dụng một kiến trúc mạng nơ-ron sâu để phát hiện đối tượng. Yolov8 sử dụng một phương pháp phát hiện đối tượng đơn giản hơn so với SSD MobileNetV2, với một lần chạy qua mạng để phát hiện tất cả các đối tượng trong ảnh. Yolov8 có thể đạt được tốc độ xử lý nhanh hơn so với SSD MobileNetV2, và độ chính xác đã cải tiến hơn rất nhiều.

3.2.4. Triển khai thuật toán đưa ra quyết định cho xe

Để mô hình có khả năng đưa ra quyết định chính xác, điều kiện tiên quyết phải đáp ứng được 1 bộ dữ liệu tốt. Tuy nhiên với kích thước mô hình còn nhỏ để thể xây dựng được bộ dữ liệu tốt ta cần liên tục điều chỉnh các tham số, lựa chọn môi trường để huấn luyện phù hợp. Sau đây là phương pháp tôi sử dụng:

a) Thu thập dữ liệu trong môi thường thực tế

Sau khi xác định được yêu cầu bài toán chúng em tiến hành đi thu thập hình ảnh từ môi trường thực tế.

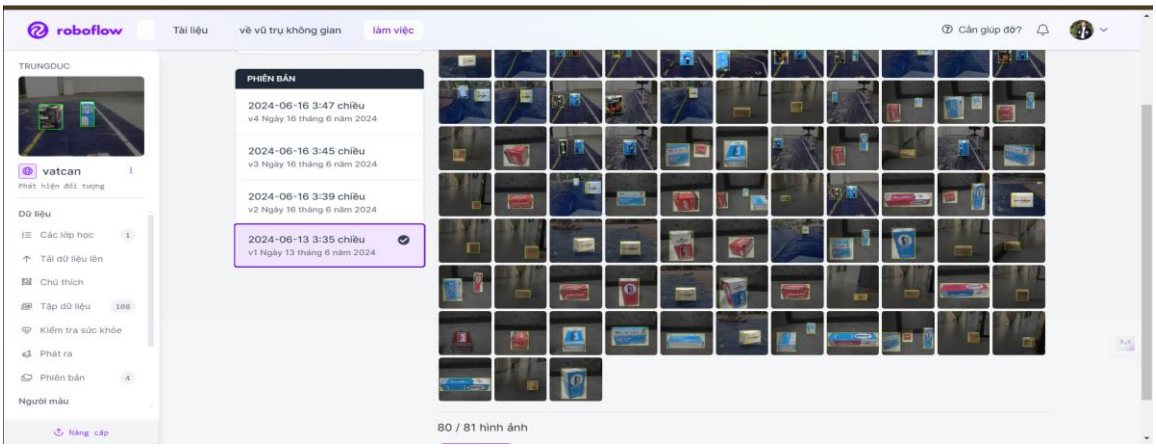


Hình 3.26: Thu thập các vật thể từ môi trường thực tế

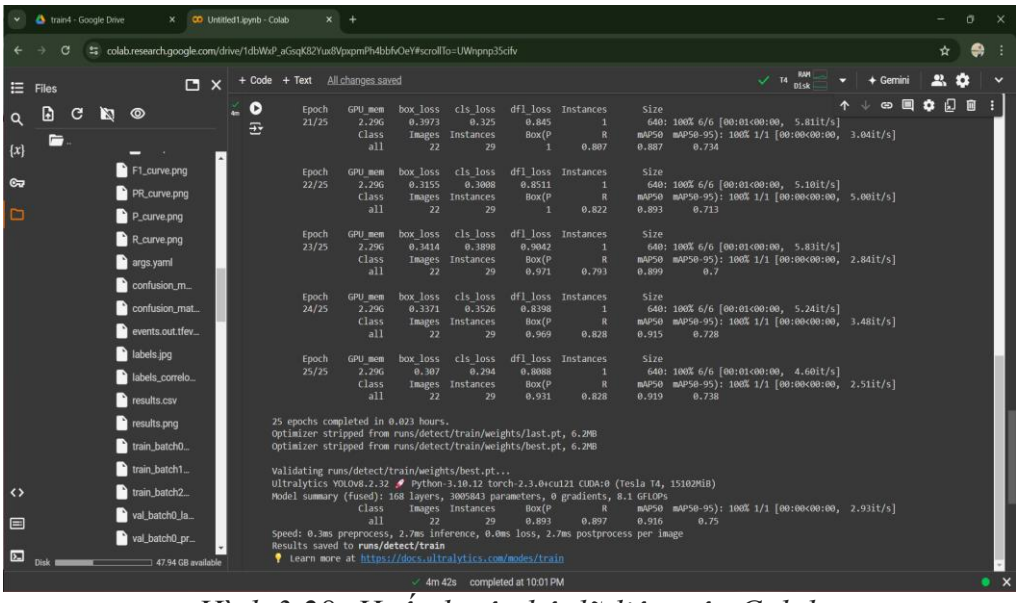
b) Huấn luyện cho bộ dữ liệu

Từ hình ảnh thu thập được từ môi trường thực tế tiến hành huấn luyện cho bộ dữ liệu:

