

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

PHẠM VĂN NHI

LUẬN VĂN TỐT NGHIỆP
XE TỰ HÀNH SỬ DỤNG THỊ GIÁC MÁY TÍNH

KỸ SƯ NGÀNH KỸ THUẬT ĐIỀU KHIỂN & TỰ ĐỘNG HÓA

TP. HỒ CHÍ MINH, 2017

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

PHẠM VĂN NHI – 41302791

LUẬN VĂN TỐT NGHIỆP
XE TỰ HÀNH SỬ DỤNG THỊ GIÁC MÁY TÍNH

KỸ SƯ NGÀNH KỸ THUẬT ĐIỀU KHIỂN & TỰ ĐỘNG HÓA

GIẢNG VIÊN HƯỚNG DẪN
BÙI THANH HUYỀN

TP. HỒ CHÍ MINH, 2017

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

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

TP. HCM, ngày....tháng.....năm 2017

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

Tên luận văn:

XE TỰ HÀNH SỬ DỤNG THỊ GIÁC MÁY TÍNH

Nhóm Sinh viên thực hiện:

Phạm Văn Nhi

Cán bộ hướng dẫn:

41302791 Bùi Thanh Huyền

Đánh giá Luận văn

1. Về cuốn báo cáo:

Số trang _____ Số chương _____

Số bảng số liệu _____ Số hình vẽ _____

Số tài liệu tham khảo _____ Sản phẩm _____

Một số nhận xét về hình thức cuốn báo cáo:

2. Về nội dung luận văn:

3. Về tính ứng dụng:

4. Về thái độ làm việc của sinh viên:

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

Điểm từng sinh viên:

Phạm Văn Nhi:...../10

Người nhận xét

(Ký tên và ghi rõ họ tên)

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

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

TP. HCM, ngày....tháng.....năm 2017

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

Tên luận văn:

XE TỰ HÀNH SỬ DỤNG THỊ GIÁC MÁY TÍNH

Nhóm Sinh viên thực hiện:

Phạm Văn Nhi

Cán bộ phản biện:

41302791

Đánh giá Luận văn

5. Về cuốn báo cáo:

Số trang	_____	Số chương	_____
Số bảng số liệu	_____	Số hình vẽ	_____
Số tài liệu tham khảo	_____	Sản phẩm	_____

Một số nhận xét về hình thức cuốn báo cáo:

6. Về nội dung luận văn:

7. Về tính ứng dụng:

8. Về thái độ làm việc của sinh viên:

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

Điểm từng sinh viên:

Phạm Văn Nhi:...../10

Người nhận xét

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Xin gửi lời cảm ơn sâu sắc đến cô hướng dẫn Bùi Thanh Huyền đã gợi ý đề tài cũng như hướng dẫn tận tình để giúp tôi hoàn thành luận văn. Cảm ơn các thầy cô trong bộ môn Tự Động nói riêng và Khoa Điện – Điện Tử nói chung đã tạo điều kiện giúp đỡ và giảng dạy nhiệt tình trong suốt năm học qua.

Xin gửi lời cảm ơn đến các anh chị, các bạn bè trong bộ môn Tự Động đã giúp đỡ và động viên tôi trong suốt quá trình học tập cũng như thực hiện luận văn.

Đặc biệt xin cảm ơn đến gia đình đã luôn hỗ trợ và động viên giúp tôi có động lực để phấn đấu trong học tập và cuộc sống.

TP. Hồ Chí Minh, tháng 12 năm 2017

TP. HCM, ngày....tháng.....năm 2017

ĐỀ CƯƠNG CHI TIẾT

TÊN ĐỀ TÀI: XE TỰ HÀNH SỬ DỤNG THỊ GIÁC MÁY TÍNH

Cán bộ hướng dẫn: Bùi Thanh Huyền

Thời gian thực hiện: Từ ngày...08/09/2017...đến ngày...17/12/2017...

Sinh viên thực hiện:

Phạm Văn Nhi - 41302791

Nội dung đề tài:

- Tìm hiểu các mô hình xe tự hành và các dự án về xe tự hành trong và ngoài nước.
- Nghiên cứu sử dụng và lập trình máy tính nhúng.
- Lập trình xử lý ảnh bằng thư viện hỗ trợ OpenCV.
- Nghiên cứu và ứng dụng thuật toán xác định làn đường bằng xử lý ảnh.
- Xây dựng mô hình xe tự hành.
- Xây dựng phương pháp điều khiển bánh lái của xe tự hành.
- Thủ nghiệm xe tự hành trên làn đường thẳng và làn đường cong.

Kế hoạch thực hiện:

- Lê mục tiêu, nhiệm vụ và tính cấp thiết của luận văn.
- Tìm hiểu và nghiên cứu xe tự hành.
- Nghiên cứu sử dụng và lập trình máy tính nhúng Raspberry Pi.
- Nghiên cứu và lập trình OpenCV Python.
- Thiết kế mô hình xe tự hành.
- Nghiên cứu và lập trình thuật toán xác định làn đường trong xử lý ảnh.
- Lập trình điều khiển bánh lái.
- Tích hợp hệ thống và điều chỉnh thông số thuật toán.
- Kiểm tra và chạy kiểm thử.
- Viết báo cáo.
- Xây dựng PPT.

Xác nhận của Cán bộ hướng dẫn

TP. HCM, ngàythángnăm 2017

Sinh viên

DANH SÁCH HỘI ĐỒNG BẢO VỆ LUẬN VĂN

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

1. – Chủ tịch.
2. – Thư ký.
3. – Ủy viên.

TÓM TẮT LUẬN VĂN

Xe tự hành đang là xu hướng nghiên cứu phát triển và mang tính công nghệ đột phá hiện nay. Do đó, lĩnh vực xe tự hành đang là có rất nhiều thử thách trước mắt và cần nghiên cứu để tạo ra một tích hợp hoàn hảo, một phương tiện tự hành thông minh. Hiện nay, có rất nhiều nghiên cứu về phát triển phần cứng, phần mềm và thuật toán để tạo ra một chiếc xe có khả năng tự hành. Bên cạnh đó, một số hệ thống đã được áp dụng trên một số xe đắt tiền như hệ thống cảnh báo xe lệch khỏi làn đường, hệ thống kiểm soát hành trình, hay hệ thống tự động đỗ xe, v.v...

Vấn đề cốt lõi của xe tự hành là xác định định làn đường, những ký hiệu trên đường và môi trường xung quanh để xe có thể tự hành dựa vào những thông tin đó. Với tính năng nhận diện làn đường, xe thu thập hình ảnh thực tế từ camera và sau đó qua các thuật toán xử lý ảnh thuận tủy. Bên cạnh sự phát triển của máy học (Machine Learning) và AI, trong tương lai sẽ có nhiều nghiên cứu sử dụng Deep Learning vào điều khiển lái xe tự hành.

Đề tài luận văn sẽ xây dựng mô hình bám làn đường của xe tự hành. Yêu cầu đối với hệ thống trong đề tài này là điều khiển xe di chuyển trên làn đường đã được đánh dấu vạch kẻ đường, dựa vào thông tin từ camera với khả năng xác định làn đường của thuật toán. Để hoàn thành đề tài, thuật toán xử lý ảnh xác định làn đường được xử lý tốt và lựa chọn thông số phù hợp cho điều khiển PID vào điều khiển bánh lái. Cuối cùng, đề tài xây dựng mô hình phần cứng nhằm chạy thử nghiệm thuật toán.

MỤC LỤC

Chương 1. Giới thiệu đề tài	2
1.1. Sơ lược về xe tự hành	2
1.2. Một số dự án xe tự hành hiện nay.....	4
1.2.1. Xe tự hành của Google	4
1.2.2. Tesla Autopilot.....	6
1.2.3. Mercedes Autonomous	6
1.3. Phạm vi nghiên cứu.....	7
1.4. Nhiệm vụ luận văn.....	8
1.5. Nội dung của đề tài	9
Chương 2. XÂY DỰNG MÔ HÌNH XE TỰ HÀNH.....	10
2.1. Sơ đồ tổng quát	10
2.2. Mô hình phần cứng xe tự hành	11
2.3. Máy tính nhúng Raspberry Pi 3	12
2.3.1. Giới thiệu máy tính nhúng Raspberry Pi 3	12
2.3.2. Hệ điều hành cho Raspberry Pi 3	13
2.3.3. Sơ đồ chân Raspberry Pi 3.....	14
2.4. Pi Camera NoIR v2.....	14
2.5. Truyền nhận dữ liệu qua UART	18
2.6. Vi điều khiển TM4C123	19
2.7. Lập trình Python.....	20
2.7.1. Giới thiệu về ngôn ngữ Python	20
2.7.2. Tổng quan về kiến trúc và cấu trúc của ngôn ngữ Python	20
2.7.3. Các thư viện.....	22

2.7.3.1. Numpy.....	22
2.7.3.2. Matplotlib	22
2.7.3.3. Picamera.....	23
2.7.3.4. Pyserial	23
2.7.3.5. TkInter	24
Chương 3. THƯ VIỆN OPENCV TRONG XỬ LÝ ẢNH	26
3.1. Tổng quan về bộ thư viện OpenCV.....	26
3.2. Các xử lý cơ bản trong xử lý ảnh	28
3.2.1. Ảnh xám (Gray image).....	28
3.2.2. Bộ lọc nhiễu cho ảnh	29
3.2.2.1. Bộ lọc trung bình	30
3.2.2.2. Làm mờ Gaussian.....	31
3.2.2.3. Làm mờ Median	32
3.2.2.4. Bộ lọc bilateral.....	34
3.2.3. Xác định cạnh trong ảnh	35
3.2.4. Thuật toán Hough.....	38
Chương 4. THUẬT TOÁN XÁC ĐỊNH LÀN ĐƯỜNG	42
4.1. Giới thiệu.....	42
4.2. Tiên xử lý ảnh	44
4.3. Sử dụng thuật toán Hough xác định đường thẳng.....	46
4.4. Phương pháp tìm độ lệch của xe	49
Chương 5. ỨNG DỤNG THUẬT TOÁN PID CHO ĐIỀU KHIỂN BÁNH LÁI.....	51
5.1. Tổng quan về bộ điều khiển PID.....	51
5.2. Ứng dụng bộ điều khiển PID trong vi điều khiển.....	52

5.3. Úng dụng bộ điều khiển PID vào điều khiển bánh lái.....	53
5.4. Phương pháp tìm các hệ số của bộ điều khiển PID	54
Chương 6. KẾT QUẢ THỬ NGHIỆM VÀ ĐÁNH GIÁ.....	57
6.1. Mô hình xe tự hành.....	57
6.2. Vận hành	57
6.3. Kết quả thử nghiệm.....	59
6.3.1. Kết quả xử lý ảnh xác định làn đường.....	60
6.3.2. Kết quả của bộ lọc ngõ của xử lý ảnh	65
6.3.3. Kết quả bộ điều khiển PID	65
Chương 7. KẾT QUẢ ĐẠT ĐƯỢC VÀ HƯỚNG PHÁT TRIỂN ĐỀ TÀI	67
7.1. Kết quả đạt được	67
7.2. Hướng phát triển đề tài	67

DANH MỤC HÌNH VẼ

Hình 1.1 Các mức độ của xe tự hành	3
Hình 1.2 Hình Xe tự hành của Google	5
Hình 1.3 Số liệu thống kê chiều dài xe chạy thử nghiệm.....	5
Hình 1.4 Hình ảnh xe tự hành của Tesla chạy thử nghiệm.....	6
Hình 1.5 Mercedes-Benz S 450 4MATIC.....	7
Hình 2.1 Sơ đồ xe tự hành sử dụng thị giác máy tính	10
Hình 2.2 Sơ đồ hệ thống bên trong xe tự hành của đề tài.....	10
Hình 2.3 Mô hình phần cứng xe tự hành được sử dụng trong đề tài	11
Hình 2.4 Mạch máy tính nhúng Raspberry Pi 3.....	12
Hình 2.5 Giao diện Desktop của hệ điều hành Raspbian	13
Hình 2.6 Sơ đồ chân chi tiết của Raspberry Pi 3	14
Hình 2.7 Hình ảnh Pi Camera thực tế.....	16
Hình 2.8 Camera được gắn với Raspberry qua cáp CSI	17
Hình 2.9 Sơ đồ chân cổng CSI-2 được thiết kế trong Raspberry Pi 3	17
Hình 2.10 Sơ đồ truyền nhận giữa Raspberry và vi điều khiển	18
Hình 2.11 Mạch vi điều khiển TM4C123 được sử dụng trong luận văn.....	19
Hình 2.12 Qui trình thông dịch của ngôn ngữ Python.....	20
Hình 2.13 Thư viện Matplotlib.....	22
Hình 2.14 Giao diện của chương trình mẫu sử dụng thư viện Tkinter	25
Hình 3.1 Các ứng dụng của OpenCV	26
Hình 3.2 Hình ảnh về ứng dụng của OpenCV trong xe tự hành.....	26
Hình 3.3 Ảnh trước và sau khi thực hiện phép chuyển ảnh gray	28
Hình 3.4 Hình ảnh mô tả thực hiện tính chập cho từng điểm ảnh	30
Hình 3.5 Ảnh trước và sau khi làm mờ bằng phương pháp trung bình.....	31
Hình 3.6 Đồ thị phân phối Gaussian với độ lệch chuẩn $\sigma = 1$	32
Hình 3.7 Ảnh trước và sau khi làm mờ bằng gaussian.....	32
Hình 3.8 Cách tính median cho một cửa sổ ảnh	33

Hình 3.9 Ảnh trước và sau khi làm mờ median.....	33
Hình 3.10 Hình Lọc nhiễu bằng bilateral	35
Hình 3.11 Hình ảnh cơ bản về xác định cạnh của ảnh.....	35
Hình 3.12 Ảnh mẫu thử các thuật toán xác định cạnh.....	36
Hình 3.13 Kết quả xác định cạnh của Sobel, Prewitt và DoG.....	36
Hình 3.14 Kết quả xác định cạnh của Canny	37
Hình 3.15 Biểu diễn đường thẳng với ρ và θ	39
Hình 3.16 Phép chuyển đổi một điểm ảnh qua không gian tham số (ρ, θ)	40
Hình 3.17 Thí dụ phép chuyển đổi Hough cho nhiều điểm cùng nằm trên một đường thẳng.....	41
Hình 3.18 Biểu diễn hai đường thẳng trên không gian tham số.....	41
Hình 4.1 Sơ đồ mô hình thuật toán xác định làn đường.....	43
Hình 4.2 Ảnh gray	44
Hình 4.3 Ảnh gray sau khi được làm mờ guassian với kernel 5x5	45
Hình 4.4 Region of Interest (ROI) bao quanh đường biên của làn đường	45
Hình 4.5 Ảnh được xác định cạnh bằng thuật toán Canny	46
Hình 4.6 Các đường thẳng được xác định từ thuật toán Hough	47
Hình 4.7 Phương pháp phân loại đường thẳng biên của làn đường.....	48
Hình 4.8 Tìm giao điểm của hai đường thẳng biên của làn đường.....	49
Hình 4.9 Hình minh họa xe đang lệch đường tâm làn đường một giá trị cte	49
Hình 4.10 Mô tả tính độ lệch của xe.....	50
Hình 5.1 Sơ đồ mô hình thuật toán PID	51
Hình 5.2 Sơ đồ điều khiển bánh lái bằng PID.....	53
Hình 5.3 Quỹ đạo của xe với hiệu chỉnh hệ số Kp.....	55
Hình 5.4 Mô tả bộ điều khiển PD.....	55
Hình 5.5 Bộ điều khiển PD khi có nhiễu.	56
Hình 6.1 Mô hình xe tự hành được xây dựng trong đề tài (top view).....	57
Hình 6.2 Mô hình xe tự hành được xây dựng trong đề tài (front view).	57
Hình 6.3 Mô hình tương tác giữa máy tính PC và xe tự hành khi vận hành	58

Hình 6.4 Màn hình Console kết nối với Raspberry Pi qua SSH.....	58
Hình 6.5 Các làn đường được sử dụng thử nghiệm: (a) làn đường thẳng; (b) làn đường có biên đứt; (c) làn đường cong	59
Hình 6.6 Thử nghiệm xử lý ảnh 1: Làn đường thẳng.....	61
Hình 6.7 Thử nghiệm xử lý ảnh 2: Làn đường cong.....	62
Hình 6.8 Thử nghiệm xử lý ảnh 3: Làn đường có một vạch biên đứt.....	63
Hình 6.9 Thử nghiệm xử lý ảnh 4: Xe có hướng lệch sang trái làn đường	64
Hình 6.10 Đồ thị ngõ ra CTE của xử lý ảnh.....	65
Hình 6.11 Kết quả điều khiển PID khi xe di chuyển trên làn đường thẳng.....	66
Hình 6.12 Kết quả điều khiển PID khi xe di chuyển qua một đoạn đường cong (Hình 6.5c)	66

DANH MỤC BẢNG

Bảng 2.1 So sánh các camera chuẩn CSI-2.....	15
Bảng 2.2 Định dạng dữ liệu truyền giữa Raspberry Pi 3 và VĐK TM4C123.....	18
Bảng 2.3 Một số interface hỗ trợ bởi thư viện Pyserial.....	23
Bảng 4.1 Phương pháp phân loại đường thẳng.....	48
Bảng 5.1 Các thông số của bánh lái.....	54

DANH MỤC TỪ VIẾT TẮT

CTE: Cross Track Error.

LDWS: Lane Departure Warning System.

LKS: Lane Keeping System.

PID: Proportional-Integral-Derivative.

UART: Universal Asynchronous Receiver-Transmitter.

VĐK: Vi điều khiển.

XTH: Xe tự hành.

TÓM TẮT LUẬN VĂN

Xe tự hành đang là xu hướng nghiên cứu phát triển và mang tính công nghệ đột phá hiện nay. Do đó, lĩnh vực xe tự hành đang là có rất nhiều thử thách trước mắt và cần nghiên cứu để tạo ra một tích hợp hoàn hảo, một phương tiện tự hành thông minh. Hiện nay, có rất nhiều nghiên cứu về phát triển phần cứng, phần mềm và thuật toán để tạo ra một chiếc xe có khả năng tự hành. Bên cạnh đó, một số hệ thống đã được áp dụng trên một số xe đắt tiền như hệ thống cảnh báo xe lệch khỏi làn đường, hệ thống kiểm soát hành trình, hay hệ thống tự động đỗ xe, v.v...

Vấn đề cốt lõi của xe tự hành là xác định định làn đường, những ký hiệu trên đường và môi trường xung quanh để xe có thể tự hành dựa vào những thông tin đó. Với tính năng nhận diện làn đường, xe thu thập hình ảnh thực tế từ camera và sau đó qua các thuật toán xử lý ảnh thuận tủy. Bên cạnh sự phát triển của máy học (Machine Learning) và AI, trong tương lai sẽ có nhiều nghiên cứu sử dụng Deep Learning vào điều khiển lái xe tự hành.

Đề tài luận văn sẽ xây dựng mô hình bám làn đường của xe tự hành. Yêu cầu đối với hệ thống trong đề tài này là điều khiển xe di chuyển trên làn đường đã được đánh dấu vạch kẻ đường, dựa vào thông tin từ camera với khả năng xác định làn đường của thuật toán. Để hoàn thành đề tài, thuật toán xử lý ảnh xác định làn đường được xử lý tốt và lựa chọn thông số phù hợp cho điều khiển PID vào điều khiển bánh lái. Cuối cùng, đề tài xây dựng mô hình phần cứng nhằm chạy thử nghiệm thuật toán.

Chương 1. Giới thiệu đề tài

1.1. Sơ lược về xe tự hành

Hệ thống cảnh báo xe lệch làn đường (Lane Departure Warning System – LDWS) trở nên phổ biến rộng rãi. Chúng ta có thể tìm thấy trong các đời xe hiện đại ngày nay. LDWS sẽ đưa cảnh báo khi xe chạy lệch khỏi làn đường, nó cũng được nâng cấp lên thành hệ thống xe bám làn đường (Lane Keeping System – LKS). Thay vì đưa ra cảnh báo cho người lái xe, LKS có thể tác động lên hệ thống lái để điều khiển bánh lái an toàn khi xe di chuyển lệch khỏi làn đường. Bên cạnh đó, xe tự hành, có thể gọi với từ khóa như Autonomous Car, Self-Driving Car hay Autopilot, là xe có khả năng nhận diện môi trường xung quanh và điều khiển lái di chuyển trên đường. Xe tự hành sử dụng rất nhiều công nghệ để nhận dạng được tất cả vật thể hay tình huống xung quanh như radar, đèn laser, GPS, camera xử lý ảnh, v.v... Hiện nay, nó là xu hướng phát triển của công nghệ xe hiện đại mà nhiều công ty lớn trên thế giới đầu tư.