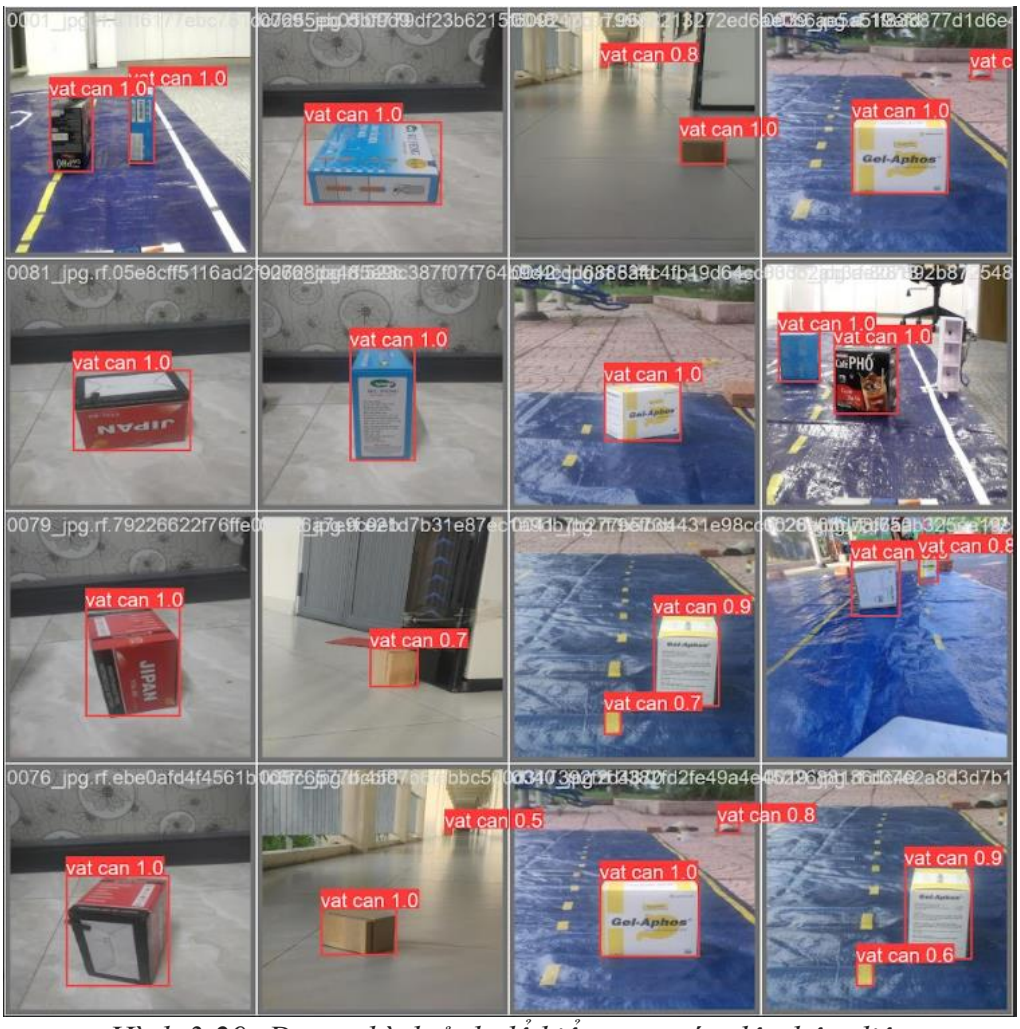
- Mỗi hình ảnh được sẽ được đánh nhãn cho từng ảnh
- Sử dụng mô hình YOLOv8 tiến hành huấn luyện cho bộ data trên Colab
- Đánh giá mô hình



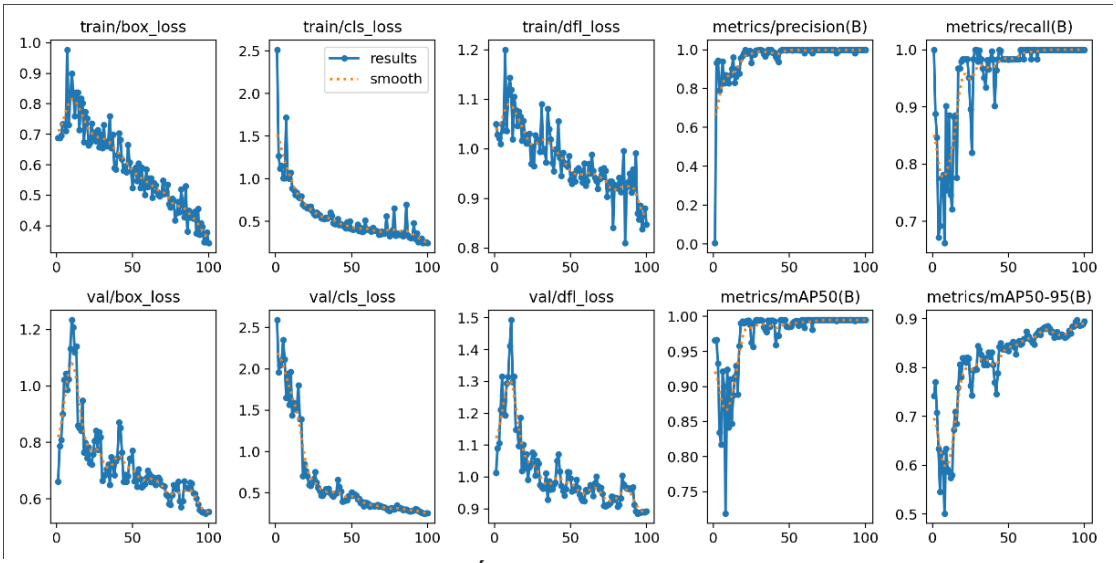
Hình 3.27: Sử dụng Roboflow để đánh nhãn cho dữ liệu đầu vào