Xe tự hành sẽ là một mô hình phương tiện di động tự động trong tương lai. XTH được đánh giá qua năm mức độ như sau:



Mức độ 0

Xe không có tính năng tự hành nhưng có các hệ thống cảnh báo như hệ thống cảnh báo xe mất lái.



Mức độ 1

Xe có thể tự xử lý trong một vài thời điểm như tự phanh, hỗ trợ đỗ xe, hay hệ thống bám làn đường.



Mức độ 2

Người lái xe có nhiệm vụ quan sát tín hiệu hay phản hồi từ hệ thống tự hành khi nếu hệ thống báo lỗi để có xử lý thích hợp.



Mức độ 3

Trong giới hạn về môi trường làn đường, người lái xe có thể an toàn mà không cần chú ý đến hệ thống lái.



Mức độ 4

Xe có thể hoạt động hoàn toàn tự động, ngoại trừ một số điều kiện môi trường xấu như thời tiết.



Mức độ 5

Chi cần cài đặt điểm đến và điểm bắt đầu, xe hoàn toàn có thể tự hành mà không cần sự can thiệp của con người.

Hình 1.1 Các mức độ của xe tự hành

Lợi ích tiềm năng của xe tự lái bao gồm:

- Tăng tính an toàn, bởi vì máy móc có thể phản ứng nhanh hơn con người.
- Nó sử dụng hiệu quả xăng dầu tốt hơn vì tốc độ, năng lượng sử dụng được điều khiển.
- Giảm tai nạn giao thông.
- Giảm chi phí nhân công.
- Có thể giới hạn được tốc độ của xe cộ và cải thiện được lường giao thông. Từ đó, chi phí cho hạ tầng giao thông giảm.
- Giao thông cá nhân sẽ mở ra tất cả các phân khúc đơn lẻ của xã hội như người già, người quá trẻ hoặc người khuyết tật và việc làm chủ 1 chiếc xe sẽ nằm trong tầm tay của tất cả mọi người.
- Thay đổi về mặt thẩm mỹ và nội thất bên trong

Với khá nhiều ưu điểm nhưng hệ thống này không thực sự hoàn hảo bởi sự phụ thuộc vào điều kiện môi trường. Tính hiệu quả của nó chỉ phát huy tối đa khi xung quanh hội tụ những tiêu chuẩn nhất định.

Tuy nhiên, rào cản lớn nhất không đến từ khía cạnh kỹ thuật mà là luật pháp. Ví dụ như khi tai nạn xảy ra, thật khó xác định lỗi thuộc về người ngồi trên xe hay không.

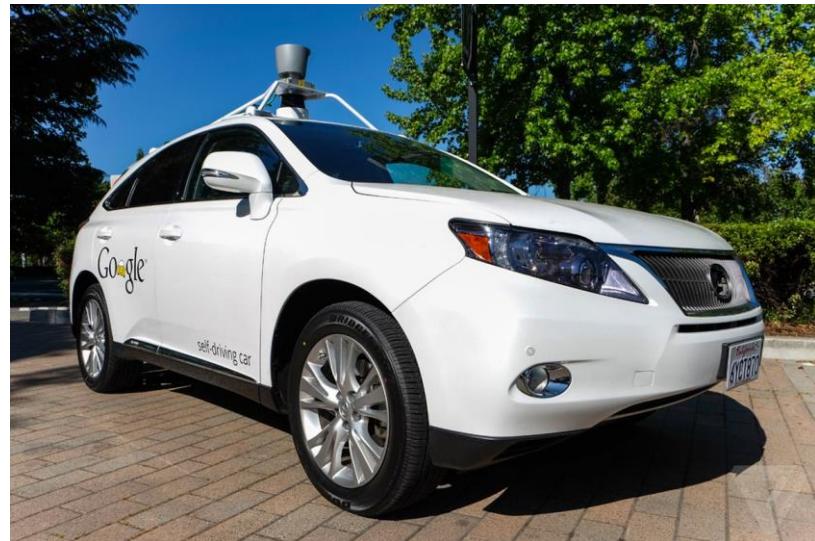
Dù tiêu cực hay tích cực, xe tự hành có thể là xu thế phát triển tất yếu của cuồng quay kỹ thuật tiên tiến.

1.2. Một số dự án xe tự hành hiện nay

XTH mang rất nhiều công nghệ hiện đại và mỗi dự án của các công ty có các tiếp cận phát triển riêng của họ để đạt được công nghệ này. Một số dự án nổi bật như sau:

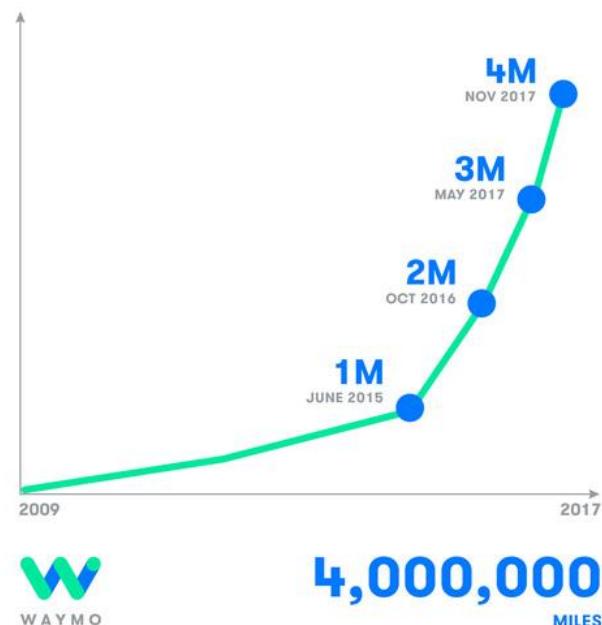
1.2.1. Xe tự hành của Google

Dự án xe tự hành Google bắt đầu vào năm 2009 dưới sự điều hành của công ty Google. Từ 2016 đến nay, dự án này được quản lý bởi công ty Alphabet (công ty mẹ của Google) và thành lập công ty Waymo.



Hình 1.2 Hình Xe tự hành của Google

Xe tự hành Google được chạy thử nghiệm trên nhiều đoạn đường ở Mỹ như Mountain View, CA; Austin, TX; Kirkland, WA; và Metro Phoenix, AZ. Xe được thử nghiệm với nhiều điều kiện môi trường khác nhau.



Hình 1.3 Số liệu thống kê chiều dài xe chạy thử nghiệm
Từ năm 2009 đến nay xe tự hành Google chạy thử nghiệm hơn 4 triệu
đặm chiều dài.

1.2.2. Tesla Autopilot

Tesla Autopilot là một tính năng hỗ trợ lái xe được cung cấp bởi *Tesla*. Ý định của công ty là cung cấp đầy đủ các tính năng trong xe tự hành tương lai. Kế hoạch của *Tesla* sẽ thử nghiệm xe tự hành vào cuối 2017 và cho phép nó hoạt động trước 2019.



Hình 1.4 Hình ảnh xe tự hành của *Tesla* chạy thử nghiệm
Đầu 2016, *Tesla Autopilot* được thử nghiệm hoạt động thật sự trên phần cứng phiên bản 1 (Hardware 1 - 2014) của họ, với chiều dài chạy nghiệm khoảng 500 triệu km. Phần cứng 1 được trang bị camera, radar và cảm biến vị trí siêu âm với nhiều tính năng nổi bật cho xe tự hành như nhận dạng làn đường, biển báo giao thông, nhận dạng ký hiệu đường và xác định vật cản.

Tháng 2/2017, phần mềm *Autopilot* của *Tesla* được thử nghiệm trên phần cứng phiên bản 2.

1.2.3. Mercedes Autonomous

Dự án *Mercedes autonomous* ra mắt *Distronic*, hệ thống radar hỗ trợ cảnh báo an toàn được tích hợp trong xe *Mercedes-Benz S-Class (W220)*. Hệ thống này có thể tự động điều khiển tốc độ lái để giữ khoảng cách an toàn so với các xe khác xung quanh.



Hình 1.5 Mercedes-Benz S 450 4MATIC

Trong đợt ra mắt tiếp theo vào năm 2005, Mercedes nâng cấp hệ thống lên phiên bản *Distronic Plus*, và được tích hợp đầu tiên trên xe *Mercedes-Benz S-Class (W221)*.

Đầu 2017, *Mercedes* mở rộng nhiều tính năng cho xe tự hành vào các sản phẩm xe hơi của họ như:

- Hệ thống tự phanh
- Điều khiển lái tự động
- Hỗ trợ đỗ xe
- Camera xử lý ban đêm

1.3. Phạm vi nghiên cứu

Xuất phát từ nghiên cứu về mô hình xe bám làn đường và áp dụng kiến thức về phần cứng cũng như thuật toán điều khiển. Đề tài tập trung vào tìm hiểu cùng với xây dựng mô hình phần cứng, thuật toán điều khiển và chạy mô phỏng.

Mục tiêu của luận văn là xây dựng thuật toán nhận dạng làn đường và thực hiện phần cứng chạy thực tế. Hệ thống tự hành của xe sử dụng máy tính nhúng Raspberry Pi 3 và Camera để nhận dạng làn đường và tính toán khoảng cách lệch của xe so với đường tâm của làn đường.

Tùy phạm vi được xác định như trên, đề tài sẽ tập trung vào hệ thống xử lý ảnh nhận làn đường và thuật toán điều khiển bánh lái của xe.

1.4. Nhiệm vụ luận văn

Nhiệm vụ của đề tài luận văn như sau:

- Nghiên cứu xe tự hành và xây dựng mô hình phần cứng xe tự hành.
- Nghiên cứu các thuật toán xử lý ảnh và sử dụng thư viện OpenCV.
- Xây dựng thuật toán nhận dạng làn đường, ngõ ra là độ lệch của xe so với đường tâm của làn đường.
- Xây dựng thuật toán điều khiển bánh lái.

1.5. Nội dung của đề tài

Nội dung của đề tài gồm các chương sau:

Chương 1: Giới thiệu

Nội dung chương này sẽ giới thiệu sơ lược về xe tự lái, và tình hình nghiên cứu phát triển của xe tự lái trong và ngoài nước. Cuối chương này trình bày phạm vi nghiên cứu và đặt ra mục tiêu của luận văn cần đạt được.

Chương 2. Xây dựng mô hình xe tự hành

Chương này sẽ trình bày những thành phần phần cứng quan trọng trong mô hình xe tự hành, lý thuyết cơ bản về lập trình Python.

Chương 3. Sử dụng OpenCV trong xử lý ảnh

Nội dung chương này sẽ trình bày một số vấn đề cơ bản trong xử lý ảnh và các thuật toán xử lý ảnh được sử dụng trong đề tài.

Chương 4. Thuật toán xác định làn đường

Chương này trình bày phương pháp xây dựng mô hình thuật toán xác định làn đường dựa vào xử lý ảnh.

Chương 5. Ứng dụng thuật toán PID vào điều khiển bánh lái

Nội dung chương này sẽ giới thiệu về thuật toán điều khiển PID và phương pháp ứng dụng vào trong điều khiển bánh lái cho xe tự hành. Cuối chương này trình bày phương pháp điều chỉnh các hệ số PID bằng phương pháp thủ công.

Chương 6. Kết quả thử nghiệm và đánh giá

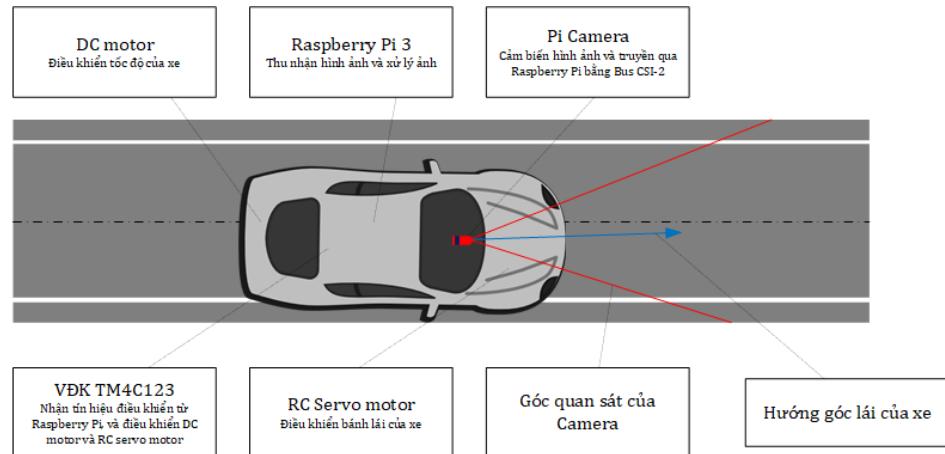
Những kết quả chạy thử nghiệm và số liệu đo đạc được sẽ được trình bày ở chương này.

Chương 7. Kết luận

Chương này trình bày về những kết quả đạt được trong luận văn, ưu điểm và nhược điểm của các giải thuật, những đóng góp và đề xuất hướng phát triển tiếp theo để hoàn thiện và mở rộng của luận văn.

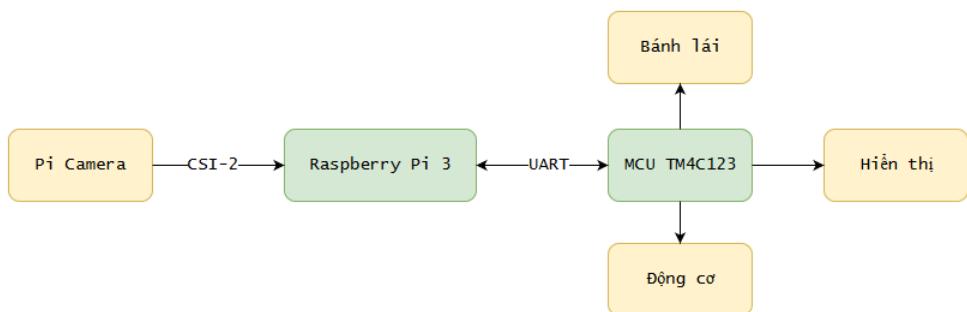
Chương 2. XÂY DỰNG MÔ HÌNH XE TỰ HÀNH

2.1. Sơ đồ tổng quát



Hình 2.1 Sơ đồ xe tự hành sử dụng thị giác máy tính
Board mạch máy tính nhúng Raspberry Pi 3 được sử dụng như thành phần trung tâm của hệ thống từ quá trình thu nhận ảnh từ camera và quá trình xử lý ảnh. Camera sử dụng trong đề tài là mô-đun *Pi NoIR Camera V2* dành riêng cho Raspberry Pi 1, 2 và 3.

Ngoài ra, vi điều khiển TM4C123 của Texas Instruments chiếm vai trò quan trọng trong việc tương tác trực tiếp điều khiển xe tự hành như bánh lái, động cơ DC và hiển thị trạng thái.

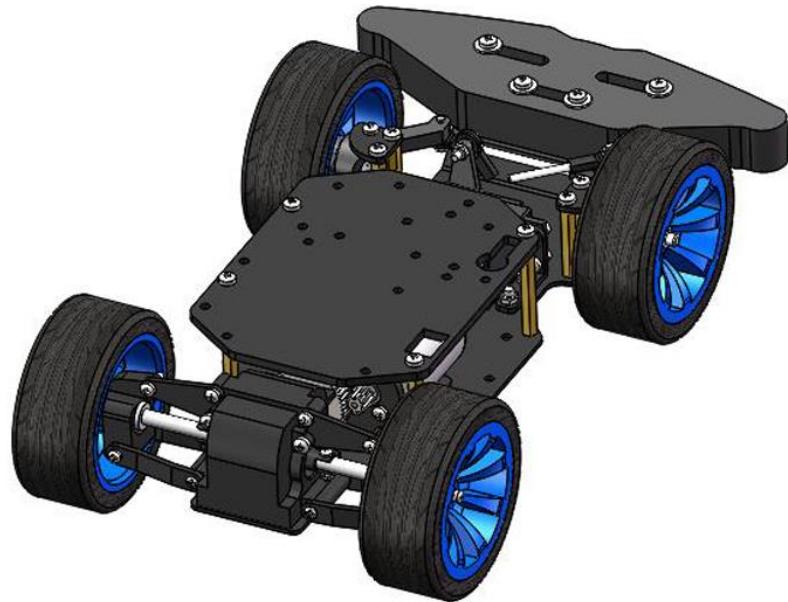


Hình 2.2 Sơ đồ hệ thống bên trong xe tự hành của đề tài
Trong hệ thống phần cứng, những thành phần chiếm vai trò quan trọng như:

- Board mạch máy tính nhúng Raspberry Pi 3 có tốc độ xử lý nhanh và 1GB RAM.
- Vi điều khiển TM4C123
- Cảm biến camera với chuẩn giao tiếp CSI-2.
- Hệ thống bánh lái sử dụng Servo RC
- Động cơ DC 12V

2.2. Mô hình phần cứng xe tự hành

Mô hình được chọn sử dụng trong đề tài là mô hình xe bốn bánh với hai bánh trước được lái bằng RC servo, và có đặc tính cơ bản gần giống như xe bốn bánh thực tế.



Hình 2.3 Mô hình phần cứng xe tự hành được sử dụng trong đề tài
Những thành phần chính của xe gồm có:

- Động cơ Servo RC MG996R
- Động cơ DC GA37Y370, 12V
- Bộ cầu vi sai giúp cho các bánh xe có thể quay với tốc độ khác nhau.

2.3. Máy tính nhúng Raspberry Pi 3

2.3.1. Giới thiệu máy tính nhúng Raspberry Pi 3

Raspberry Pi 3 là một mạch máy tính nhúng được phát triển bởi Raspberry Pi Foundation, dùng cho việc giảng dạy trong trường học và các nước phát triển. Đây là máy tính nhúng giá rẻ và có cộng đồng hỗ trợ mạnh mẽ, do đó việc thực hiện các mẫu ứng dụng thường chọn Raspberry Pi.



Hình 2.4 Mạch máy tính nhúng Raspberry Pi 3
Các tính năng nổi bật của Raspberry như:

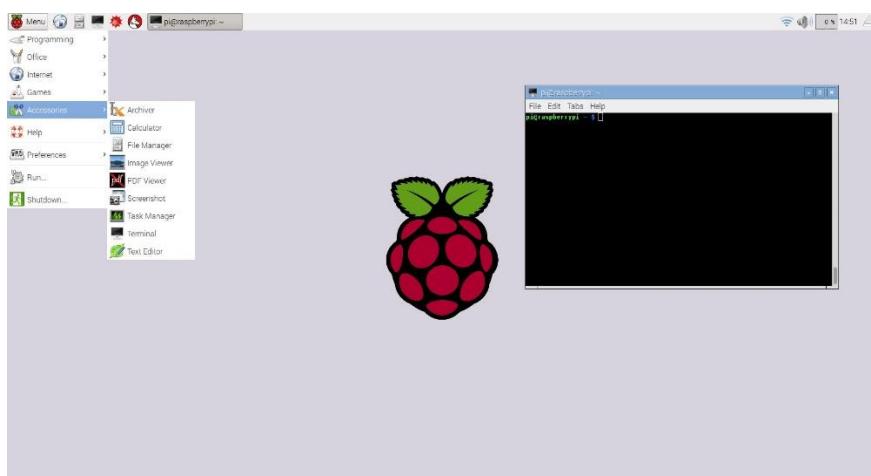
- Sử dụng CPU phiên bản mới BCM2837 từ Broadcom với tốc độ 1.2GHz 4 nhân với kiến trúc ARM Cortex A53 64-bit. Tốc độ của Raspberry Pi 2 sẽ vượt trội hơn 50-60% so với Raspberry Pi 2.
- 1GB RAM (LPDDR2 SDRAM)
- Tích hợp WiFi chuẩn 802.11n và Bluetooth 4.1
- Hỗ trợ cổng một Ethernet, một cổng HDMI và một cổng giao tiếp LCD chuẩn DSI (Display Interface).
- Hỗ trợ CSI-2 giao tiếp với camera tốc độ cao.
- Có 4 cổng USB
- 40 chân GPIO

- Tích hợp cổng âm thanh 3.5mm

Do nhu cầu của đề tài cần một máy tính nhúng với tốc độ xử lý trên 1GHz và hỗ trợ bus CSI-2 để tăng tốc độ xử lý ảnh, Raspberry Pi 3 được chọn với những tính năng thỏa nhu cầu của đề tài. Ngoài ra, Raspberry Pi hỗ trợ Wireless và có cộng đồng chia sẻ mạnh mẽ cũng tạo nhiều thuận lợi khi thực hiện đề tài.

2.3.2. Hệ điều hành cho Raspberry Pi 3

Dựa vào kiến trúc ARM 64-bit như mô tả ở trên, có rất nhiều sự lựa chọn hệ điều hành cho Raspberry Pi. Ước tính có hơn 80 phân phối dựa trên Linux (Linux-based distribution) với Raspberry Pi như Raspbian, PiDora, ArchLinux, Linutop, PiBang, v.v...



Hình 2.5 Giao diện Desktop của hệ điều hành Raspbian Raspbian được xây dựng dựa vào hệ điều hành Debian dành riêng cho Raspberry Pi và Banana Pi. Hầu như các dự án đều chọn hệ điều hành Raspbian khi sử dụng. Raspbian được cung cấp bởi Raspberry Pi Foundation, vì thế chúng ta có thể tự tin rằng nó cung cấp những tính năng tốt nhất.

Ngoài ra hệ điều hành Linux ra, Raspberry Pi 3 cũng có thể chạy trên các hệ điều hành khác như Windows IoT Core, Android, Chromium OS, FreeBSD, v.v...

Trong đề tài này, hệ điều hành Raspbian Stretch được chọn cài đặt cho Raspberry Pi 3. *Raspbian Strech* là phiên bản mới nhất của Raspbian với

nhiều tối ưu và nâng cấp các phần mềm bên trong hệ điều hành như vim, Python3, cũng như Linux kernel (release version 3.16 → release version 4.9).

2.3.3. Sơ đồ chân Raspberry Pi 3

Raspberry Pi 3 tương thích với thiết kế phần cứng và phần mềm trên các phiên bản cũ là Raspberry Pi 1 và 2.

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I2C)	DC Power 5v	04
05	GPIO03 (SCL1 , I2C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)	(I2C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40



Hình 2.6 Sơ đồ chân chi tiết của Raspberry Pi 3

Các chân của Raspberry Pi 3 ngoài có chức năng IO, ngoài ra, nó còn có một số chức năng ngoại vi khác như: PWM, UART, I2C, SPI, v.v.... Điều này giúp cho việc giao tiếp với các thiết bị bên ngoài dễ dàng hơn.

2.4. Pi Camera NoIR v2

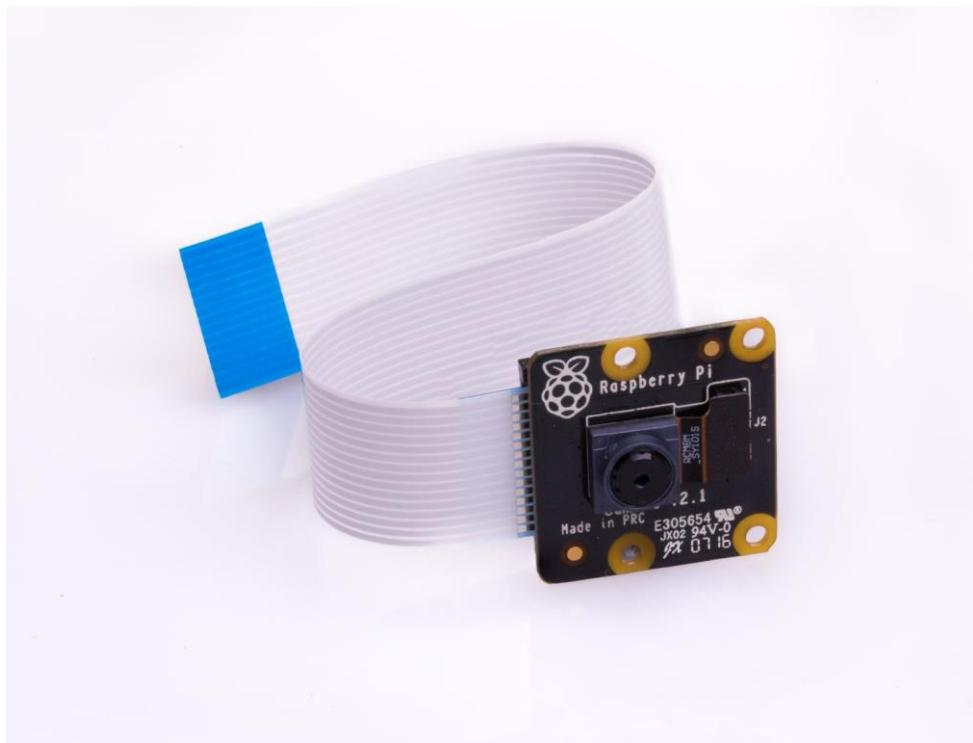
Có rất nhiều lựa chọn camera được sử dụng trong đề tài này. Về chuẩn giao tiếp, ta có thể thấy loại camera USB và camera CSI-2. Camera sử dụng bus giao tiếp CSI-2 có tốc độ nhanh hơn chuẩn USB và nó là chuẩn giao tiếp hiện đại, được sử dụng trong xe tự hành, thí dụ như chip xử lý R-CarV3M của Renesas dành

riêng cho thu nhận camera trong công nghiệp xe tự hành. Các loại camera chuẩn CSI-2 dạng mạch có thể được lựa chọn qua đánh giá như sau:

Bảng 2.1 So sánh các camera chuẩn CSI-2.

Camera	Chip	Megapi xels	Góc quan sát của camera (FOV)
Raspberry Pi Camera	OV5647	5MP	54°(h) x 41°(v)
Raspberry Pi V2 Camera	IMX219	8MP	62.2°(h) x 48.8°(v)
Arducam 5MP Rpi Camera	OV5647	5MP	54°(h) x 41°(v)
Waveshare RPi Camera (I)	OV5647	OV5647	170°
Waveshare RPi Camera (J)	OV5647	OV5647	222°
Waveshare RPi Camera IR-CUT	OV5647	OV5647	75.7°

Như bảng so sánh ở trên, ta có nhiều lựa chọn khác nhau, đặc biệt và độ phân giải và giá trị FOV của camera. Tuy nhiên mô hình ứng dụng nhỏ, nên đề tài không cần đến những camera có FOV lớn. Do đó, Raspberry Pi V2 Camera phù hợp với những phân tích nhu cầu ứng dụng như trên.

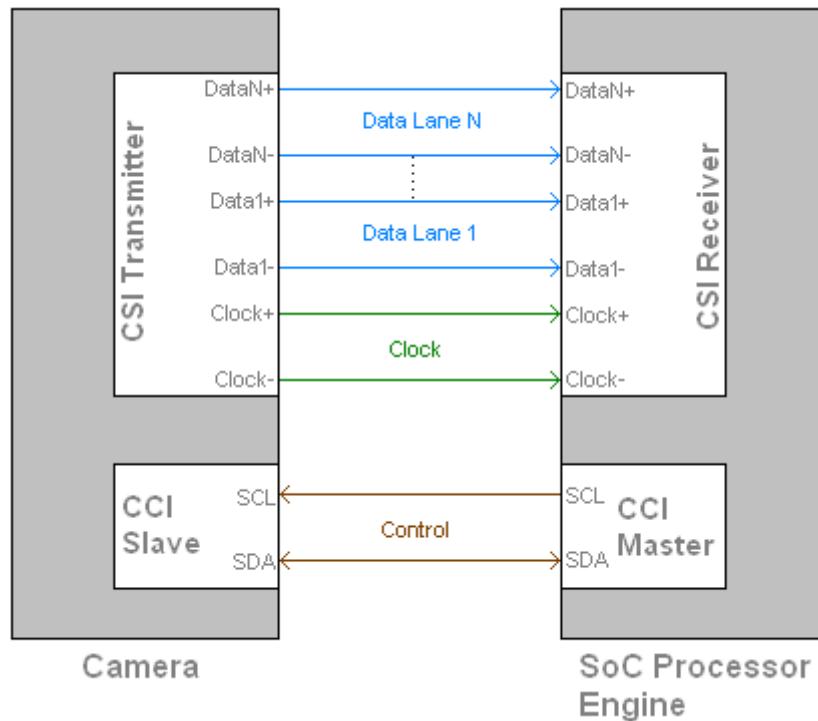


Hình 2.7 Hình ảnh Pi Camera thực tế

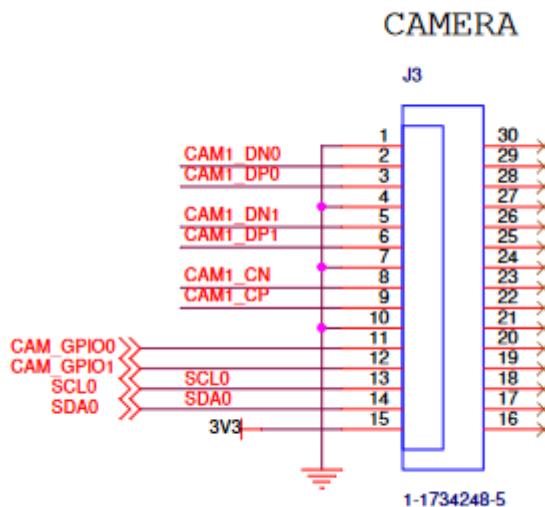
Camera Pi NoIR V2 là phiên bản camera mới nhất dành cho Raspberry Pi sử dụng cảm biến ảnh IMX219 8-Megapixel từ Sony. Với cảm biến IMX219 từ Sony, Camera cho Raspberry Pi đã có được sự nâng cấp vượt trội về cả chất lượng hình ảnh, video cũng như độ bền.

Nó được thiết kế nhỏ gọn, thích hợp cho nhiều ứng dụng công nghệ. Các thông số kỹ thuật:

- Có thể sử dụng trong điều kiện thiếu sáng
- Cảm biến IMX219 từ Sony
- Số điểm ảnh: 8MP
- Độ phân giải camera: 3280x2464 pixels
- Độ phân giải video: HD 1080p30, 720p60 và 640x480p90



Hình 2.8 Camera được gắn với Raspberry qua cáp CSI



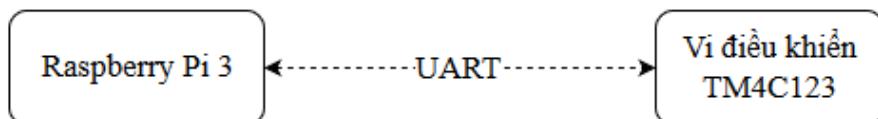
Hình 2.9 Sơ đồ chân cổng CSI-2 được thiết kế trong Raspberry Pi 3
 CSI-2 là chuẩn bus giao tiếp dữ liệu với tốc độ cao, ít nhiễu và với băng thông rộng 2G. Nó hỗ trợ nhiều định dạng dữ liệu đầu ra như RGB, RAW, YUV.

Chuẩn giao tiếp CSI-2 được sử dụng rộng rãi trong công nghiệp di động như trong xe tự lái, camera drone, ứng dụng IoT, và hệ thống an ninh. Nó hỗ trợ nhiều ứng dụng với hiệu năng cao như video 1080p, 4K, 8K, v.v...

Có rất nhiều bộ thư viện được cộng đồng Raspberry Pi phát triển trên Python giúp cho việc tìm hiểu và sử dụng trở nên dễ dàng hơn rất nhiều. Trong đề tài, thư viện *Picamera* được chọn sử dụng với các hàm API dễ sử dụng và dễ cài đặt.

2.5. Truyền nhận dữ liệu qua UART

Truyền nhận dữ liệu chiếm vai trò quan trọng trong mô hình, nếu quá trình truyền nhận dữ liệu tốt thì đáp ứng của xe sẽ nhanh hơn. Dựa vào những giao tiếp có sẵn của mạch máy tính nhúng Raspberry Pi và tính đơn giản hệ thống, đề tài chọn giao tiếp dữ liệu qua UART để truyền nhận dữ liệu giữa Raspberry Pi và VĐK TM4C123.



Hình 2.10 Sơ đồ truyền nhận giữa Raspberry và vi điều khiển
Để đạt hiệu quả cao trong quá trình truyền nhận dữ liệu, việc định nghĩa hay định dạng dữ liệu là rất cần thiết. Dữ liệu truyền nhận được định dạng theo từng gói 4 byte. Với các qui ước như sau :

Bảng 2.2 Định dạng dữ liệu truyền giữa Raspberry Pi 3 và VĐK TM4C123

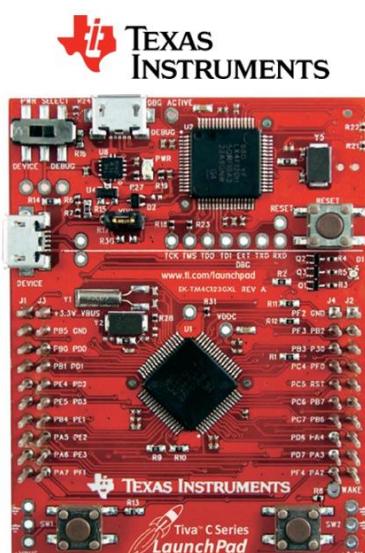
Dữ liệu truyền	Thực thi
Độ lệch đo được từ Raspberry	-160÷160
Khởi động xe	254
Ngừng xe	255
Cập nhật thông số K_p trong PID	240 + K_p

Cập nhật thông số K_I trong PID	$250 + K_I$
Cập nhật thông số K_D trong PID	$260 + K_D$
Tốc độ xe	$270 + \text{Speed}$

Gói dữ liệu điều khiển được truyền từ Raspberry qua UART là một số nguyên có dấu (signed integer). Vi điều khiển thu nhận dữ liệu và chấp hành thực thi điều khiển xe.

2.6. Vi điều khiển TM4C123

Xe tự hành của đề tài sử dụng VDK để điều khiển động cơ RC servo và động cơ DC cho bánh lái và tốc độ di chuyển của xe. Hiện nay, hầu hết các vi điều khiển đều sử dụng kiến trúc ARM Cortex M. Nó được biết đến về tốc độ xử lý, cơ chế quản lý ngắt, FPU, v.v... Những ứng dụng điều khiển cơ bản thường sử dụng các VDK 32-bit có tần số xử lý nhanh và hỗ trợ FPU trong việc tính toán thuật toán, thí dụ các vi điều khiển ARM Cortex-M4 thường dùng như: TM4C123, TM4C129 của Texas Instruments; STM32Fx của ST, v.v... Trong đề tài này, tôi sử dụng VDK TM4C123, ngoài những tính năng có được của ARM Cortex-M4 thì nó còn có bộ thư viện chuẩn hỗ trợ lập trình và phần mềm IDE miễn phí.



Hình 2.11 Mạch vi điều khiển TM4C123 được sử dụng trong luận văn
Một số đặc điểm kỹ thuật của vi điều khiển TivaC TM4C123GH6PM:

- Nhân ARM Cortex M4 32-bit, xung nhịp tối đa 80MHz.
- Tích hợp đơn vị tính toán dấu chấm động – FPU.
- Có 256KB Flash và 2KB EEPROM.
- Tích hợp các chuẩn giao tiếp UART, SPI, I2C và USB.
- Có khả năng xuất tối đa 16 kênh PWM.

2.7. Lập trình Python

2.7.1. Giới thiệu về ngôn ngữ Python

Python là trình thông dịch cấp cao (Interpreted language), chương trình khi viết xong sẽ được chạy trực tiếp bởi trình thông dịch mà không cần biên dịch (compile) trước. Các cú pháp và tính chất của ngôn ngữ Python giống các ngôn ngữ lập trình hướng đối tượng khác như C#, C++, Java, ... Tuy nhiên một ưu thế lớn nhất của Python là mã nguồn mở nên được sử dụng ngày càng rộng rãi trên thế giới. Ngôn ngữ lập trình Python ít được sử dụng trong các trường đại học ở Việt Nam, tuy nhiên theo khuynh hướng thế giới thì các phần mềm mã nguồn mở sẽ ngày càng phát triển. Vì các lý do trên mà trong đề tài này em sử dụng Python (version 2.7) để lập trình xử lý ảnh với thư viện OpenCV. Ngoài lý do đặc điểm ngôn ngữ ra, ngôn ngữ lập trình Python có rất nhiều thư viện hỗ trợ như xử lý ảnh (OpenCV), đồ họa, machine learning, v.v... phù hợp cho hướng đề tài phát triển sau này.



Hình 2.12 Qui trình thông dịch của ngôn ngữ Python

2.7.2. Tổng quan về kiến trúc và cấu trúc của ngôn ngữ Python

Tương tự như các ngôn ngữ khác Python cũng cung cấp các kiểu biến, các toán tử và các biểu thức điều kiện như int, float, string, object, v.v... cộng, trừ, nhân, chia, mũ, v.v... if, for, while, v.v... Sau đây là ví dụ một số chương

trình nhỏ viết bằng Python để làm rõ hơn cấu trúc của chương trình phần mềm viết bằng Python.

Chương trình 1: *Hello World* trong Python

```
print('Hello World')
```

Chương trình 2: Người dùng nhập hai kích thước rộng và dài, sau đó chương trình sẽ tính chu vi và diện tích hình chữ nhật.

```
def chu_vi(chieu_dai, chieu_rong):
    return 2 * (chieu_dai + chieu_rong)
def dien_tich(chieu_dai, chieu_rong):
    return (chieu_dai * chieu_rong)
dai = input('Chieu dai : ')
rong = input('Chieu rong : ')
print('Chu vi : ', chu_vi(dai, rong))
print('Dien tich : ', dien_tich(dai, rong))
```

Chương trình 3: Viết hàm tính đệ quy dãy số *Fibonacci*, sau đó tính trong trường hợp $n = 6$.

```
def fibonacci(n):
    """ f(n) = f(n-1) + f(n-2) """
    if n == 0 or n == 1:
        return 1
    else:
        res = fibonacci(n-1) + fibonacci(n-2)
        return res
print fibonacci(6)
```

Ở chương trình 1 lệnh *print* sẽ in ra chuỗi “Hello World” ra màn hình. Ở chương trình 2 hàm *input* sẽ xuất ra tên các kích thước và người dùng phải nhập vào để tích. Có hai chương trình con tính diện tích và chu vi là hàm *chu_vi* và hàm *dien_tich*. Ở chương trình 3 là hàm đệ quy để tính chuỗi số Fibonacci.

Từ các chương trình ví dụ trên, ta có thể hiểu được cách viết một chương trình đơn giản bằng Python, các định nghĩa hàm dùng từ khóa *def* và cách sử dụng điều kiện *if*.