Hình 3.28: Huấn luyện bộ dữ liệu trên Colab



Hình 3.29: Detect hình ảnh để kiểm tra mức độ nhận diện



Hình 3.30: Kết quả đánh giá nhận diện

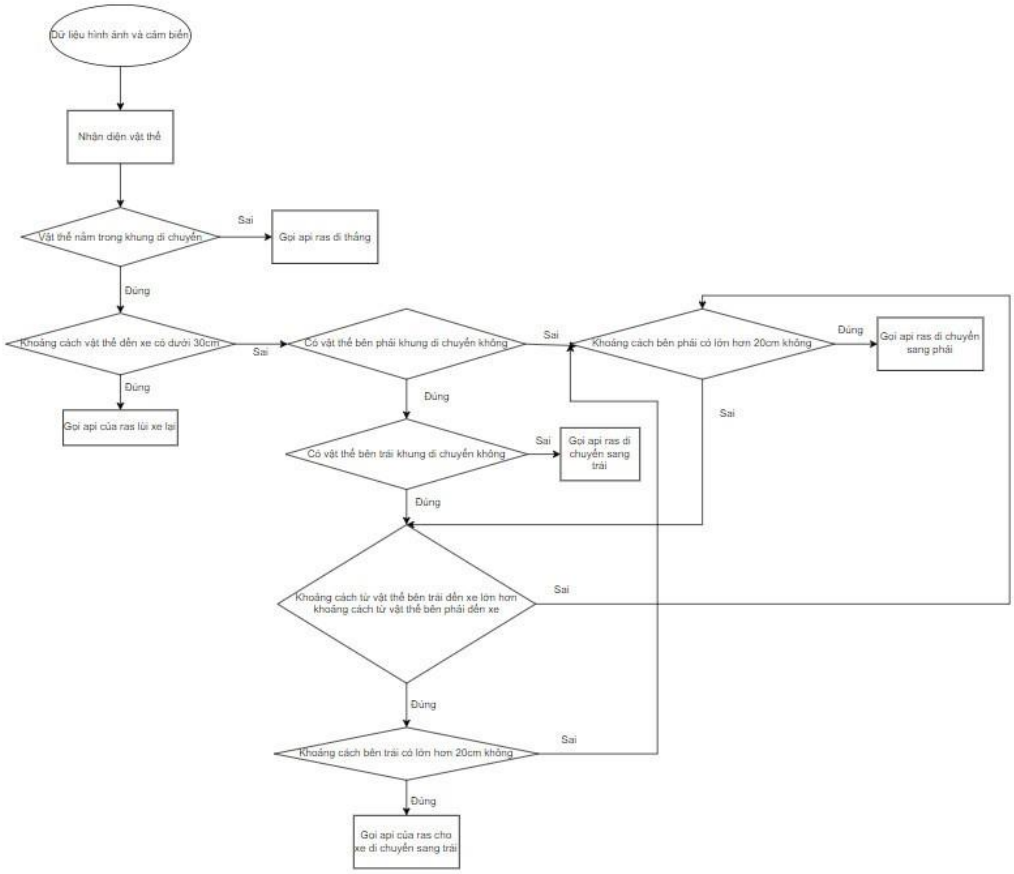
Dựa vào các kết quả thu được cho ta thấy được mức độ nhận diện tương đối tốt. Chúng em đã thử nghiệm ở nhiều môi trường khác nhau và đã cho kết quả nhận diện tốt. Mô hình sẽ cần được thông minh và nhận diện tốt hơn nếu có thêm nhiều hình ảnh đầu vào giúp mô hình nhận diện ngày càng chính xác.

Sau khi huấn luyện mô hình với 100 Epoch cho kết quả huấn luyện rất tốt. Kết quả huấn luyện cho thấy :

- **mAP50 = 0.995(~99.5%)** điều này cho ta thấy mô hình rất chính xác khi sử dụng với ngưỡng IoU là 0.5
- **mAP50-95 (IoU từ 0.50 đến 0.95)** là 89.5%, điều này cho thấy mô hình vẫn duy trì hiệu suất cao trên các ngưỡng IoU khác nhau, mặc dù có giảm chút ít khi ngưỡng IoU tăng lên.

c) Triển khai nhận diện cho xe

Để triển khai cho xe nhận diện các vật thể cũng như làn đường chúng em cần phải xử lý hình ảnh đầu vào. Tại đây chúng em lấy hình ảnh thu được từ camera Imou thông qua giao thức RTSP (Real time streaming protocol) với fps = 30. Sau khi thử nghiệm thực tế thì việc thông qua 1 đường truyền RSTP đem lại hình ảnh thu được khá chậm và không được realtime. Chúng em đã xử lý hình ảnh bằng cách sử dụng giao thức RTSP để biến camera Imou có khả năng xử lý ảnh như Webcam của máy tính khi đó hình ảnh thu được sẽ được cải thiện realtime nhất với độ trễ chỉ 0,1s.



Hình 3.31: Sơ đồ thuật toán ra quyết định tránh vật cản cho xe

- Sau khi đọc dữ liệu hình ảnh từ camera vào, cài đặt kích thước cho khung hình và đưa hình ảnh vào model để nhận diện vật cản

```
# Đọc dữ liệu từ camera và xử lý hình ảnh
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
    # Nhận diện vật thể
    frame = cv2.resize(frame,  dsize: (int(width), int(height)))
    results = model.predict(source=frame, stream=True, save=False, imgsz=320, conf=0.25)
```

- Lấy giá trị tọa độ, nhãn vật cản nhận diện được sau khi qua model

```
        for box in boxes:
            x1, y1, x2, y2 = box.xyxy[0].tolist()
            conf = box.conf[0].item()
            cls = box.cls[0].item()
            class_name = model.names[int(cls)]
```

- Nếu vật cản nằm trong khung di chuyển thì xác định vị trí vật cản so với khung di chuyển và tính khoảng cách từ vật cản đến xe

```
# Xử lý hình ảnh nhận diện được nằm trong phạm vi tính toán di chuyển
if y2 > 280:
    # Xác định vị trí của vật cản so với khung di chuyển
    line_start_left = (150, 600)
    line_end_left = (330, 280)
    line_start_right = (650, 600)
    line_end_right = (470, 280)

    point1 = (x1, y2)
    point2 = (x2, y2)

    sideleft1 = point_side_of_line(line_start_left, line_end_left, point1)
    sideleft2 = point_side_of_line(line_start_left, line_end_left, point2)
    siderright1 = point_side_of_line(line_start_right, line_end_right, point1)
    siderright2 = point_side_of_line(line_start_right, line_end_right, point2)

    # Khoảng cách từ xe đến các vật cản
    if sideleft2 >= 0:
        if maxindexleft < y2:
            maxindexleft = y2

    elif siderright1 <= 0:
        if maxindexright < y2:
            maxindexright = y2

    else:
```

```
     else:
         between = True

         if sideleft1 >= 0:
             if maxindexleft < y2:
                 maxindexleft = y2

         if siderright2 <= 0:
             if maxindexright < y2:
                 maxindexright = y2

         if maxindexbetween < y2:
             maxindexbetween = y2

 cv2.putText(frame, text: f"{class_name} {conf:.2f}", org: (int(x1 + 20), int(y1) + 20),
             cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5,
             color: (0, 0, 255), thickness: 2)
 cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 0, 255), 2)
```