2.7.3. Các thư viện

2.7.3.1. Numpy

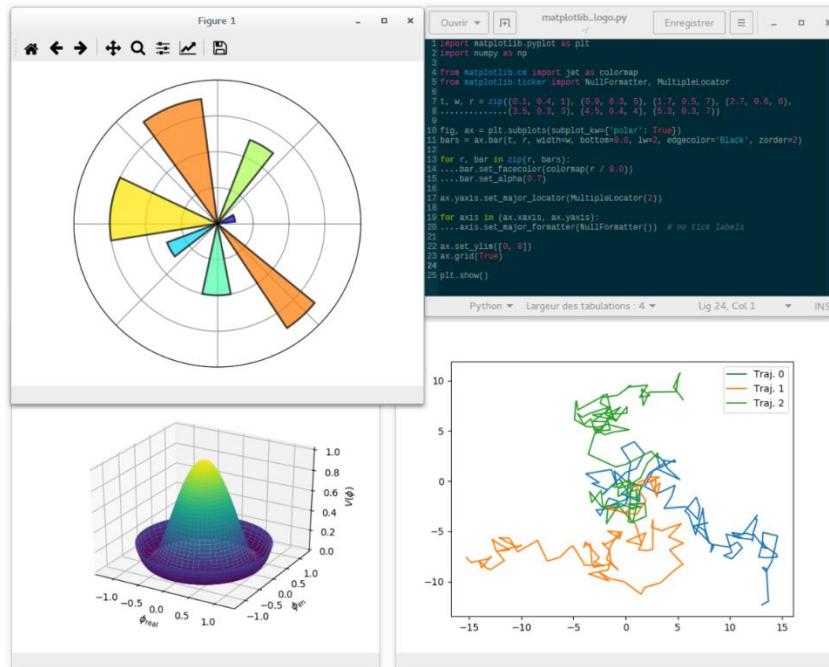
Numpy là một gói thư viện cốt lõi hỗ trợ mạnh mẽ cho khoa học máy tính trong Python. Nó mang lại hiệu năng tính toán cao khi làm việc với mảng nhiều chiều. Vì vậy nó rất cần thiết trong xử lý ảnh cũng như tính toán số trong khoa học tính toán.

Ví dụ chương trình tạo một mảng dữ liệu sử dụng *Numpy*:

```
import numpy as np
data = np.array([1.0, 2.0, 3.0])
print(type(data))
print(data[0], data[1], data[2])
data[0] = 5.0
print(data[0])
```

2.7.3.2. Matplotlib

Matplotlib là thư viện hỗ trợ mạnh mẽ trong việc vẽ 2D. Đặc biệt quan trọng là sử dụng mô-đun *matplotlib.pyplot* cho việc vẽ đồ thị phục vụ phân tích kết quả đo đặt.



Hình 2.13 Thư viện Matplotlib

2.7.3.3. Picamera

Thư viện *Picamera* cung cấp các hàm API để giao tiếp với mô-đun *Pi Camera*. Nó hỗ trợ nhiều API cho việc thu thập hình ảnh.

2.7.3.4. Pyserial

Thư viện *PySerial* được phát triển bởi Chris Liechti nhằm phục vụ mục đích tạo ra một *interface* đơn giản để giao tiếp giữa Python và phần cứng thông qua cổng nối tiếp. Các đặc điểm của thư viện này là:

- Hỗ trợ các giao tiếp bao gồm: UART, TCP/IP.
- Cùng chung một *interface* cho tất cả hai kiểu giao tiếp trên.
- Hỗ trợ các kiểu truy cập khác nhau như truyền nhận từng byte, chuỗi, v.v...
- API đơn giản dễ sử dụng.

Một số *interface* hỗ trợ bởi thư viện là:

Bảng 2.3 Một số *interface* hỗ trợ bởi thư viện Pyserial

Interface	Chức năng
<code>__inti__()</code>	Khởi động các giá trị cho cổng như tên cổng, baudrate, số bit dữ liệu, thời gian timeout, ...
<code>open()</code>	Mở cổng sẵn sàng cho sử dụng truyền nhận.
<code>close()</code>	Đóng cổng nối tiếp.
<code>read(size)</code>	Nhận dữ liệu với số byte được xác định trước.
<code>write(data)</code>	Truyền dữ liệu.
<code>readline()</code>	Nhận dữ liệu cho đến khi đọc được ký tự '\n'
<code>writeline(string)</code>	Truyền một chuỗi data dạng <i>String</i> ra cổng nối tiếp.

Ví dụ chương trình mở cổng nối tiếp, truyền 4 byte dữ liệu dùng *PySerial*:

```
import serial
port = serial.Serial(port="/dev/ttyS0", baudrate=115200, timeout=3.0)
port.write(struct.pack('i', int(254)))
port.close()
```

2.7.3.5. TkInter

Tkinter là từ viết tắt của *Tk interface*, là một trong các bộ thư viện dành cho lập trình giao diện (GUI) của Python được phát triển bởi Sun Labs.

Chương trình cơ bản sử dụng Tkinter:

```
from Tkinter import *

root = Tk()
header = Label(root, font="Times 30").grid(row=0)

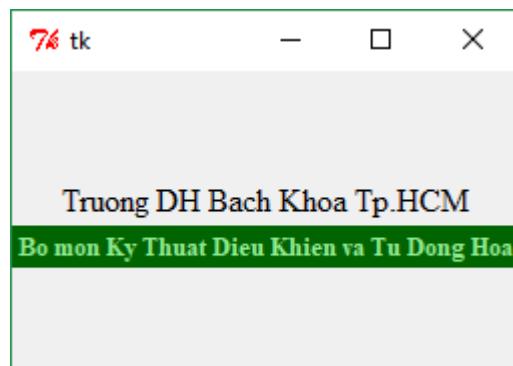
w1 = Label(root,
           text="Truong DH Bach Khoa Tp.HCM",
           font="Times").grid(row=1)

w2 = Label(root,
           text="Bo mon Ky Thuat Dieu Khien va Tu Dong Hoa",
           fg="light green",
           bg="dark green",
           font="Times 10 bold").grid(row=2)

footer = Label(root, font="Times 30").grid(row=3)

root.mainloop()
```

Sau khi chạy chương trình, kết quả giao diện của chương trình như sau:

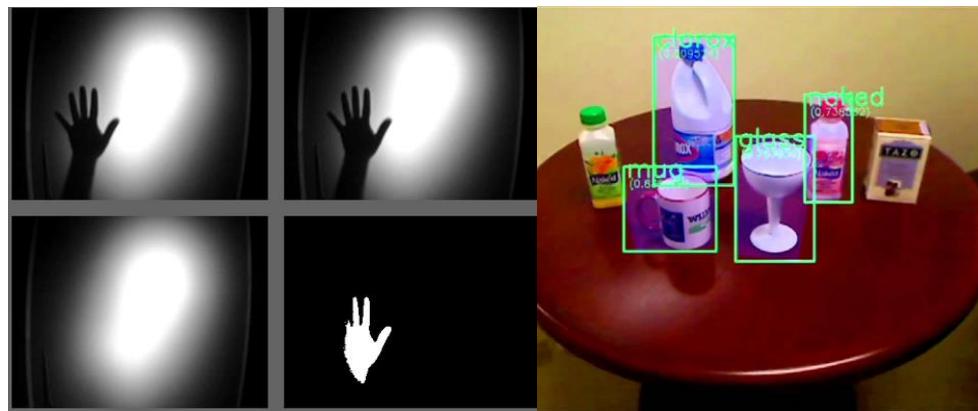


Hình 2.14 Giao diện của chương trình mẫu sử dụng thư viện Tkinter

Chương 3. THƯ VIỆN OPENCV TRONG XỬ LÝ ẢNH

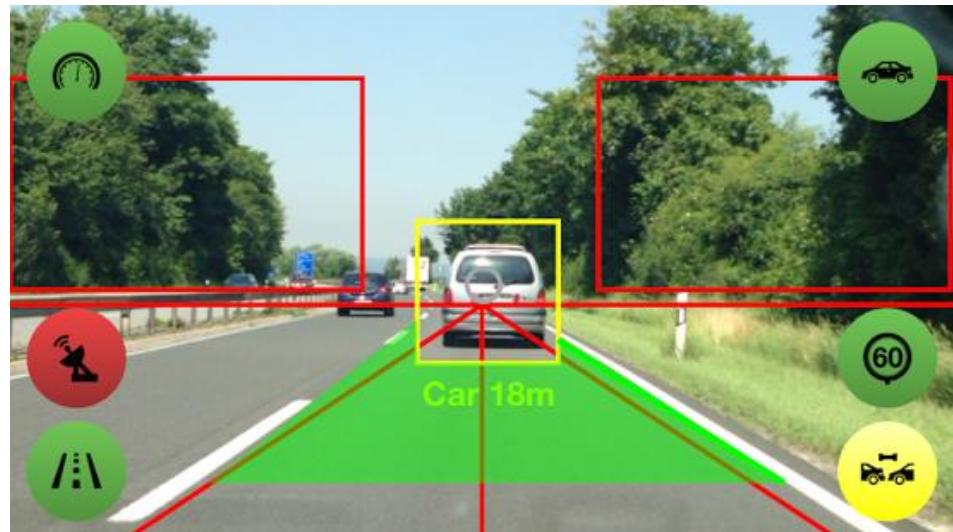
3.1. Tổng quan về bộ thư viện OpenCV

OpenCV được viết tắt của Open Source Computer Vision, là thư viện xử lý ảnh mã nguồn mở hoàn toàn miễn phí của Intel. OpenCV là một thư viện mở gồm các hàm được xây dựng phục vụ cho việc xử lý thị giác máy tính thời gian thực (Real time computer vision). Các thuật toán xử lý ảnh thông thường lẩn cao cấp đều được tối ưu hóa bởi các nhà phát triển thư viện thành các hàm đơn giản và rất dễ sử dụng.



Hình 3.1 Các ứng dụng của OpenCV

Ứng dụng của OpenCV trong nhận dạng cử chỉ tay của người, hay dùng trong nhận dạng vật thể trong nhà với thuật toán nâng cao Deep Learning.



Hình 3.2 Hình ảnh về ứng dụng của OpenCV trong xe tự hành

Intel đưa ra phiên bản OpenCV đầu tiên vào năm 1999. Ban đầu nó yêu cầu phải có thư viện xử lý ảnh của Intel. Sau đó vì lê thuộc này mà họ đã phải gỡ bỏ và bây giờ chúng ta có thể sử dụng thư viện này hoàn toàn độc lập.

Thời điểm ban đầu của OpenCV, mục tiêu của nó tóm lượt như sau:

- Hỗ trợ nghiên cứu bằng cung cấp những mã tối ưu hóa một cách cơ bản.
- Phổ biến kiến thức về xử lý ảnh bằng cách cung cấp các cơ sở phổ biến mà các nhà phát triển có thể xây dựng trên, vì vậy mà mã sẽ dễ dàng hơn và có thể hiểu và chỉnh sửa dễ dàng hơn.
- Không đòi hỏi bản quyền hay thương mại hóa và hoàn toàn miễn phí.

Phiên bản alpha đầu tiên của OpenCV đã được phát hành tại Hội nghị IEEE về “Computer Vision and Pattern Recognition” trong năm 2000, và năm lần tung ra các phiên bản beta giữa năm 2001 và 2005. Phiên bản 1.0 đầu tiên được phát hành vào năm 2006. Vào giữa năm 2008, OpenCV nhận được hỗ trợ tích cực từ các doanh nghiệp, và phát triển mạnh mẽ. Phiên bản 1.1 phát hành thử nghiệm vào tháng 10 năm 2008. Cùng thời gian đó, một cuốn sách của hai tác giả viết về OpenCV được xuất bản bởi O'Reilly ra thị trường.

Tháng 10 năm 2009, OpenCV ra phiên bản thứ 2, OpenCV 2.0 bao gồm những thay đổi lớn vào API C++, nhằm dễ dàng hơn, nhiều kiểu mẫu an toàn, các chức năng mới, và thực thi tốt hơn về hiệu suất (đặc biệt là trên hệ thống đa lõi ngày càng phổ biến).

Tính tới thời điểm hiện tại, OpenCV đã bao gồm hơn 500 thuật toán giải thuật ứng dụng trên hơn 28 lĩnh vực của xử lý ảnh, với những ưu điểm như sau:

- Đồng thời hỗ trợ rất nhiều hệ điều hành như: Windows, Linux và MacOSX.
- Viết trên nhiều nền tảng C++, Python. Ngoài ra, còn có thể viết trên nền C# hay VB với việc dùng các thư viện hỗ trợ truy cập OpenCV.
- Luôn luôn cập nhật liên tục nhưng giải thuật mới.

Hiện tại, OpenCV đã phát hành bản OpenCV 3.3 với sự hỗ trợ mạnh mẽ hơn trong lĩnh vực Machine Learning và AI.

3.2. Các xử lý cơ bản trong xử lý ảnh

Xử lý ảnh dùng để tăng cường chất lượng ảnh hoặc trích xuất những thông tin cần thiết. Qui ước rằng các phép xử lý ảnh bên dưới sẽ thực hiện trên ảnh số (digital image), là ảnh biểu diễn trong không gian rời rạc 2D.

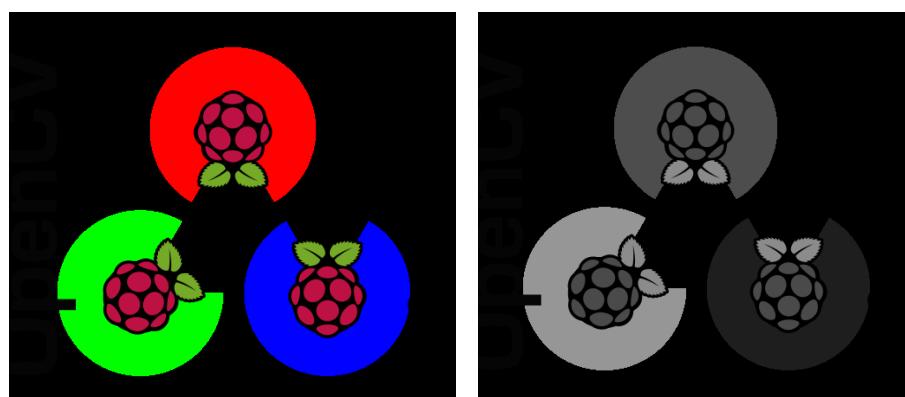
3.2.1. Ảnh xám (Gray image)

OpenCV có hơn 150 hàm để chuyển đổi không gian màu. Nhưng thường chỉ sử dụng hai loại chính là $BGR \leftrightarrow Gray$ và $BGR \leftrightarrow HSV$.

Công thức chuyển đổi từ một ảnh BGR sang ảnh đơn màu:

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

Để chuyển đổi từ một ảnh BGR sang ảnh gray, ta sử dụng hàm `cv2.cvtColor(hinh_anh_bgr, loai_chuyen_doi)` trong OpenCV Python, trong đó `loai_chuyen_doi` là cờ (flag) để xác định loại chuyển đổi, ở đây, chúng ta sử dụng cờ `cv2.COLOR_BGR2GRAY`.



Hình 3.3 Ảnh trước và sau khi thực hiện phép chuyển ảnh gray

Việc chuyển ảnh BGR sang ảnh gray sẽ giúp cho việc xử lý về sau nhanh

hơn và nó đã đủ thông tin cho bất cứ nhiệm vụ nào của xử lý ảnh. Ảnh màu sẽ không giúp ta xác định được các cạnh của ảnh hay các tính năng khác trong xử lý ảnh.

3.2.2. Bộ lọc nhiễu cho ảnh

Trong xử lý ảnh, chúng ta có thể lọc ảnh bằng nhiều thuật toán như lọc thông thấp (low-pass filtering), lọc thông cao (high-pass filter), lọc FFT, v.v... Trong đó, bộ lọc thông thấp thường sử dụng trong các kỹ thuật khử nhiễu và làm mờ nhiễu. Bộ lọc thông cao giúp xác định các cạnh trong ảnh.

Làm mờ ảnh được thực hiện bởi tính chập ảnh với một kernel lọc thông thấp. Nó hữu ích trong việc khử nhiễu trong ảnh, bởi nó thật sự khử đi các thành phần tần số cao (như nhiễu và cạnh trong ảnh) trong ảnh. Vì vậy, các cạnh của ảnh cũng bị làm mờ đi một ít khi thực hiện phép toán làm mờ ảnh.

Cho một ảnh ngõ vào $f(x, y)$, ảnh ngõ ra $g(x, y)$ được tính bởi một phép toán với các giá trị điểm ảnh khu vực lân cận (neighbourhood) xung quanh của một điểm ảnh bên trong $f(x, y)$. Phép toán đó có thể tuyến tính hoặc không tuyến tính. Biểu diễn chung của phép toán như sau:

$$g(x, y) = H_N(f(x, y))$$

Trong đó, H_N là phép toán khu vực lân cận với các điểm ảnh xung quanh và có size là N .

Với phép toán khu vực lân cận H_N không tuyến tính, ảnh ngõ ra không thể tính được bằng phép toán tính chập.

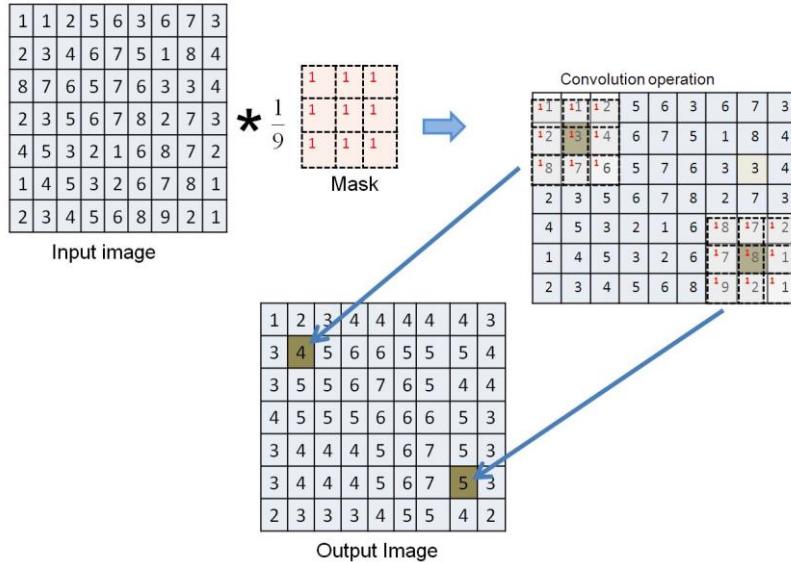
Ngược lại, đối với phép toán H_N là tuyến tính thì phép toán tính chập trong xử lý ảnh được biểu diễn theo công thức sau:

$$g(x, y) = \sum_{u=a}^a \sum_{v=b}^b w(u, v) f(x + u, y + v)$$

Trong đó, $f(x, y)$ là ảnh đầu vào, $g(x, y)$ là ảnh đầu ra khi tính chập với cửa sổ $w(x, y)$. Với cửa sổ kernel có kích thước là $s \times t$ thì khi đó $a = (s - 1)/2$ và $b = (t - 1)/2$.

Thí dụ chúng ta có một cửa sổ kernel 3x3 sẽ được biểu diễn như sau:

$w(-1, -1)$	$w(-1, 0)$	$w(-1, 1)$
$w(0, -1)$	$w(0, 0)$	$w(0, 1)$
$w(1, -1)$	$w(1, 0)$	$w(1, 1)$



Hình 3.4 Hình ảnh mô tả thực hiện tính chập cho từng điểm ảnh

Phép toán tính chập trên được sử dụng cho phép toán lọc nhiễu hay làm mượt ảnh trong xử lý ảnh và cửa sổ kernel được xem như một bộ lọc. Đối với bộ lọc thông thấp, các hệ số bên trong cửa sổ kernel sẽ là số dương, và tổng các hệ số trong kernel phải bằng 1 để đảm bảo rằng điểm ảnh trung bình trong ảnh đã chỉnh sửa giống như ảnh gốc ban đầu.

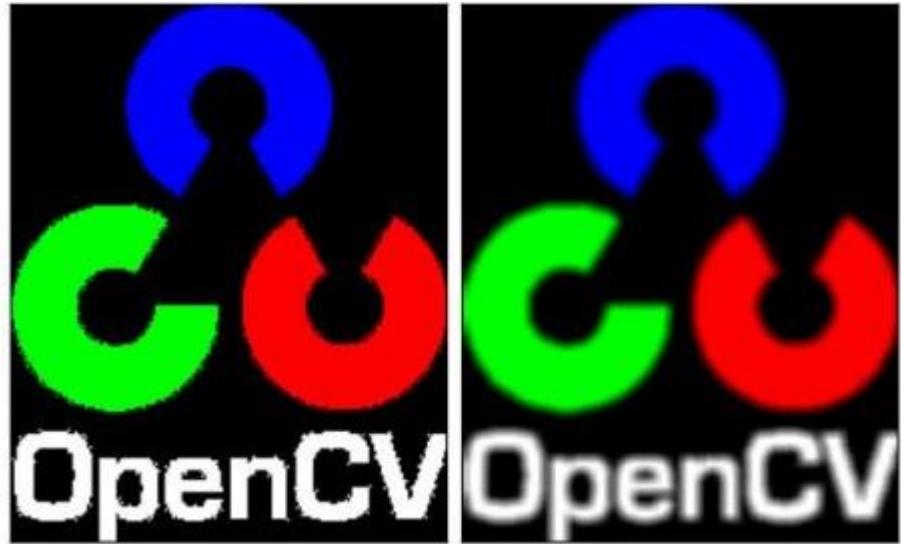
OpenCV cung cấp 4 kỹ thuật làm mờ ảnh chính như sau:

3.2.2.1. Bộ lọc trung bình

Kỹ thuật làm mờ ảnh bằng phương pháp tính trung bình sử dụng một kernel với các hệ số bên trong kernel bằng nhau.