- Lấy giá trị cảm biến và kiểm tra nếu vật cản nằm trong khung di chuyển và nằm trong vùng nguy hiểm thì lùi xe lại

```
Distance()

if between and maxindexbetween > 330:

    # Vật cản nằm trong vùng nguy hiểm
    if maxindexbetween > 440:
        color = (0, 0, 255)
        colorx = (0, 0, 255)

        if behind_Distance > 20:
            # Lùi về khoảng cách an toàn
            runbackward()
            checkright = True
            checkforward = True
            checkleft = True
            checkstop = True
            colorx = (0, 255, 255)

            if behind_Distance > 30:
                colorx = (0, 255, 0)

        else:
            stop()
```

- Vật cản nằm trong khung di chuyển thì tính toán hướng di chuyển né vật cản cho xe

```
# Vật cản nằm trong khung di chuyển
elif maxindexbetween > 330:

    color = (0, 255, 255)
    # Kiểm tra việc di chuyển sang trái
    if ((maxindexleft < maxindexright and checkrunleft) or checkrunright is False) and onrunleft:
        colorx = (0, 0, 255)
        if left_Distance > 20:
            # Di chuyển sang trái
            runleft()
            onrunright = False
            checkright = True
            checkforward = True
            checkbackward = True
            checkstop = True
            colorx = (0, 255, 255)
            if left_Distance > 30:
                colorx = (0, 255, 0)

        else:
            stop()
            checkrunleft = False
            onrunright = True

        cv2.putText(frame, text: f"left", org: (370, 580),
                    cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5,
                    colorx, thickness: 2)
```

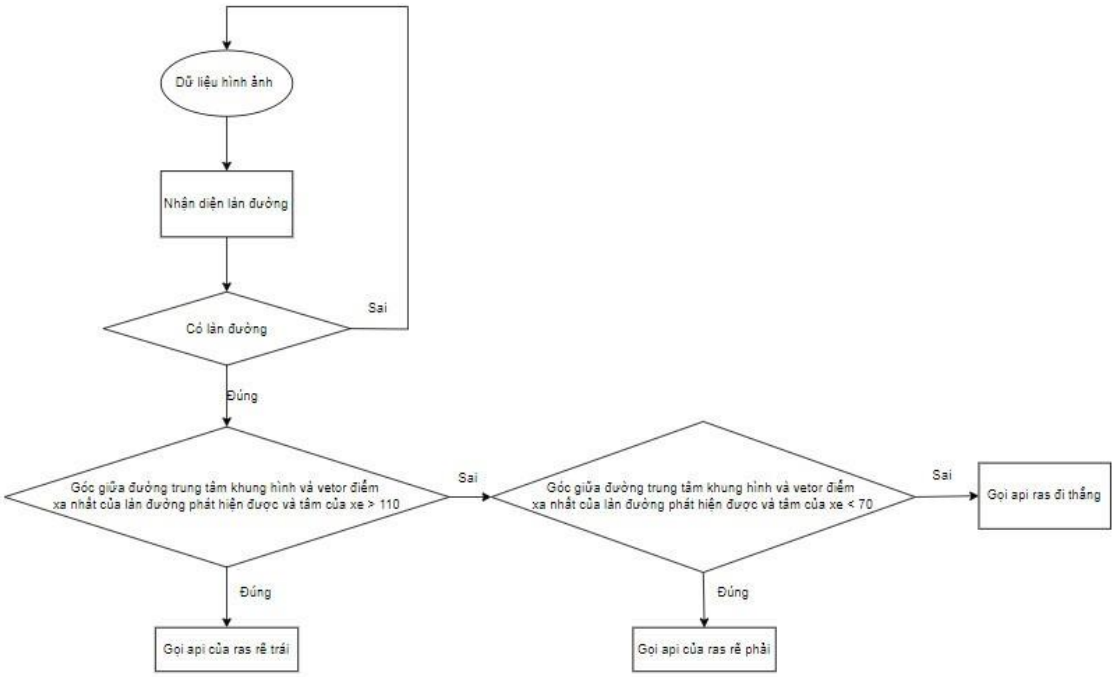
```
# Kiểm tra việc di chuyển sang phải
elif checkrunright and onrunright:
    colorx = (0, 0, 255)
    if right_Distance > 20:
        runright()
        onrunleft = False
        checkleft = True
        checkforward = True
        checkbackward = True
        checkstop = True
        colorx = (0, 255, 255)

        if right_Distance > 30:
            colorx = (0, 255, 0)

    else:
        stop()
        checkrunright = False
        onrunleft = True
    cv2.putText(frame, text: f"right", org: (370, 580),
                cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5,
                colorx, thickness: 2)
```

- Nếu vật cản nằm ngoài khung di chuyển thì xe di chuyển thẳng

```
# Không có vật cản trong khung di chuyển
else:

    color = (0, 255, 0)
    colorx = (0, 0, 255)
    # Di chuyển tiến
    if front_Distance > 20:
        runforward()
        onrunleft = True
        onrunright = True
        checkleft = True
        checkright = True
        checkbackward = True
        checkstop = True
        checkrunleft = True
        checkrunright = True
        colorx = (0, 255, 255)

        if front_Distance > 30:
            colorx = (0, 255, 0)

    else:
        stop()

    cv2.putText(frame, text: f"forward", org: (370, 580),
                cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5,
                colorx, thickness: 2)
```