Thí dụ thực hiện làm mờ ảnh với bộ lọc kernel 5x5 như sau:

$$kernel = \frac{1}{255} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



Hình 3.5 Ảnh trước và sau khi làm mờ bằng phương pháp trung bình
Kỹ thuật tính trung bình này có ưu điểm phép tính đơn giản nên tốc độ tính toán sẽ nhanh hơn các kỹ thuật lọc khác. Nhưng ngược lại nó là làm mờ đi các cạnh của ảnh, gây bất lợi cho quá trình xử lý về sau.

3.2.2.2. Làm mờ Gaussian

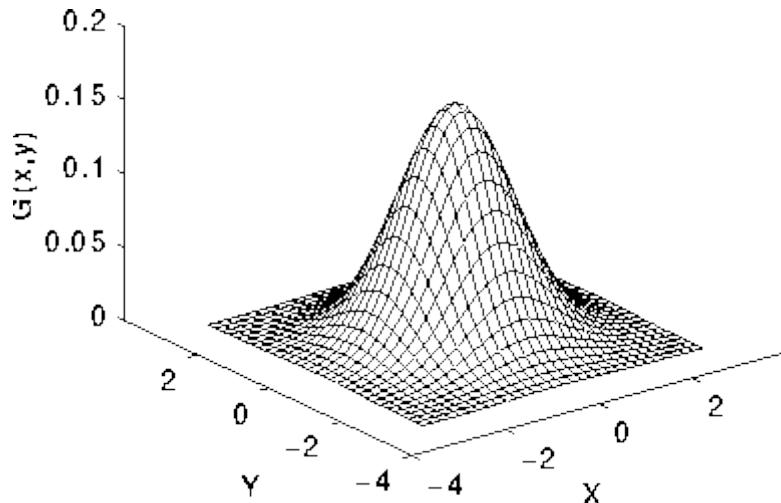
Cùng loại với bộ lọc thông thấp, chúng ta có thể sử dụng kỹ thuật làm mờ ảnh Gaussian. Thay vì, chúng sử dụng các hệ số trong cửa sổ kernel giống nhau như bộ lọc trung bình ở trên, các hệ số bên trong sẽ được tính bởi phân phối Gaussian. Công thức tính phân phối Gaussian trong không gian ảnh 2 chiều được biểu diễn như sau:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Trong đó, σ là độ lệch chuẩn của phân phối Gaussian.

Với cửa sổ kernel là 3×3 thì giá trị x và y được biểu diễn như sau:

$$x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$



Hình 3.6 Đồ thị phân phối Gaussian với độ lệch chuẩn $\sigma = 1$

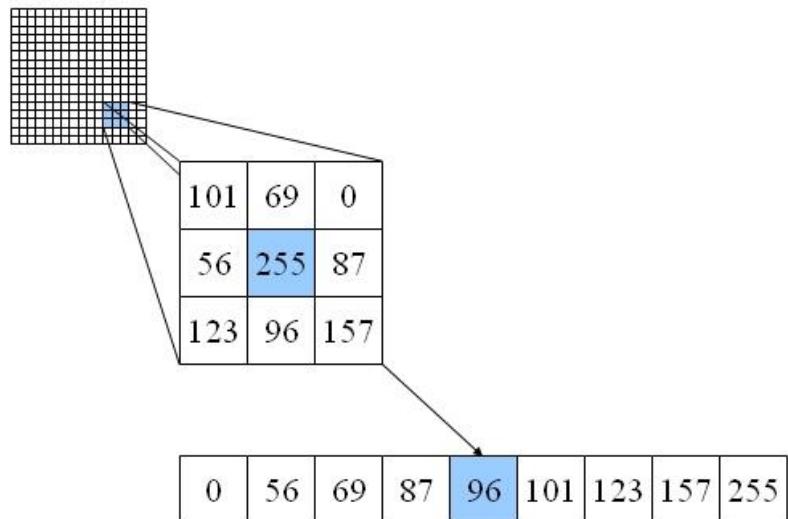


Hình 3.7 Ảnh trước và sau khi làm mờ bằng gaussian

3.2.2.3. Làm mờ Median

Đây là kỹ thuật làm mờ median không tuyến tính nên không thể áp dụng phép tính chập. Làm mờ median cũng sử dụng một cửa sổ kernel với kích thước được xác định trước, và tính median của tất cả các điểm ảnh dưới cửa sổ kernel, sau đó điểm ảnh chính giữa cửa sổ kernel được thay bằng giá trị median vừa tính.

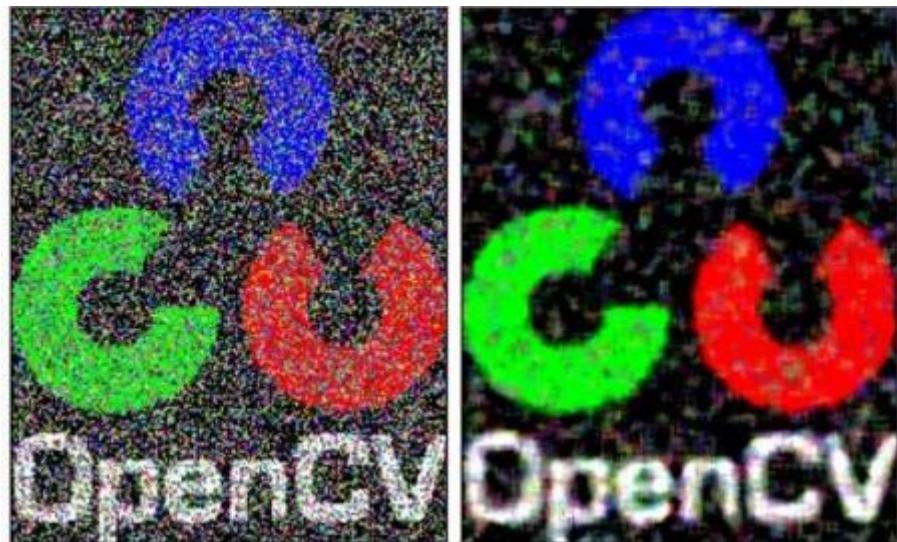
Cách tính median cho một cửa sổ ảnh được mô tả như hình sau:



Hình 3.8 Cách tính median cho một cửa sổ ảnh

Chi phí thời gian tính toán cho một ảnh của phương pháp này nhiều hơn khi sử dụng cách làm mờ Gaussian, bởi phép tính median phải sắp xếp tất cả các giá trị trong cửa sổ kernel và nó thực hiện lặp lại cho tất cả các điểm ảnh. Phương pháp này phù hợp cho các loại nhiễu chấm hột trên ảnh (salt-and-pepper).

Thí dụ, ảnh gốc ngõ vào đã được cộng thêm 50% nhiễu salt-and-pepper và áp dụng phương pháp làm mờ median. Kết quả như hình sau:



Hình 3.9 Ảnh trước và sau khi làm mờ median

3.2.2.4. Bộ lọc bilateral

Các phương pháp làm mờ ảnh ở trên đều ảnh hưởng đến các cạnh trong ảnh. Cùng là phép toán khu vực lân cận như kỹ thuật làm mờ median, phương pháp lọc và làm mờ ảnh *bilateral* đạt hiệu quả cao trong lọc nhiễu ảnh nhưng vẫn bảo vệ được các cạnh của ảnh không ít bị mờ. Nhưng ngược lại, phương pháp này lại tốn rất nhiều thời gian tính toán, làm ảnh hưởng đến thời gian xử lý của ứng dụng.

Bộ lọc bilateral được định nghĩa như sau:

$$I^{filtered}(x) = \frac{1}{W_D} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

Trong đó:

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

Trọng số W_p tính độ chênh lệch không gian và cường độ của điểm ảnh.

- $I^{filtered}$ là ảnh đã được lọc
- I là ảnh gốc ban đầu đưa vào bộ lọc
- x là tọa độ của điểm ảnh đang được lọc
- Ω là cửa sổ kernel với tâm cửa sổ là x
- f_r là hàm cho sự khác biệt cường độ của điểm ảnh (hàm này có thể là hàm Gaussian)
- g_s là hàm tính sự khác biệt không gian (hàm này có thể là hàm Gaussian)

Bộ lọc bilateral cũng thực hiện lọc gaussian như phương pháp làm mờ gaussian, nhưng nó thêm một lọc gaussian khác cho độ lệch giữa các cường độ của điểm ảnh (hay giá trị của điểm ảnh đó). Hàm gaussian cho không gian đảm bảo rằng chỉ các điểm ảnh gần mới được xem xét làm mờ. Trong khi đó, hàm gaussian cho độ lệch cường độ điểm ảnh đảm bảo rằng những điểm ảnh có cường độ giống nhau mới được xem xét làm mờ. Vì vậy, phương pháp lọc này bảo vệ được các cạnh của ảnh, bởi vì các điểm ảnh tại các cạnh sẽ có độ thay đổi cường độ lớn.



Hình 3.10 Hình Lọc nhiễu bằng bilateral

Tóm lại, tùy vào những ràng buộc và đặc trưng của từng ứng dụng mà chúng ta chọn bộ lọc phù hợp.

Để tài xe tự hành chọn kỹ thuật làm mờ gaussian để lọc nhiễu của ảnh.

3.2.3. Xác định cạnh trong ảnh

Các cạnh của ảnh là những vùng ảnh mà có độ tương phản cao. Vì thế các cạnh thường xuyên xuất hiện tại những vị trí được thấy như là những đường bao quanh một vật trên hình ảnh. Xác định cạnh thường được dùng phổ biến trên những hình ảnh có nhiều vật thể khác nhau khi ta muốn chia hình thành những vùng khác nhau có chứa vật thể hay tách đặc trưng của ảnh đó. Biểu diễn một hình ảnh bằng các cạnh thì có nhiều thuận lợi hơn là làm giảm được dữ liệu ảnh trong ảnh trong khi vẫn đảm bảo giữ được những thông tin cần thiết trong ảnh.



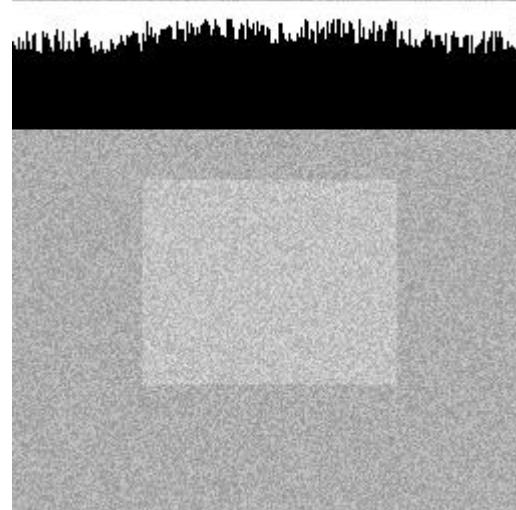
Hình 3.11 Hình ảnh cơ bản về xác định cạnh của ảnh

Các cạnh chủ yếu là tần số cao nên theo lý thuyết, dò cạnh sử dụng bộ lọc tần số cao bằng phương pháp Fourier hay bằng cách nhân chập hình ảnh với những kernel thích hợp trong miền không gian Fourier. Trên thực tế dò cạnh

được thực hiện trong miền không gian vì thực hiện dễ dàng hơn và thường có kết quả tốt hơn.

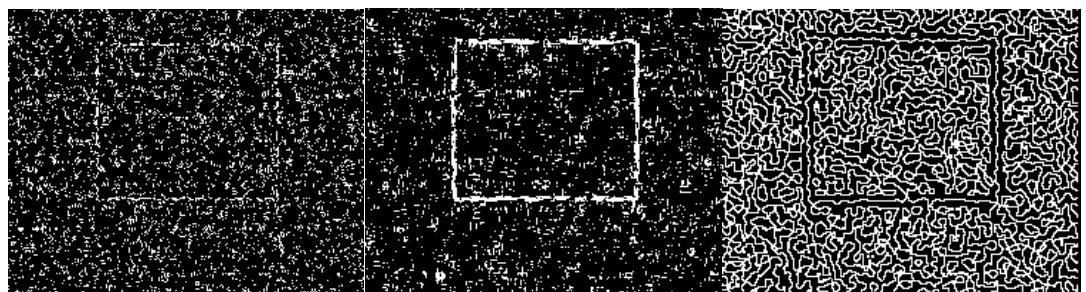
Có nhiều phương pháp xác định cạnh trong ảnh như Robert Cross, Sobel, Canny, Perwitt, DoG, v.v...

Thí dụ ta cho một hình ảnh ngõ vào chứa hình chữ nhật như sau:

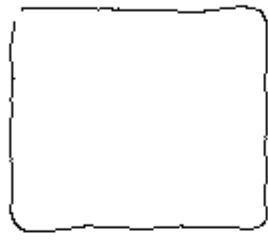


Hình 3.12 Ảnh mẫu thử các thuật toán xác định cạnh

Kết quả xác định cạnh của các thuật toán Sobel, Prewitt, DoG và Canny như sau:



Hình 3.13 Kết quả xác định cạnh của Sobel, Prewitt và DoG



Hình 3.14 Kết quả xác định cạnh của Canny

Thuật toán Canny được sử dụng phổ biến trong việc xác định cạnh trong ảnh. Nó thỏa mãn 3 tiêu chí như sau:

- Tỉ lệ lỗi thấp.
- Xác định vị trí cạnh chính xác: khoảng cách giữa điểm ảnh và cạnh nhỏ mức tối thiểu.
- Cạnh nhỏ tối thiểu.

Xác định cạnh Canny phải thực hiện nhiều bước tính toán bên trong nó mới xác định được cạnh của ảnh. Để đơn giản, thuật toán Canny được chia thành 6 bước như sau:

Bước 1: Làm mượt ảnh bộ lọc Gaussian để giảm nhiễu và những chi tiết không mong muốn.

$$g(m, n) = G_\sigma(m, n) * f(m, n)$$

Trong đó,

$$G_\sigma = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{m^2+n^2}{2\sigma^2}}$$

Bước 2: Tính gradient của $g(m, n)$ bằng cách sử dụng phép toán gradient.

$$\text{Biên độ của gradient: } M(m, n) = \sqrt{g_m^2(m, n) + g_n^2(m, n)}$$

$$\text{Hướng của gradient: } \theta(m, n) = \tan^{-1}\left(\frac{g_m(m, n)}{g_n(m, n)}\right)$$

Bước 3: Sử dụng ngưỡng T để lọc những giá trị nhiễu của $M(m, n)$. Khi đó :

$$M_T(m, n) = \begin{cases} M(m, n), & \text{nếu } M(m, n) > T \\ 0, & \text{nếu } M(m, n) \leq T \end{cases}$$

Trong đó, giá trị T được chọn sao cho giá trị cạnh của ảnh được giữ lại và loại bỏ hết nhiễu.

Bước 4: Giảm những điểm ảnh non-maxima (non-maxima pixels) trong các cạnh trong M_T được tìm ở trên để thu hẹp đường cạnh của ảnh. Để làm được như vậy, kiểm tra xem liệu mỗi giá trị non-zero $M_T(m, n)$ có lớn hơn giá trị lân cận nó theo hướng gradient $\theta(m, n)$ hay không. Nếu có, giữ giá trị $M_T(m, n)$ không thay đổi, còn không thì gán nó bằng 0.

Bước 5: Sử dụng hai ngưỡng τ_1 và τ_2 ($\tau_1 < \tau_2$) lọc kết quả ở bước trước để có được ảnh đơn màu T_1 và T_2 . Ảnh đơn màu T_2 ít nhiễu và ít cạnh sai hơn T_1 , nhưng có khoảng ngắt đoạn lớn hơn giữa các đoạn cạnh của ảnh so với ảnh T_1 với τ_1 .

Bước 6: Nối các đoạn cạnh của ảnh T_2 để hình thành một cạnh liên tục. Bằng cách từ điểm cuối của từng đoạn cạnh trên ảnh T_2 , ta tìm vùng lân cận của nó trên ảnh T_1 một đoạn cạnh để nối giữa khoảng ngắt T_2 đến khi nào bắt gặp đoạn ảnh khác trên ảnh T_2 .

3.2.4. Thuật toán Hough

Thuật toán xác định đường thẳng HoughLine là chiếm vai trò chính yếu trong việc xác định làn đường. Nó cũng rất hay sử dụng trong nhiều ứng dụng thị giác máy tính. Nguồn gốc của thuật toán chuyển đổi Hough là dùng để xác định đường thẳng. Hơn thế, sau này nó còn dùng trong việc xác định đường tròn, đường elip, v.v...

Biến đổi Hough là một kỹ thuật thường sử dụng nhận dạng một hình dạng cụ thể như đường thẳng, hình tròn, hình tam giác, v.v... trong một ảnh.

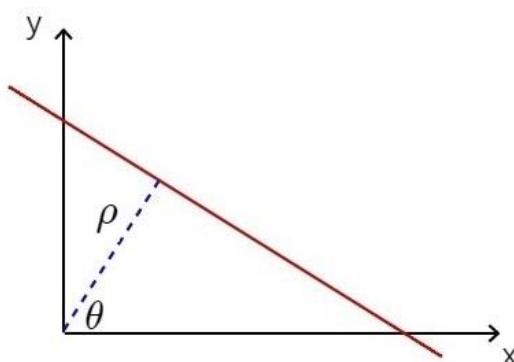
Biến đổi Hough ngày nay được ứng dụng nhiều trong lĩnh vực, đặc biệt trong lĩnh vực y học. Ưu điểm chính của kỹ thuật biến đổi Hough là nó nhận dạng một cách toàn cục nên không bị ảnh hưởng bởi nhiễu ảnh.

Trong quá trình xử lý ảnh, một vấn đề cơ bản nhưng rất thường gặp là phải nhận dạng các đường thẳng, hình tròn hay các đa giác. Thông thường, ta phải tiền xử lý ảnh trước bằng thuật toán phát hiện biên, và các góc từ đó nhận dạng ra hình dạng của vật. Tuy nhiên, nếu trong ảnh đó, do nhiễu nên mất đi một hay nhiều pixel khiến biên của vật không liên tục thì biên của vật cũng không liên tục, dẫn đến nhiều khó khăn. Vì lý do này, biến đổi Hough ra đời.

Về cơ bản, biến đổi Hough dựa vào việc ước tính các tham số từ các đường ranh giới, rồi trên cơ sở xác định các tập hợp các tham số nào thỏa một ngưỡng cho phép. Trên cơ sở này, phép biến đổi Hough có thể nhận dạng rất đa dạng các hình dạng khác nhau.

Trong không gian ảnh, một đường thẳng có phương trình: $y = m \cdot x + b$ và được vẽ bằng các điểm ảnh (x, y) thỏa phương trình.

Ta có hai tham số m và b . Từ đó một đường thẳng sẽ trở thành 1 điểm (m, b) trong không gian tham số. Tuy nhiên, phương trình này không tốt cho việc tính toán do trường hợp các đường thẳng thẳng đứng sẽ làm cho tham số m và b trở nên không xác định. Vì thế, để tiện việc tính toán, phép biến đổi Hough chọn 2 tham số khác cho đường thẳng là ρ (Rho) và θ (theta).