- Xác định khoảng cách cho 4 hướng và vẽ mũi tên kí hiệu

```
# Hiển thị mũi tên xác định khoảng cách
colorline = (0, 255, 0)
if 10 < behind_Distance < 30:
    colorline = (0, 255, 255)
elif behind_Distance <= 10:
    colorline = (0, 0, 255)

cv2.arrowedLine(frame, pt1: (400, 500), pt2: (400, 550), colorline, thickness: 2, tipLength=0.2)
cv2.putText(frame, text: f"{behind_Distance}", org: (403, 525),
            cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5,
            colorline, thickness: 2)

colorline = (0, 255, 0)
if 10 < front_Distance < 30:
    colorline = (0, 255, 255)
elif front_Distance <= 10:
    colorline = (0, 0, 255)

cv2.arrowedLine(frame, pt1: (400, 500), pt2: (400, 450), colorline, thickness: 2, tipLength=0.2)
cv2.putText(frame, text: f"{front_Distance}", org: (403, 475),
            cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5,
            colorline, thickness: 2)

colorline = (0, 255, 0)
if 10 < left_Distance < 30:
    colorline = (0, 255, 255)
elif left_Distance <= 10:
    colorline = (0, 0, 255)


cv2.arrowedLine(frame, pt1: (400, 500), pt2: (350, 500), colorline, thickness: 2, tipLength=0.2)
cv2.putText(frame, text: f"{left_Distance}", org: (360, 495),
            cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5,
            colorline, thickness: 2)

colorline = (0, 255, 0)
if 10 < right_Distance < 30:
    colorline = (0, 255, 255)
elif right_Distance <= 10:
    colorline = (0, 0, 255)

cv2.arrowedLine(frame, pt1: (400, 500), pt2: (450, 500), colorline, thickness: 2, tipLength=0.2)
cv2.putText(frame, text: f"{right_Distance}", org: (410, 495),
            cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5,
            colorline, thickness: 2)

# Khung di chuyển của xe
cv2.line(frame, pt1: (300, 336), pt2: (500, 336), color, thickness: 2)
cv2.line(frame, pt1: (240, 442), pt2: (560, 442), color, thickness: 2)
cv2.polylines(frame, pts: [pts], isClosed: True, color, thickness: 2)
```



Hình 3.32: Sơ đồ ra quyết định di chuyển theo làn cho xe

- Đọc hình ảnh vào và giảm bớt số lượng khung ảnh tính toán trên 1s

```
# Đọc hình ảnh từ camera
while True:
    ret, frame = cap.read()
    if not ret:
        break
    # Chỉ tính khung hình này để giảm thời gian chạy
    if frame_count % (skip_frames + 1) == 0:
        output_img, lanes_points, lanes_detected, cfg, draw_points = lane_detector.detect_lanes(frame)
```

Lấy những điểm làn giữa phát hiện được và tính góc giữa 2 vector là đường thẳng chính giữa khung hình và đường thẳng từ điểm làn xa nhất phát hiện được đến điểm trung tâm của xe góc dưới hình ảnh

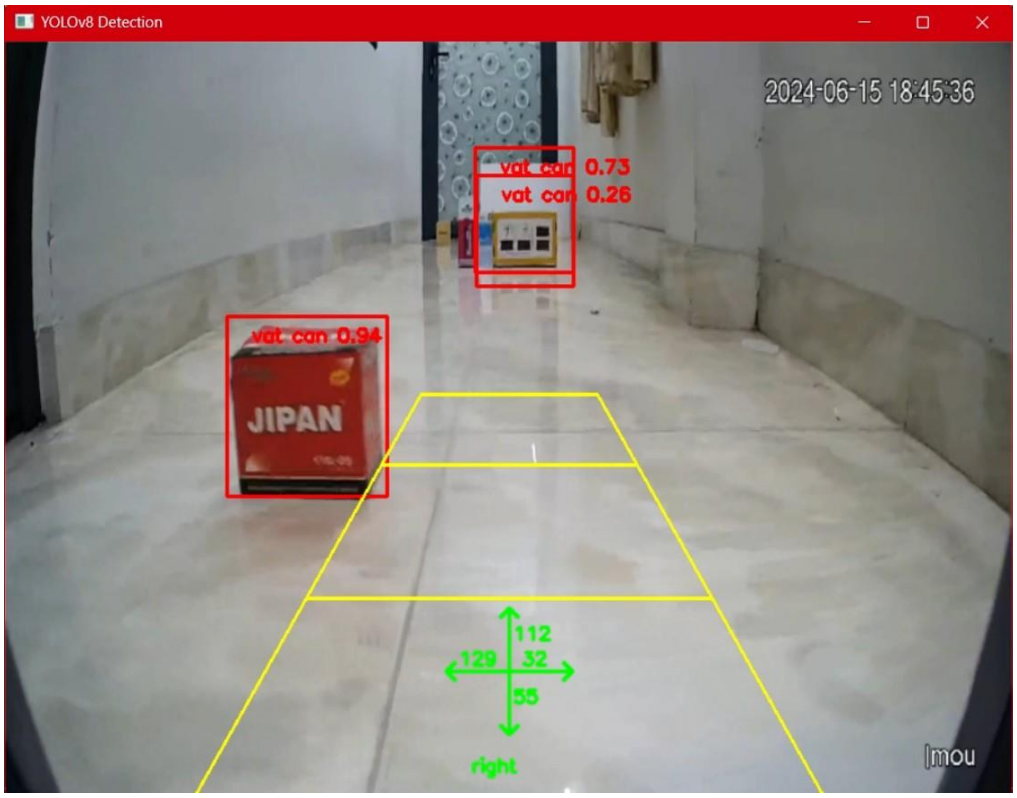
- Nếu góc trong khoảng 70-110 độ thì xe di chuyển thẳng
- Nếu góc nhỏ hơn 70 độ thì xe rẽ phải
- Nếu góc lớn hơn 110 độ thì rẽ trái

```
for lane_num, lane_points in enumerate(lanes_points):
    # Lấy những điểm làn chính giữa
    if lane_num == 1:
        # Lấy điểm làn xa nhất phát hiện được
        max_y_point = min(lane_points, key=lambda point: point[1])

        # Tính toán góc giữa 2 vector
        vector1 = np.array([max_y_point[0] - width // 2, max_y_point[1] - height])
        vector2 = np.array([width - width // 2, height - height])

        unit_vector1 = vector1 / np.linalg.norm(vector1)
        unit_vector2 = vector2 / np.linalg.norm(vector2)
        dot_product = np.dot(unit_vector1, unit_vector2)
        angle = np.arccos(dot_product)
        angle_degrees = np.degrees(angle)

        cv2.putText(output_img, text: f'Angle: {angle_degrees:.2f} degrees', org: (50, 50), cv2.FONT_HERSHEY_SIMPLEX,
        # Đi thẳng nếu góc tính được trong khoảng 70 -> 110 độ
        if int(angle_degrees) > 70 and int(angle_degrees) < 110:
            cv2.putText(output_img, text: f'forward', org: (width // 2, height - 30), cv2.FONT_HERSHEY_SIMPLEX,
                        fontScale: 1, color: (0, 255, 0), thickness: 2)
            runforward()
            checkleft = True
            checkright = True

        # Rẽ phải nếu góc tính được <= 70 độ
        elif int(angle_degrees) <= 70:
    # Rẽ phải nếu góc tính được <= 70 độ
    elif int(angle_degrees) <= 70:
        cv2.putText(output_img, text: f'right', org: (width // 2, height - 30), cv2.FONT_HERSHEY_SIMPLEX,
                    fontScale: 1, color: (0, 255, 0), thickness: 2)
        runright()
        checkleft = True
        checkforward = True
    # Rẽ trái nếu góc tính được >= 110 độ
    else:
        cv2.putText(output_img, text: f'left', org: (width // 2, height - 30), cv2.FONT_HERSHEY_SIMPLEX,
                    fontScale: 1, color: (0, 255, 0), thickness: 2)
        runleft()
        checkforward = True
        checkright = True
```