Hình 3.15 Biểu diễn đường thẳng với ρ và θ

Khi đó phương trình đường thẳng được viết lại như sau:

$$y = \left(-\frac{\cos\theta}{\sin\theta} \right) x + \left(\frac{\rho}{\sin\theta} \right)$$

Hay ở dạng khác là :

$$\rho = x \cdot \cos\theta + y \cdot \sin\theta$$

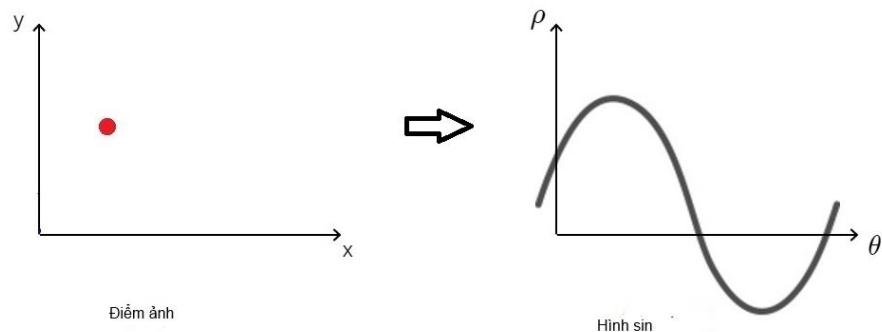
Trong đó, ρ là khoảng cách từ đường thẳng tới gốc tọa độ và θ là góc giữa vector từ gốc giữa pháp tuyến của đường thẳng với trục hoành.

Không gian Hough – không gian tham số ở trường hợp này là mặt phẳng (ρ, θ) với $\rho \in R$ và $\theta \in [0, \pi]$.

Ta có, một đường thẳng ở không gian (x, y) khi biến đổi Hough sẽ là 1 điểm trong không gian tham số (ρ, θ) và ngược lại.

Tập hợp các đường thẳng qua một điểm trong không gian ảnh (x_0, y_0) khi biến đổi Hough sẽ trở thành một đường cong hình \sin trong không gian tham số và tuân theo phương trình:

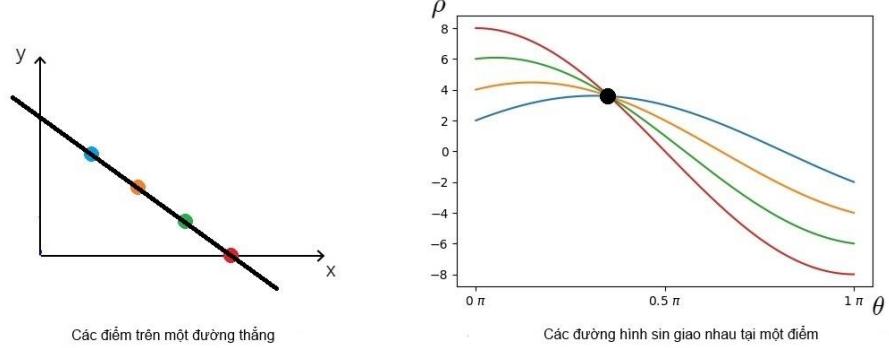
$$\rho(\theta) = x_0 \cdot \cos\theta + y_0 \cdot \sin\theta$$



Hình 3.16 Phép chuyển đổi một điểm ảnh qua không gian tham số (ρ, θ)

Giả sử ta từ hai điểm trên không gian ảnh, thì giao điểm của hai đường cong tạo từ hai điểm ấy giao nhau tại 1 điểm trên không gian tham số. Điểm giao ấy chính là tham số của đường thẳng đi qua hai điểm ấy trong không gian ảnh.

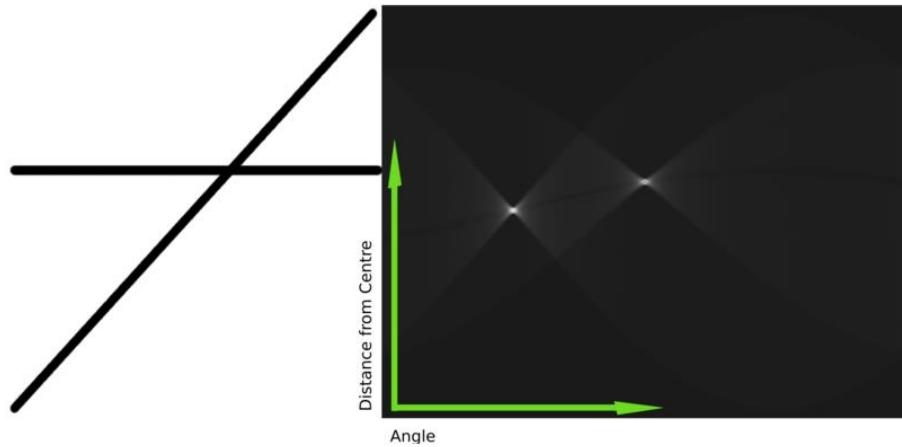
Tập hợp các điểm của một đường thẳng sẽ được biến đổi thành tập hợp các đường cong, giao điểm chung của các đường cong này chính là nghiệm tham số của bài toán.



Hình 3.17 Thí dụ phép chuyển đổi Hough cho nhiều điểm cùng nằm trên một đường thẳng

Trên cơ sở này, bài toán được chuyển từ việc tìm tham số của một đường thẳng sang tìm các giao điểm của các đường cong trên không gian tham số (ρ, θ) (không gian Hough).

Nhìn hình bên dưới, hai đường thẳng trong không gian (x, y) được chuyển đổi qua không gian Hough. Và ta dễ dàng nhận thấy có hai giao điểm được biểu diễn trên hình trong không gian Hough.



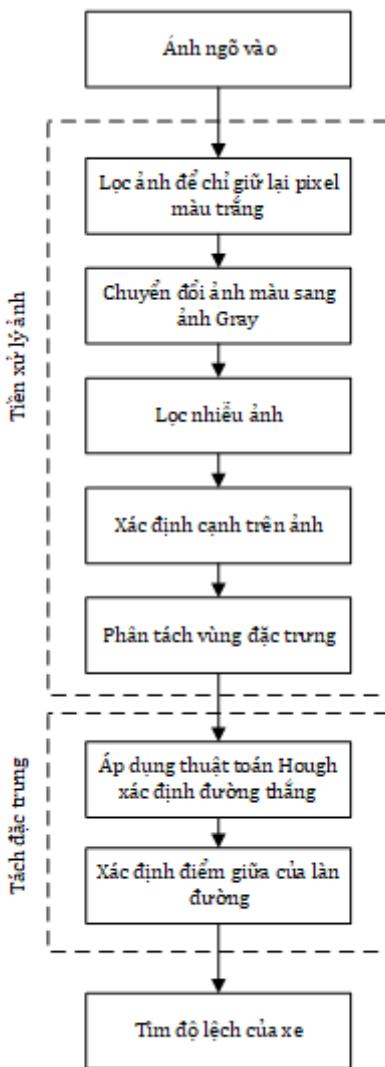
Hình 3.18 Biểu diễn hai đường thẳng trên không gian tham số

Chương 4. THUẬT TOÁN XÁC ĐỊNH LÀN ĐƯỜNG

4.1. Giới thiệu

Máy tính nhúng Raspberry Pi nhận hình ảnh từ Pi Camera trong thời gian thực với độ phân giải 640x480. Mỗi frame ảnh sẽ được chuyển đổi từ ảnh màu sang ảnh gray. Sau đó, sử dụng thuật toán xác định cạnh canny để xác định các cạnh trên ảnh. Dựa vào ảnh đơn màu từ thuật toán Canny, ta bắt đầu chạy thuật toán xác định đường thẳng Hough để xác định các đường thẳng đánh dấu của làn đường.

Về mô hình chung được mô tả như ở trên, chúng ta có thể xác định được làn đường trong một vài trường hợp. Nhưng để thực hiện được tốt hơn, đề tài áp dụng một số công đoạn như lọc ảnh đầu vào, lọc nhiễu trên ảnh gray, phân khúc ảnh hay tách vùng đặc trưng và lọc đường thẳng được định từ thuật toán chuyển đổi Hough.



Hình 4.1 Sơ đồ mô hình thuật toán xác định làn đường

Dựa vào sơ đồ mô hình trên, ta có thể nhóm các giai đoạn xử lý thành các bước như:

- Tiền xử lý ảnh.
- Sử dụng thuật toán Hough xác định đường thẳng, được gọi là bước trích tách đặc trưng của ảnh.
- Tìm độ lệch của xe.

Việc phân nhóm như vậy giúp ta có thể tập trung vào xử lý và lựa chọn thuật toán tốt nhất cho mô hình chung. Hệ thống thực hiện tốt khi ngõ ra của thuật toán xử lý từng bước trên.

4.2. Tiền xử lý ảnh

Tiền xử lý ảnh có hai mục đích chính: loại bỏ nhiễu và chuẩn bị ảnh cho bước kế tiếp. Nhiễu của ảnh bao gồm bóng tối, sự thay đổi màu đường, hay nhiễu ngẫu nhiên nào đó có trong ảnh. Ảnh có thể thu hẹp (cắt bớt) lại, chỉ giữ lại những phần cần thiết để giảm sự phức tạp trong tính toán và giảm nhiễu.

Như qui trình tiền xử lý ảnh đã được mô tả, ban đầu ảnh sẽ được chuyển đổi sang ảnh gray cho thao tác dễ dàng hơn, nó cũng là một cách để giảm nhiễu cho ảnh.



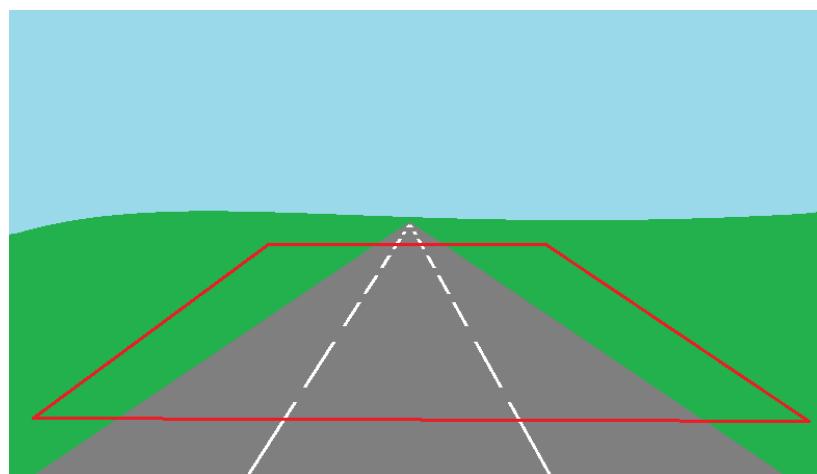
Hình 4.2 Ảnh gray

Một cách giảm nhiễu khác là áp dụng thuật toán làm mờ ảnh. Như đánh giá ở phần cơ bản xử lý ảnh, đề tài chọn cách làm mờ Gaussian (Gaussian Blurring) để áp dụng trong việc lọc nhiễu.



Hình 4.3 Ảnh gray sau khi được làm mờ guassian với kernel 5x5

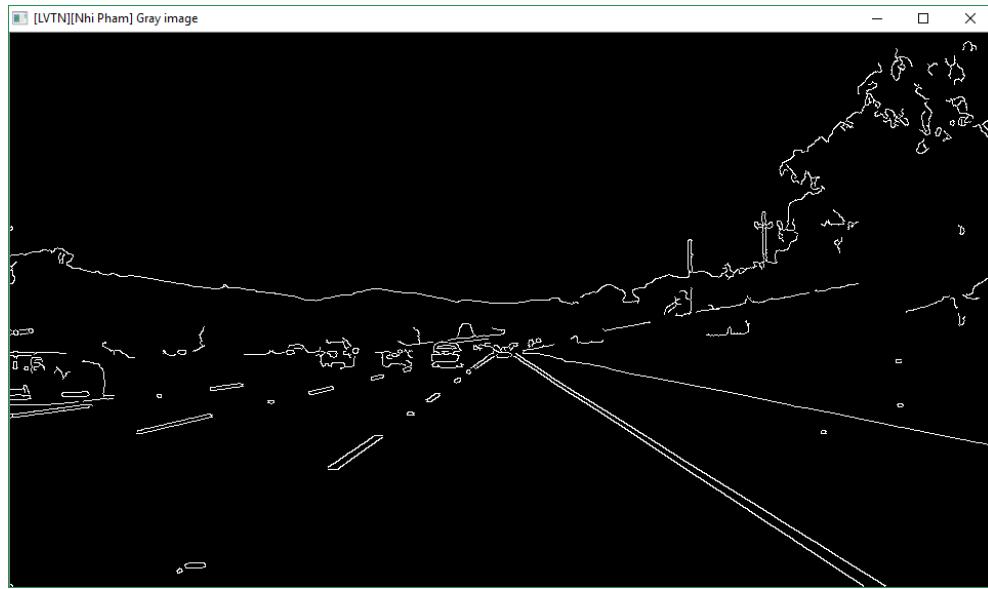
Để tăng tính hiệu quả của thuật toán, ảnh đầu vào được cắt giảm và chỉ tập trung vào phần chứa đường biên của làn đường, được định nghĩa là Region of Interest (ROI). Bởi kích thước ảnh giảm đi, bước kế tiếp ít để xử lý, do đó làm giảm độ phức tạp tính toán sau này. ROI là một hình thang với kích thước vừa đủ bao quanh đường biên của làn đường.



Hình 4.4 Region of Interest (ROI) bao quanh đường biên của làn đường

Một phần quan trọng trong việc trích xuất đặc trưng trong ảnh là xác định các cạnh trên ảnh. Nó là điểm mốc chốt để hệ thống tìm ra đường biên của làn

đường. Tôi xác định cạnh bởi sử dụng thuật toán xác định cạnh Canny, nó được biết là thuật toán tối ưu.



Hình 4.5 Ảnh được xác định cạnh bằng thuật toán Canny

Tôi sử dụng các ngưỡng (threshold) được tạo tự động cho Canny. Cách này sẽ tốt hơn việc xác định các giá trị cứng (hardcoded value). Chương trình tự động cập nhật thông số ngưỡng cho Canny như sau:

```
def edges(img):
gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
v = np.median(gray_img)
sigma = 0.33
```

4.3. Sử dụng thuật toán Hough xác định đường thẳng

Quá trình xử lý chi tiết trong bước xác định đường thẳng bao gồm:

- Sử dụng thuật toán Hough xác định các đường thẳng có trong ảnh.
- Lọc những đường thẳng nhiễu, những đường thẳng không thuộc đường biên trái hoặc biên phải.
- Phân loại đường thẳng là biên trái hoặc biên phải của làn đường.
- Tìm đường ra một đường thẳng duy nhất từ những đường thẳng thuộc biên trái hoặc biên phải.

Từ ảnh ngõ ra của bước tiền xử lý ảnh, ta áp dụng ngay thuật toán Hough để xác định các đường thẳng trong hình. Ngõ ra của thuật toán Hough là tập hợp

các đường thẳng có trong hình và mỗi đường thẳng được xác định qua giá trị của hai điểm ảnh trên đường thẳng đó (x_1, y_1) và (x_2, y_2) .

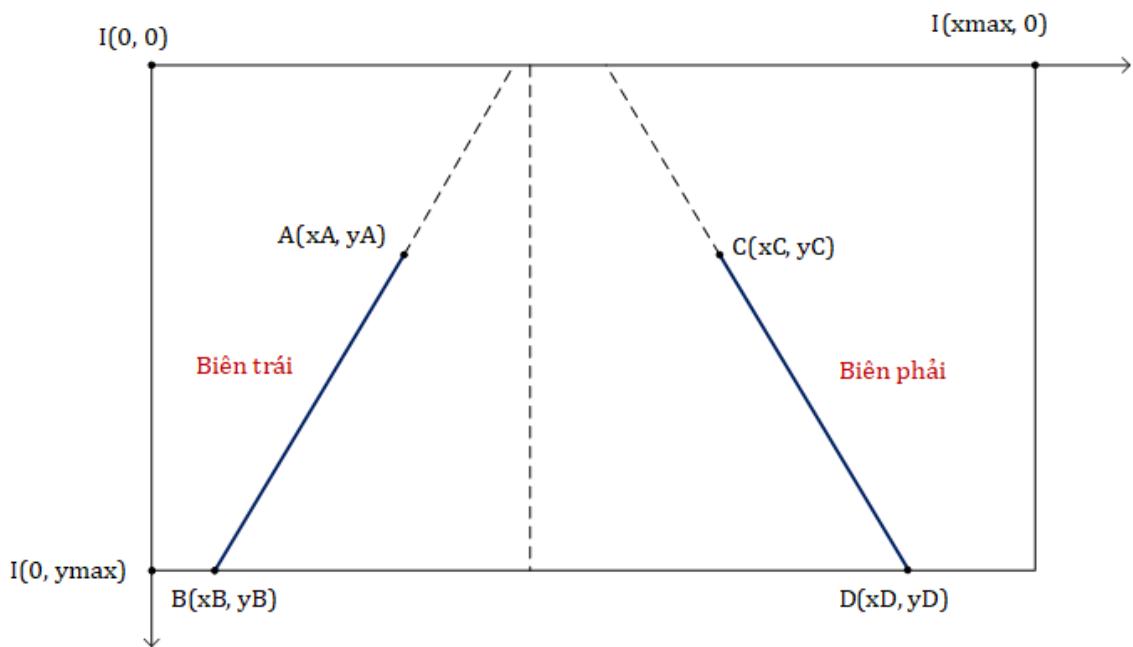


Hình 4.6 Các đường thẳng được xác định từ thuật toán Hough

Từ các đường thẳng được xác định trên, chúng ta cần lọc bỏ những đường thẳng có độ dốc (slope of line) lớn hơn 0.5 (xét trong không gian tọa độ của ảnh), và chỉ giữ lại những đường thẳng hai biên của làn đường. Việc xác định độ dốc của đường thẳng được thể hiện qua công thức sau:

$$slope = \frac{y_2 - y_1}{x_2 - x_1}$$

Tiếp theo, chúng ta cần phân loại đường thẳng thuộc biên trái hay biên phải của làn đường.