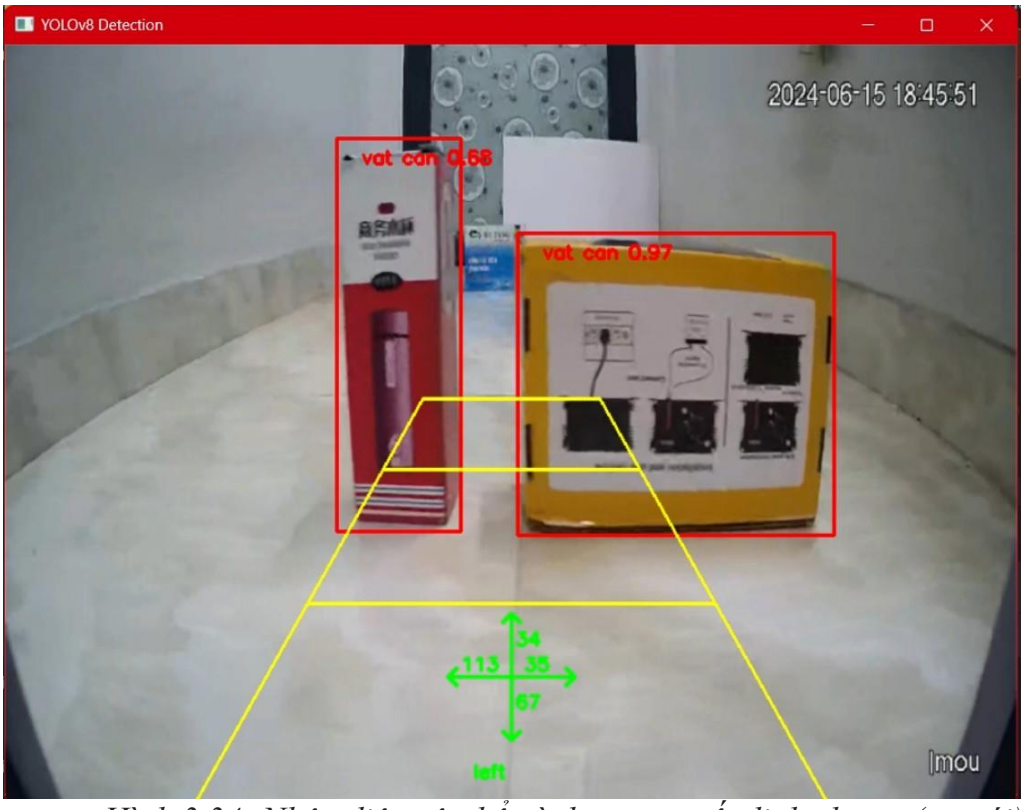
3.3. Kết quả

3.3.1. Nhận diện vật cản cho xe và đưa ra quyết định

Kết quả sử dụng mô hình Yolov8 cho thấy thuật toán hoạt động 1 cách hiệu quả và có độ chính xác tương đối lớn khi có thể phát hiện vật thể.Dựa vào mô hình nhận diện và cảm biến, vật thể được phát hiện và được xử lý qua các thuật toán hướng đi giúp xe có khả năng tự quyết định hướng đi và tính toán được khoảng cách đường đi tốt nhất cho xe. Dưới đây là kết quả thực tế cho mô hình này.



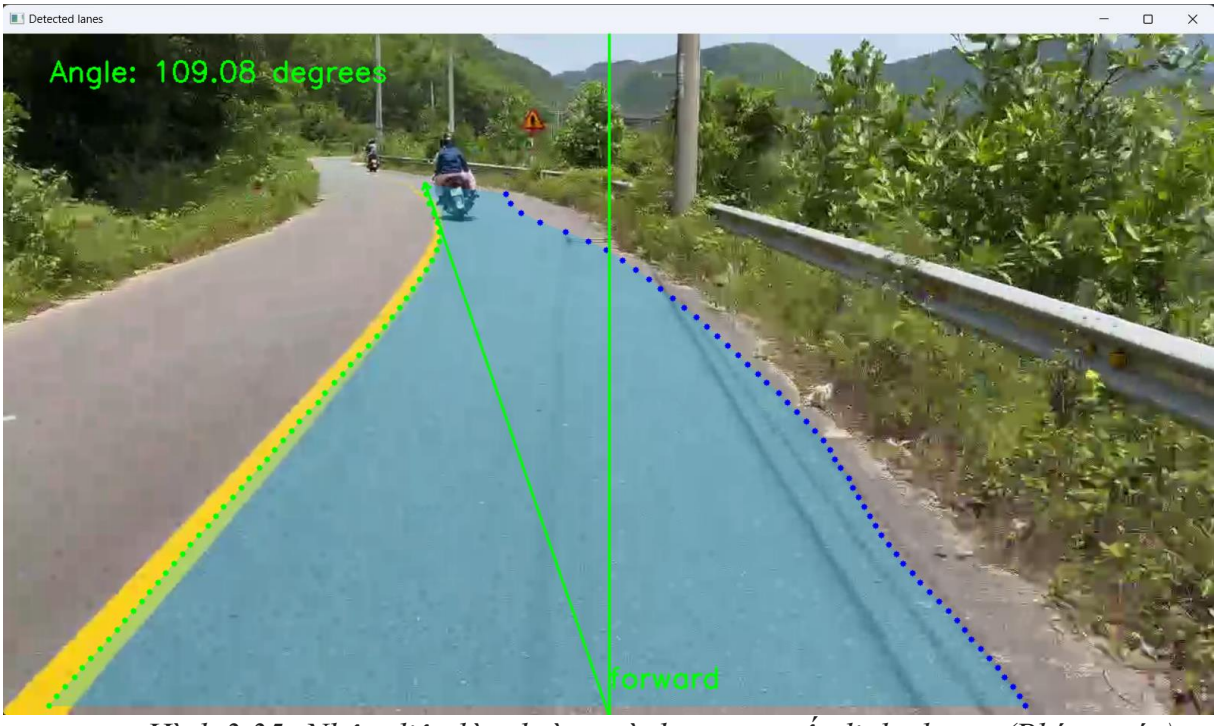
Hình 3.33: Nhận diện vật thể và đưa ra quyết định cho xe (xe đi phải)