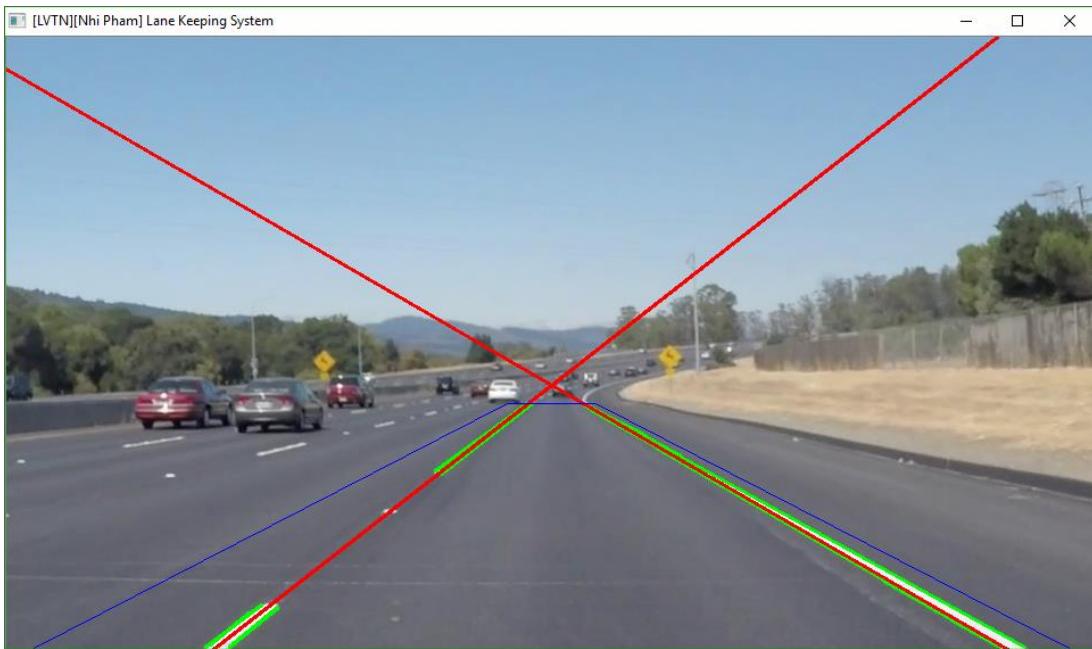
Hình 4.7 Phương pháp phân loại đường thẳng biên của lằn đường

Việc xác định này dựa vào tọa độ hai điểm được xác định ở trên. Giả sử ta có hai điểm có tọa độ (x_1, y_1) và (x_2, y_2) thì phương pháp phân loại được mô tả như sau:

Bảng 4.1 Phương pháp phân loại đường thẳng

	Độ dốc của đường thẳng	Tọa độ điểm (x_1, y_1)	Tọa độ điểm (x_2, y_2)
Biên trái	< 0	Bên phải đường tâm ảnh	Bên phải đường tâm ảnh
Biên phải	> 0	Bên trái đường tâm ảnh	Bên trái đường tâm ảnh

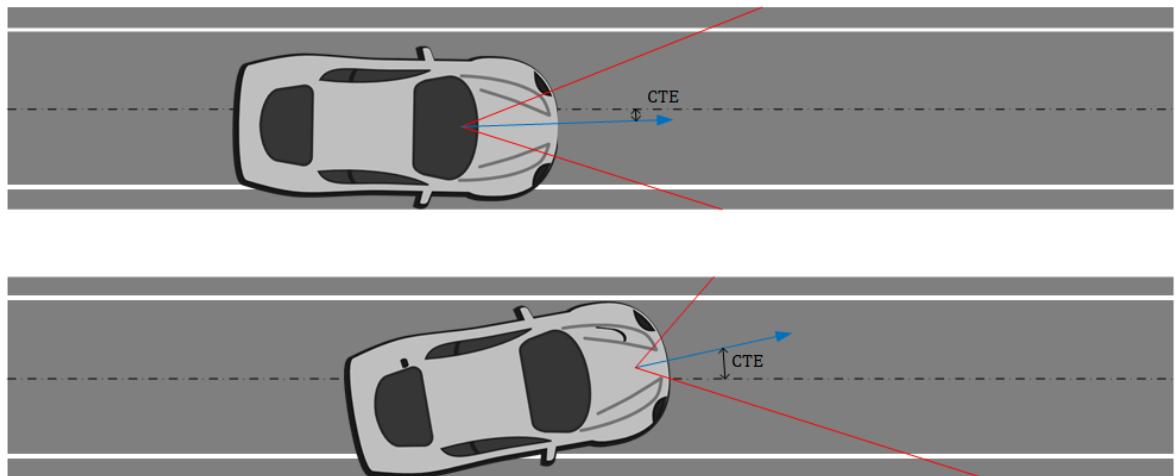
Từ những đường thẳng đã được phân loại, chúng ta tính nội suy đa thức để tìm chính xác một đường thẳng biên trái và một đường thẳng biên phải cho việc tìm độ lệch của xe ở bước kế tiếp. Đề tài này sử dụng hàm polyfit được hỗ trợ bởi thư viện *Numpy*, ngõ ra của hàm polyfit là giá trị (m, b) của phương trình đường thẳng $y = mx + b$.



Hình 4.8 Tìm giao điểm của hai đường thẳng biên của làn đường

Hai đường thẳng biên của làn đường được dùng để xét độ lệch của xe, do đó giai đoạn tìm hai đường thẳng biên chiếm vai trò quan trọng trong độ chính xác của các thuật toán về sau.

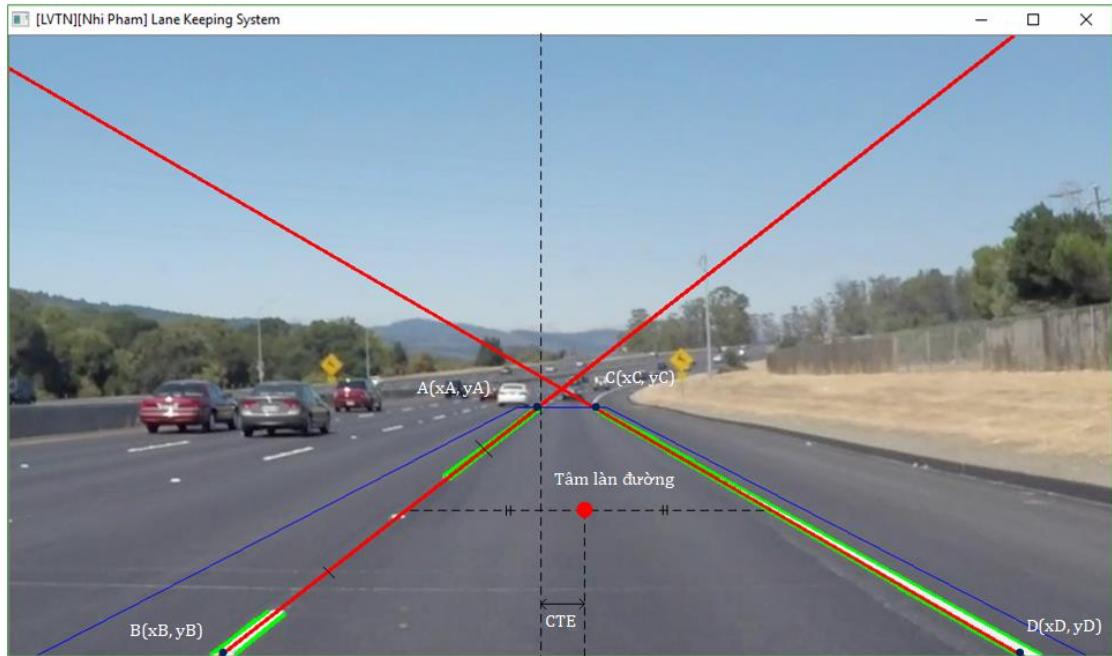
4.4. Phương pháp tìm độ lệch của xe



Hình 4.9 Hình minh họa xe đang lệch đường tâm làn đường một giá trị cte

Hình minh họa trên mô tả xe đang có xu hướng lệch khỏi làn đường, xe cần điều khiển bánh lái hợp lý để xe đi đúng chính giữa làn đường. Khoảng cách sai

số hay độ lệch được gọi là Cross Track Error (CTE) trong đề tài, được mô tả như hình sau:



Hình 4.10 Mô tả tính độ lệch của xe

Do camera của xe được gắn chính giữa xe nên tâm của xe bằng $\frac{1}{2}$ cạnh rộng của ảnh. Công thức tính tọa độ tâm làn đường dựa vào đường thẳng biên của làn đường trong không gian ảnh 2D như sau:

$$x_{center} = \frac{1}{4}(x_A + x_B + x_C + x_D)$$

$$y_{center} = \frac{1}{2}(y_A + y_B)$$

Trong đó, tọa độ điểm $A(x_A, y_A)$ và điểm $B(x_B, y_B)$ thuộc đường thẳng biên trái; và tọa độ điểm $C(x_C, y_C)$ và điểm $D(x_D, y_D)$ thuộc đường thẳng biên phải.

Cuối cùng, giá trị sai lệch của xe được tính bởi công thức:

$$CTE = \frac{image.shape[0]}{2} - x_{center}$$

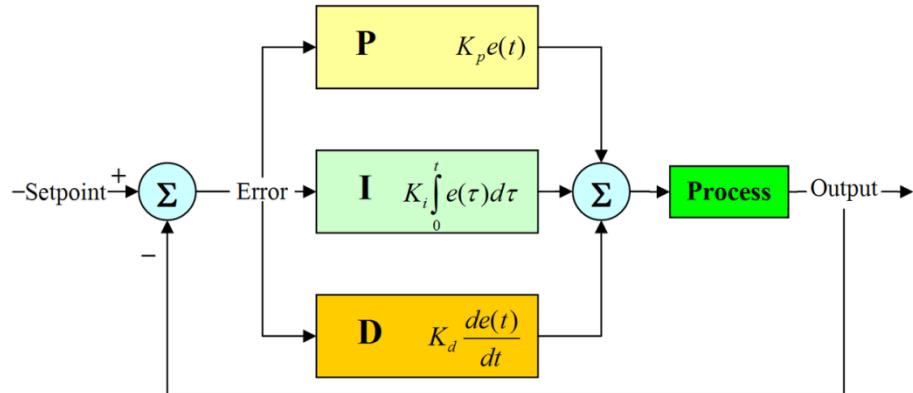
Trong đó $image.shape[0]$ là kích thước chiều rộng của ảnh đầu vào.

Chương 5. ÚNG DỤNG THUẬT TOÁN PID CHO ĐIỀU KHIỂN BÁNH LÁI

5.1. Tổng quan về bộ điều khiển PID

Tuy ra đời từ những năm 1890, nhưng đến nay, bộ điều khiển PID và các bộ PID cải tiến vẫn được sử dụng trong rất nhiều bộ điều khiển thuộc nhiều lĩnh vực như: điều khiển động cơ DC, điều khiển lò nhiệt, v.v...

PID là bộ điều khiển vi tích phân tỉ lệ (Proportional Integral Derivative) là một cơ chế phản hồi vòng điều khiển (bộ điều khiển) tổng quát được sử dụng rộng rãi trong các hệ thống điều khiển công nghiệp. Bộ điều khiển PID sẽ tính toán giá trị "sai số" là hiệu số giữa giá trị đo thông số biến đổi và giá trị đặt mong muốn. Bộ điều khiển sẽ thực hiện giảm tối đa sai số bằng cách điều chỉnh giá trị điều khiển đầu vào. Để đạt được kết quả tốt nhất, các thông số PID sử dụng trong tính toán phải điều chỉnh theo tính chất của hệ thống- các thông số phải phụ thuộc vào đặc thù của hệ thống.



Hình 5.1 Sơ đồ mô hình thuật toán PID

Giải thuật tính toán bộ điều khiển PID bao gồm 3 thông số riêng biệt, do đó đôi khi nó còn được gọi là điều khiển ba khâu: các giá trị tỉ lệ, tích phân và đạo hàm, viết tắt là P, I, và D. Giá trị tỉ lệ xác định tác động của sai số hiện tại, giá trị tích phân xác định tác động của tổng các sai số quá khứ, và giá trị vi phân xác định tác động của tốc độ biến đổi sai số. Tổng hợp của ba tác động này dùng để điều chỉnh quá trình thông qua một phần tử điều khiển như vị trí của van điều

khiển hay bộ nguồn của phần tử gia nhiệt. Nhờ vậy, những giá trị này có thể làm sáng tỏ về quan hệ thời gian: P phụ thuộc vào sai số hiện tại, I phụ thuộc vào tích lũy các sai số quá khứ, và D dự đoán các sai số tương lai, dựa vào tốc độ thay đổi hiện tại.

Bằng cách điều chỉnh 3 hằng số trong giải thuật của bộ điều khiển PID, bộ điều khiển có thể dùng trong những thiết kế có yêu cầu đặc biệt. Đáp ứng của bộ điều khiển có thể được mô tả dưới dạng độ nhạy sai số của bộ điều khiển, giá trị mà bộ điều khiển vọt lố điểm đặt và giá trị dao động của hệ thống. Lưu ý là công dụng của giải thuật PID trong điều khiển không đảm bảo tính tối ưu hoặc ổn định cho hệ thống.

Vài ứng dụng có thể yêu cầu chỉ sử dụng một hoặc hai khâu tùy theo hệ thống. Điều này đạt được bằng cách thiết đặt độ lợi của các đầu ra không mong muốn về 0. Một bộ điều khiển PID sẽ được gọi là bộ điều khiển PI, PD, P hoặc I nếu vắng mặt các tác động bị khuyết. Bộ điều khiển PI khá phổ biến, do đáp ứng vi phân khá nhạy đối với các nhiễu đo lường, trái lại nếu thiếu giá trị tích phân có thể khiến hệ thống không đạt được giá trị mong muốn.

5.2. Ứng dụng bộ điều khiển PID trong vi điều khiển

Hàm truyền PID liên tục:

$$u(t) = MV(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t)$$

Vấn đề đặt ra, hàm truyền của bộ PID liên tục không thể đưa vào vi điều khiển để tính toán (vì có khâu tích phân và khâu vi phân), chính vì vậy, ta phải tiến hành rời rạc hóa hàm truyền PID:

- Khâu tích phân được rời rạc hóa với thời gian lấy mẫu Δt như sau:

$$\int_0^{t_k} e(\tau) d\tau = \sum_{i=1}^k e(t_i) \Delta t$$

- Khâu vi phân được xác định bởi:

$$\frac{de(t_k)}{dt} = \frac{e(t_k) - e(t_{k-1})}{\Delta t}$$

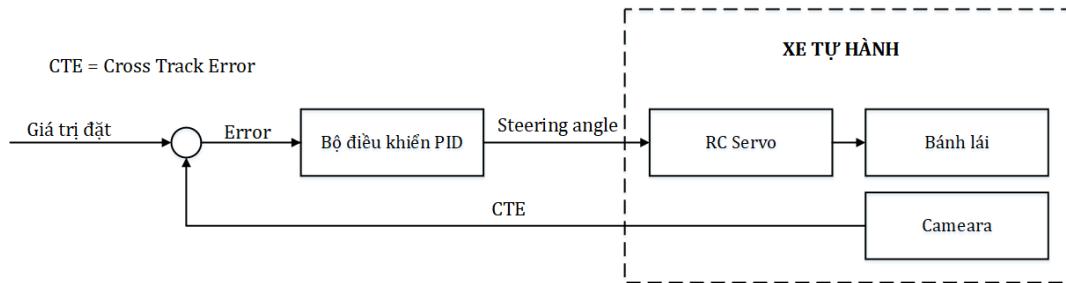
Cuối cùng, ta được hàm truyền PID rời rạc như sau:

$$u(t_k) = u(t_{k-1}) + K_p \left[\left(1 + \frac{\Delta t}{T_i} + \frac{T_d}{\Delta t} \right) e(t_k) + \left(-1 - \frac{2T_d}{\Delta t} \right) e(t_{k-1}) + \frac{T_d}{\Delta t} e(t_{k-2}) \right]$$

Với hàm truyền rời rạc trên, ta tiến hành viết code C để lập trình cho vi điều khiển thực hiện việc tính toán ngõ ra PID theo tín hiệu sai số ngõ vào và các thông số của bộ PID.

5.3. Ứng dụng bộ điều khiển PID vào điều khiển bánh lái

Thuật toán điều khiển PID được sử dụng trong bộ điều khiển bánh lái của xe tự hành để điều khiển xe di chuyển bám theo làn đường bằng cách áp dụng độ lệch của xe (được tính từ quỹ đạo giữa của làn đường so với vị trí của xe) – Cross Track Error. Bộ điều khiển PID giúp cho xe bám làn đường tốt hơn, mượt hơn và không bị dao động khi chạy.



Hình 5.2 Sơ đồ điều khiển bánh lái bằng PID.

Ngõ vào của bộ điều khiển PID là giá trị nhận được từ ngõ ra xử lý ánh của Raspberry Pi, CTE. Giá trị góc lái được tính từ bộ điều khiển PID được được dùng điều khiển bánh lái của xe tự hành.

Trong đề tài này, hệ thống lái cho bánh lái trước của xe được làm bằng động cơ RC Servo. Do đó, góc điều khiển lái và góc điều khiển servo được tính như sau:

$$\text{Angle}_{servo} = \text{Angle}_{steering} + \text{Centered_Angle}_{servo}$$

Trong đó, $Centered_Angle_{servo}$ là giá trị $Angle_{servo}$ khi xe bánh lái xe đang hướng thẳng, nó bằng 90° đối với mô hình robot được sử dụng trong đề tài này.

Các thông số kỹ thuật được sử dụng cho bánh lái (servo) như sau:

Bảng 5.1 Các thông số của bánh lái

Thông số	Giá trị
Độ phân giải góc lái	1°
Góc lái cực đại trái	145°
Góc lái cực đại phải	20°

Trong trường hợp, hệ thống bánh lái được nâng cấp sử dụng động cơ DC thay vì động cơ Servo thì chúng ta vẫn áp dụng được giá trị góc lái (steering angle) của bộ điều khiển PID.

5.4. Phương pháp tìm các hệ số của bộ điều khiển PID

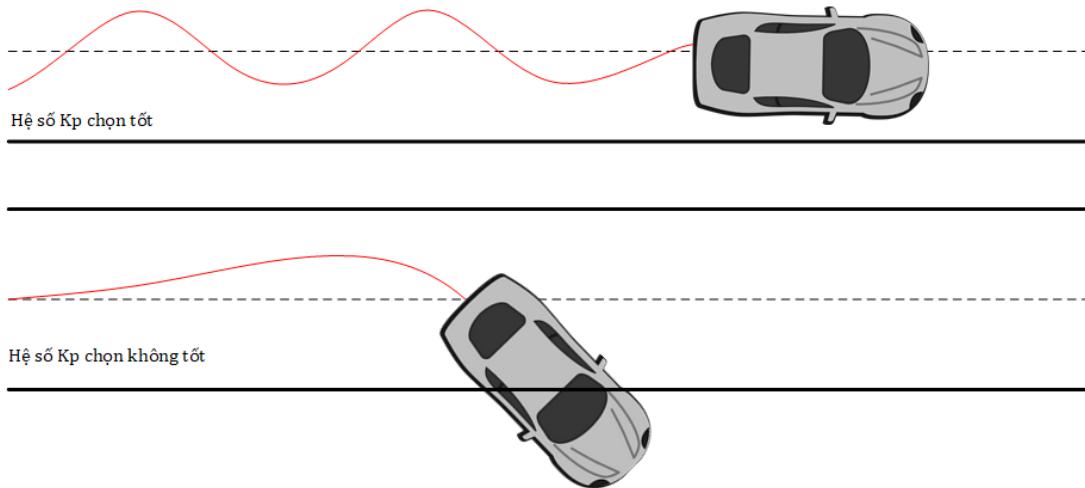
Như đã đề cập ở trên, bằng cách điều chỉnh 3 thông số của bộ PID (K_p , K_i , K_d) ta sẽ thay đổi được đáp ứng và chất lượng của ngõ ra để phù hợp với yêu cầu điều khiển.

Để đạt được kết quả tốt nhất, các thông số PID sử dụng trong tính toán phải điều chỉnh theo tính chất của hệ thống- các thông số phải phụ thuộc vào đặc thù của hệ thống.

Trong điều chỉnh các hệ số của bộ điều khiển chúng ta có thể điều chỉnh bằng phương pháp thủ công (offline), hoặc chúng ta áp dụng thuật toán điều chỉnh tự động như giải thuật Twiddle. Phương pháp điều chỉnh offline sẽ được thực hiện như sau:

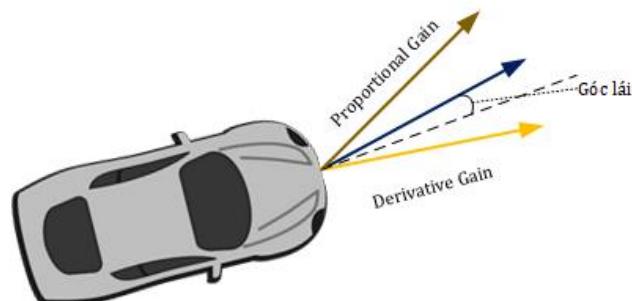
Đầu tiên, chúng ta điều chỉnh hệ số K_p đến khi nào quỹ đạo của xe dao động điều quanh đường tâm của làn đường. Nếu điều chỉnh không tốt hệ số K_p thì xe sẽ đi lệch khỏi làn đường.

$$\text{Góc lái} = K_p \cdot e_p$$



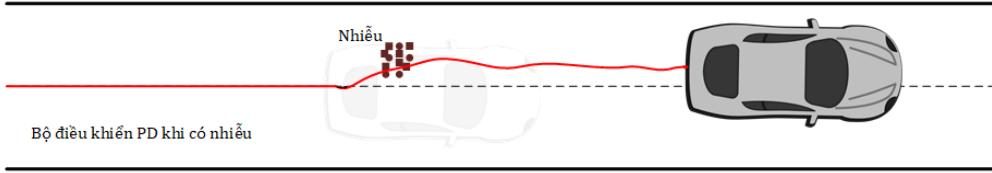
Hình 5.3 Quỹ đạo của xe với hiệu chỉnh hệ số Kp

Khi đó, ta tiếp tục điều chỉnh hệ số Kd (điều khiển PD). Khâu vi phân $K_d \cdot \frac{d}{dt} e(t)$ đại diện cho tốc độ thay đổi của sai số. Khâu này làm chậm lại tốc độ thay đổi đầu ra của bộ điều khiển, từ đó, điều khiển vi phân được sử dụng để giảm biên độ vọt lố và tăng cường bộ điều khiển hỗn hợp.