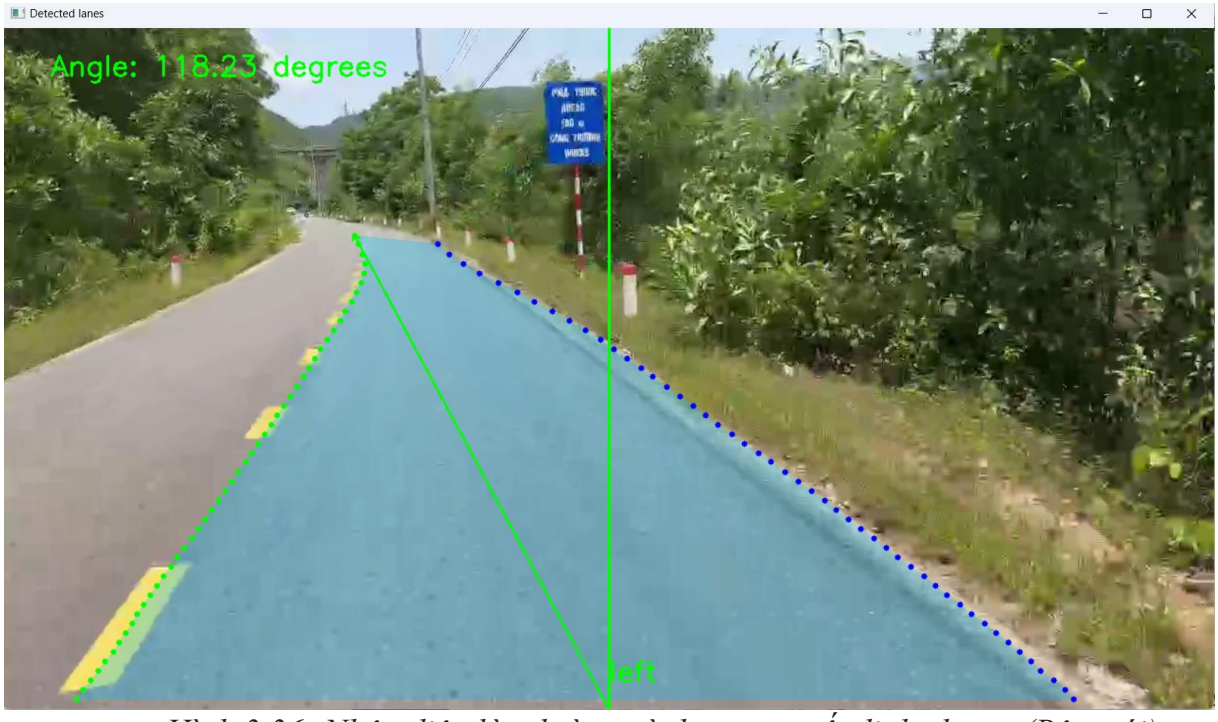
Hình 3.34: Nhận diện vật thể và đưa ra quyết định cho xe (xe trái)

3.3.2. Nhận diện làn đường

Kết quả sử dụng mô hình Ultra Fast Lane Detection cho ta thấy được hình ảnh làn đường được bắt khá tốt. Việc triển khai nhận diện làn cho dự án này cần phải phát triển thêm đặc biệt là cải thiện được hệ thống camera tốt hơn và setup cho mô hình thực tế hơn. Giới hạn cho việc triển khai lần này là mô hình khá nhỏ chưa thể nhận diện được 1 cách thực tế nhất. Tuy nhiên mô hình triển khai được detect cho kết quả khá tốt: Dưới đây là kết quả nhận diện làn đường và đưa ra kết quả quyết định hướng đi cho xe:



Hình 3.35: Nhận diện làn đường và đưa ra quyết định cho xe (Phía trước)



Hình 3.36: Nhận diện làn đường và đưa ra quyết định cho xe (Bên trái)

3.3.3. Mô hình xe tự hành

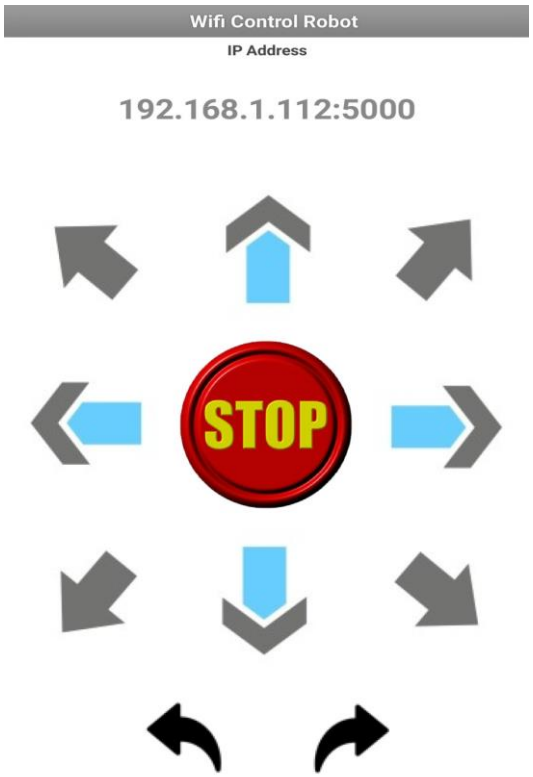
Sau khi thực nghiệm mô hình thì chúng em đã thiết kế mô hình của xe 1 cách gọn gàng và hợp lý nhất. Xe được thiết kế khung bao bộc giúp xe trông thích mắt và gọn gàng nhất.Đây là hình ảnh thực tế của xe:

Hình 3.37: Mô hình trực quan xe tự hành (Phía trước)

Hình 3.40: Mô hình trực quan xe tự hành (Bên phải)

3.3.4. Xây dựng App điều khiển xe

Chúng em xây dựng App điều khiển xe qua Wifi thông qua việc gọi tới địa chỉ IP. App sẽ gửi tới các Api gọi tới các nút Button "Tiến,Lùi,Trái,Phải,Ngang,Chéo và Xoay". Giúp chúng tôi có thể xử lý ở trong những trường hợp khẩn cấp như mất kết nối hay hệ thống AI đưa ra kết quả không chính xác do môi trường chưa thuận lợi.



Hình 3.41: App điều khiển xe tự hành

3.3.5. Kết quả thuật toán ra quyết định

Với việc sử dụng mô hình Yolov8 nhận diện vật cản và Ultra Fast Lane Detection nhận diện làn đường và thuật toán xử lý để ra quyết định cho xe. Ta có được kết quả tính toán tương đối chính xác. Hình ảnh được xử lý nhanh và cho ra quyết định tốt khi phát hiện vật thể.

Dưới đây là thống kế thời gian xử lý trung bình trên hệ thống:

Xử lý	Thời gian (s)
Thời gian khởi động chương trình(server)	5.5
Xác định vật cản	0.05
Ra quyết định	0.01
Độ trễ hình ảnh của camera	0.2

KẾT LUẬN

1. KẾT QUẢ ĐẠT ĐƯỢC

Trong thời gian tìm hiểu,nghiên cứu cơ sở lý thuyết và triển khai dự án, dự án đã được được những kết quả sau:

- Về mặt lý thuyết: Chúng em đã nắm được kiến thức cơ bản để xây dựng 1 hệ thống IOT bằng cách sử dụng các kỹ thuật về cơ khí, hệ thống nhúng, xử lý hình ảnh và các thuật toán nhận diện AI.
- Về mặt ứng dụng: Chúng em đã xây dựng và triển khai thực tế thành công mô hình xe tự hành đa hướng
- Xe hoạt động ổn trong môi trường thực tế
- Hệ thống đã xử lý được các bài toán ra quyết định cho hướng đi của xe trong nhiều trường hợp thực tế.
- Dự án đã giải quyết được các mục tiêu đề ra và cơ bản hoàn thành đúng như hệ thống được thiết kế, tiến độ đặt ra ban đầu.

2. NHỮNG VẤN ĐỀ CHƯA GIẢI QUYẾT

Ngoài những mặt đã đạt được, dự án vẫn còn 1 số nhược điểm cần cải thiện như:

- Mô hình hoạt động chưa được trơn tru do hệ thống động cơ còn thô sơ và chưa được đảm bảo
- Cần tăng khả năng nhận diện cho model vì khả năng nhận diện của xe đang bị giới hạn bởi các vật thế cố định
- Dự án cần cải tiến mô hình dự đoán 1 cách chính xác và giải quyết được nhiều bài toán thông thực tế

3. HƯỚNG PHÁT TRIỂN

Với những mặt hạn chế đã kết luận, dự án cần có những cải thiện trong tương lai để hoàn thiện một cách tốt nhất:

- Cần nâng cấp các trang thiết bị phần cứng để đảm bảo cho hệ thống được hoạt động một cách tốt nhất.
- Tích hợp các công nghệ khác có khả năng điều khiển xe bằng giọng nói
- Cải thiện thuật toán đường đi của xe, tích hợp chức năng xe đi theo bản đồ

- Cải thiện các thuật toán về nhận diện môi trường và bổ sung thêm nhận diện biển báo

TÀI LIỆU THAM KHẢO

[1] Yolo: YOLO You Only Look Once https://phamdinhkhanh.github.io/2020/03/09/DarknetAlgorithm.html

[2] Structures of the Omnidirectional Robots with Swedish Wheels - Mihai Olimpiu Tătar*m Cătălin Popovici, Dan Mândru*, Alin Pleşa*, *Department of Mechatronics and Machine Dynamics Technical University of Cluj-Napoca Cluj-Napoca, România, Ioan Ardelean*,March 2013 Diffusion and Defect Data Pt.B: Solid State Phenomena 198:132-137 DOI: 10.4028/www.scientific.net/SSP.198.132

[3] https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[4] P. Viboonchaicheep, A. Shimada and Y. Kosaka, "Position rectification control for Mecanum wheeled omni-directional vehicles," IECON'03. 29th Annual Conference of the IEEE Industrial Electronics Society (IEEE Cat. No.03CH37468), Roanoke, VA, USA, 2003, pp. 854-859 vol.1, doi: 10.1109/IECON.2003.128009

[5] Fully Convolutional Neural Networks for Road Detection with Multiple Cues Integration, Xiaofeng Han, Jianfeng Lu, Chunxia Zhao, Hongdong li May 2018, DOI: 10.1109/ICRA.2018.8460663 Conference: 2018 IEEE International Conference on Robotics and Automation (ICRA)

[6] A Simple and Efficient Multi-task Network for 3D Object Detection and Road Understanding 6 Mar 2021 · Di Feng, Yiyang Zhou, Chenfeng Xu, Masayoshi Tomizuka, Wei Zhan

[7] https://pytorch.org/docs/stable/index.html

[8] https://www.raspberrypi.com/documentation/

[9] https://code.visualstudio.com/docs/remote/ssh

[10] https://docs.nvidia.com/#all-documents