Hình 5.4 Mô tả bộ điều khiển PD.

Tuy nhiên, phép vi phân của một tín hiệu sẽ khuếch đại nhiễu và do đó khâu này sẽ nhạy hơn đối với nhiễu trong sai số.



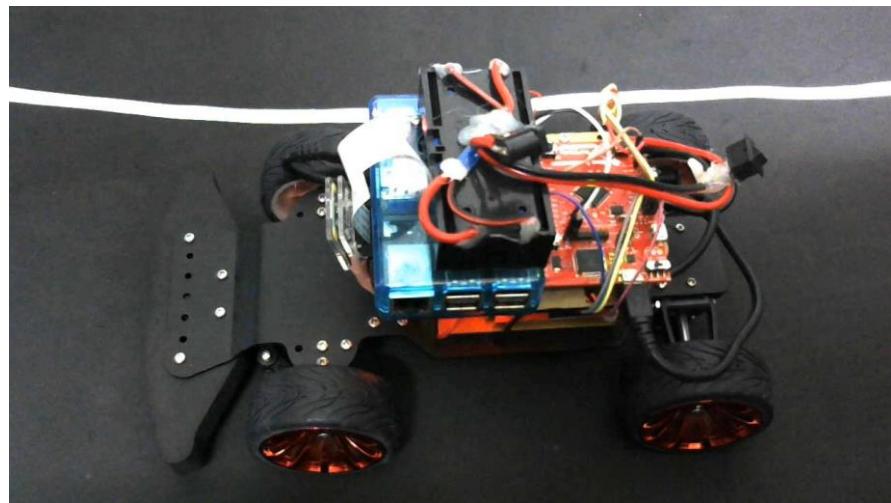
Hình 5.5 Bộ điều khiển PD khi có nhiễu.

Tuy nhiên, để hệ thống ổn định bộ điều khiển, ta điều chỉnh hệ số tích phân Ki thích hợp.

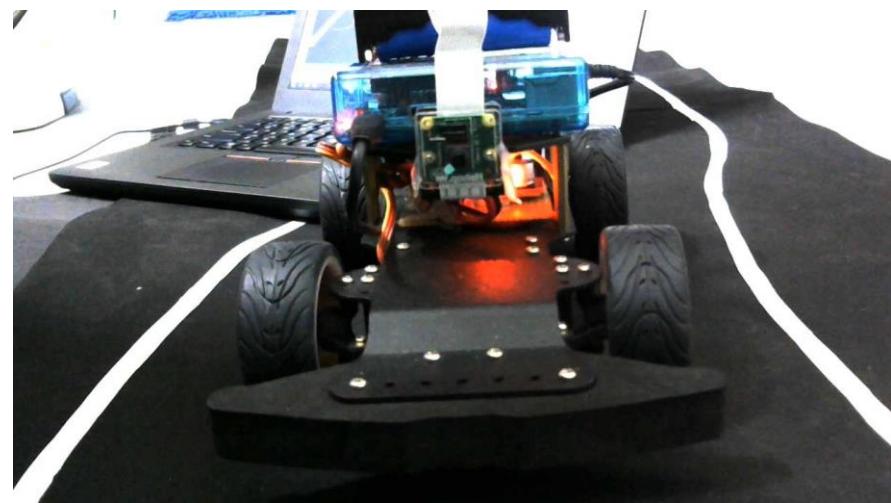
Chương 6. KẾT QUẢ THỬ NGHIỆM VÀ ĐÁNH GIÁ

6.1. Mô hình xe tự hành

Đề tài đã xây dựng được mô hình xe tự hành như mục tiêu ban đầu đặt ra.



Hình 6.1 Mô hình xe tự hành được xây dựng trong đề tài (top view).



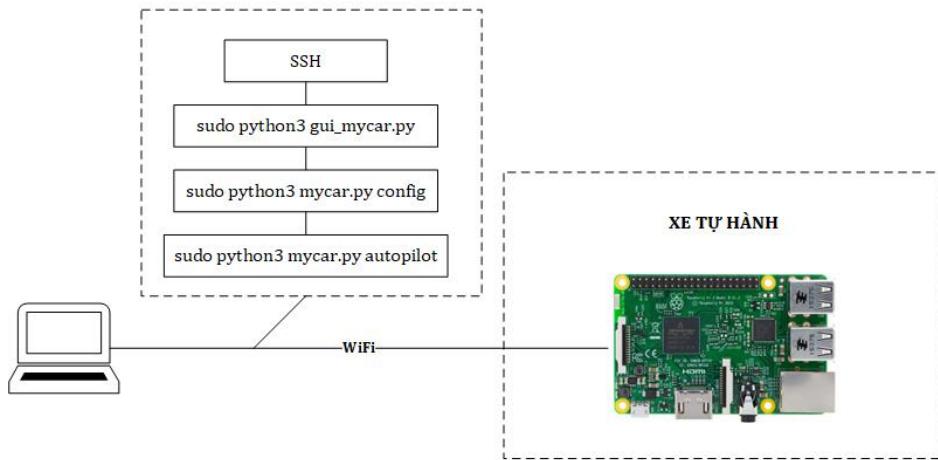
Hình 6.2 Mô hình xe tự hành được xây dựng trong đề tài (front view).

Với kết quả của mô hình này, ta tiến hành vận hành và thực hiện các thử nghiệm để kiểm tra thuật toán xử lý ảnh và thuật toán điều khiển bánh lái như mục tiêu ban đầu đề tài đặt ra.

6.2. Vận hành

Các công cụ được sử dụng trong đề tài như sau:

- Hệ điều hành Ubuntu 16.04 LTS.
- Phần mềm FileZilla dành cho truyền nhận file hoặc sử dụng *scp* trên Ubuntu.



Hình 6.3 Mô hình tương tác giữa máy tính PC và xe tự hành khi vận hành
Các bước thực hiện khởi động xe tự hành như sau:

Bước 1: Bật nguồn xe, và sử dụng máy tính (laptop) kết nối với Raspberry bằng SSH.

```
pi@raspberrypi: ~
nhivp@JACOB-PHAM:~$ ssh pi@192.168.19.196
pi@192.168.19.196's password:
Linux raspberrypi 4.9.59-v7+ #1047 SMP Sun Oct 29 12:19:23 GMT 2017 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Dec 23 11:53:22 2017 from 192.168.19.94

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

pi@raspberrypi:~ $
```

Hình 6.4 Màn hình Console kết nối với Raspberry Pi qua SSH.

Bước 2: Khởi động chương trình GUI Python để điều chỉnh xử lý ảnh.

```
sudo python3 gui_mycar.py
```

Bước 3: Chạy chương trình mycar.py trên Raspberry ở chế độ cài đặt.

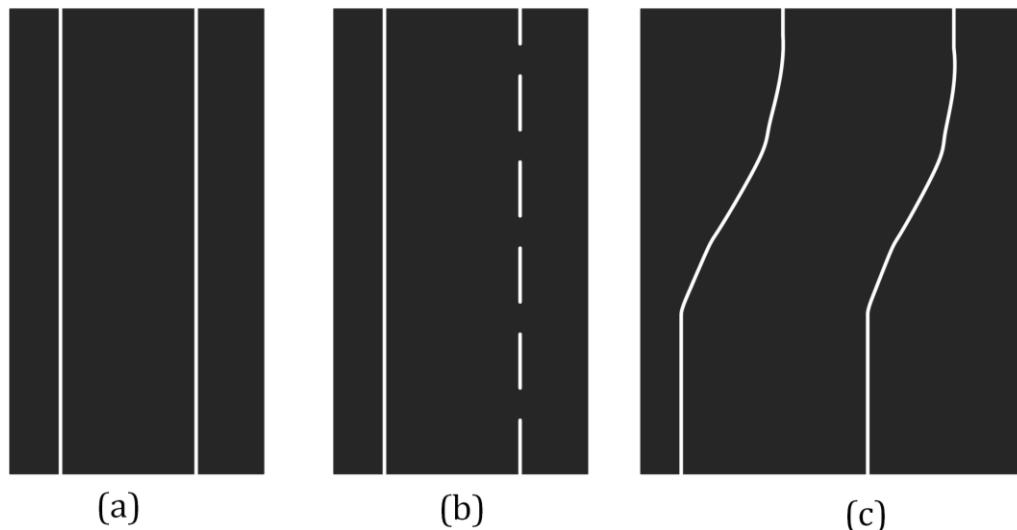
```
sudo python3 mycar.py config
```

Bước 4: Tắt chương trình Python, và chạy chương trình mycar.py ở chế độ xe tự hành.

```
sudo python3 mycar.py autopilot
```

6.3. Kết quả thử nghiệm

Các quỹ đạo của làn đường được thử nghiệm như sau:



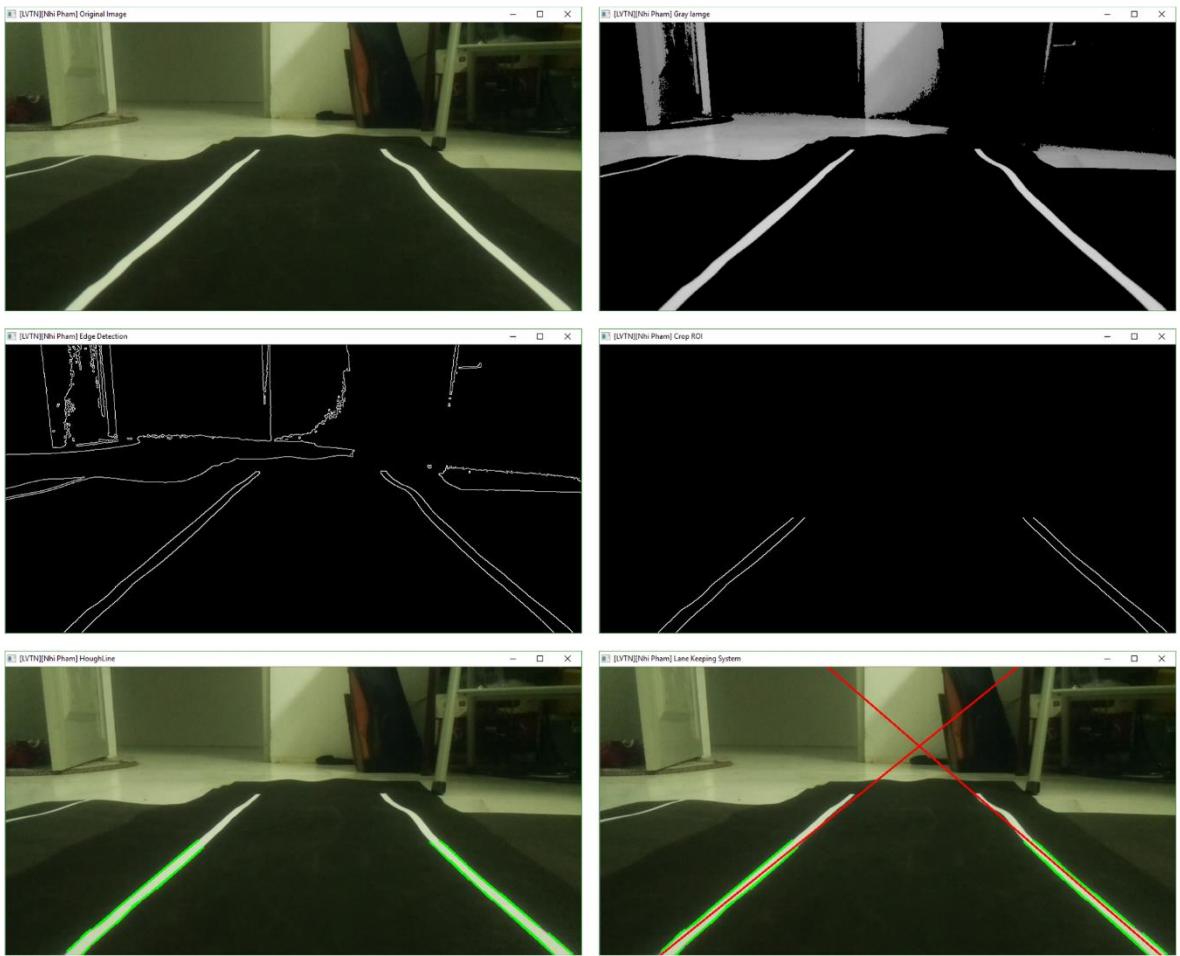
Hình 6.5 Các làn đường được sử dụng thử nghiệm: (a) làn đường thẳng; (b) làn đường có biên đứt; (c) làn đường cong

6.3.1. Kết quả xử lý ảnh xác định làn đường

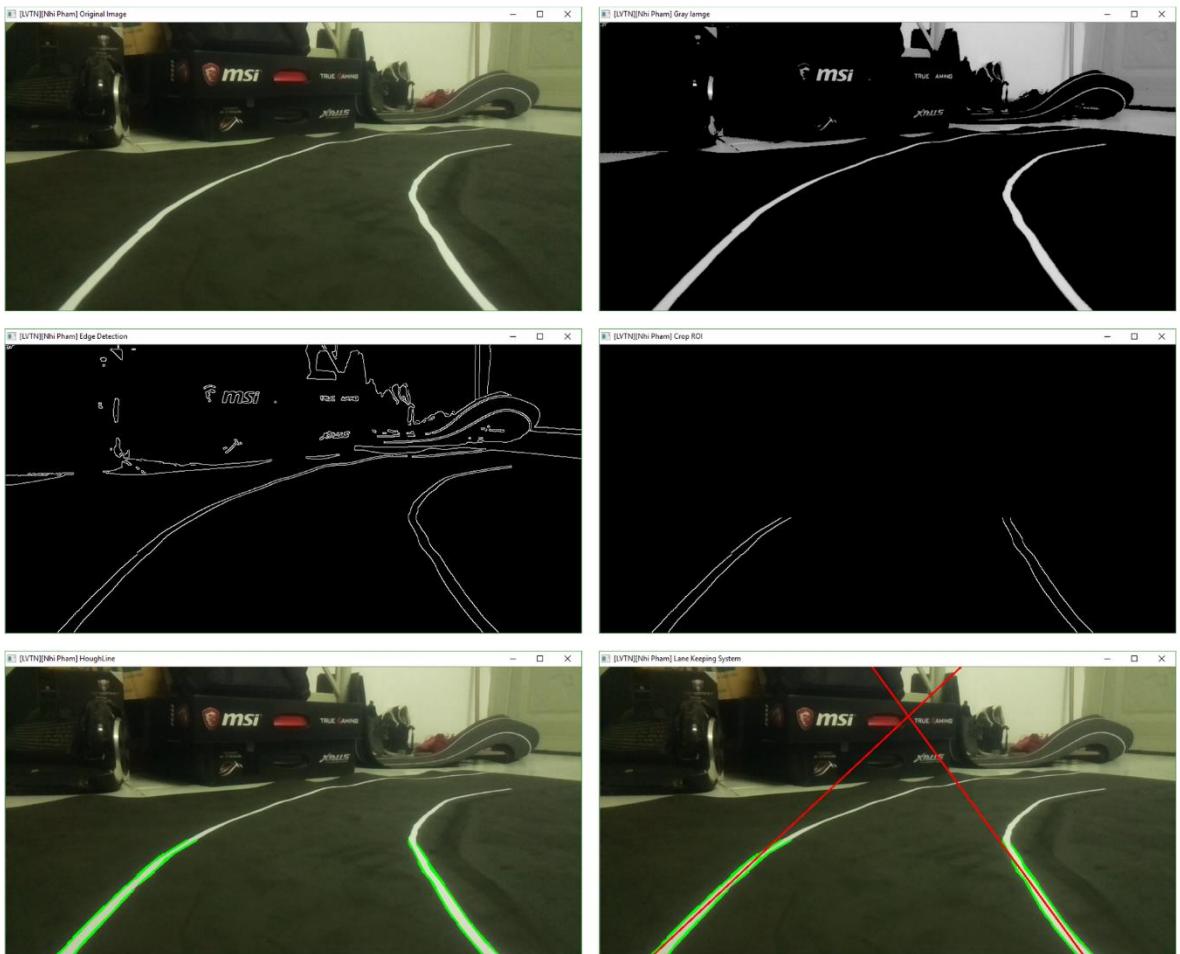
Tốc độ của xử lý ảnh:

Thử nghiệm	Thời gian xử lý (ms)
Mẫu 1	27.03
Mẫu 2	29.13
Mẫu 3	28.15
Mẫu 4	29.39
Mẫu 5	27.72
<i>Trung bình</i>	28.28

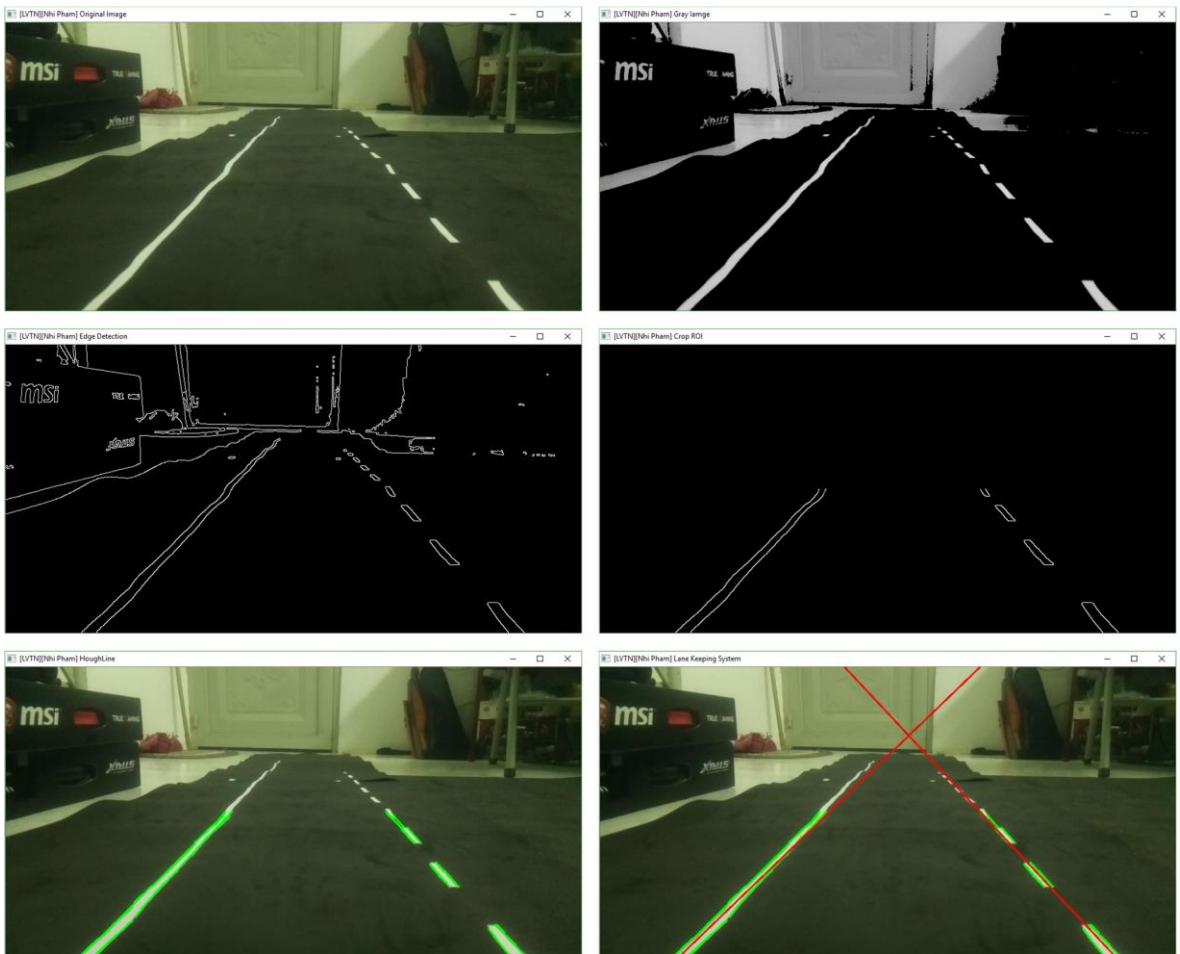
Sau đây là những kết quả thử nghiệm thuật toán xử lý ảnh:



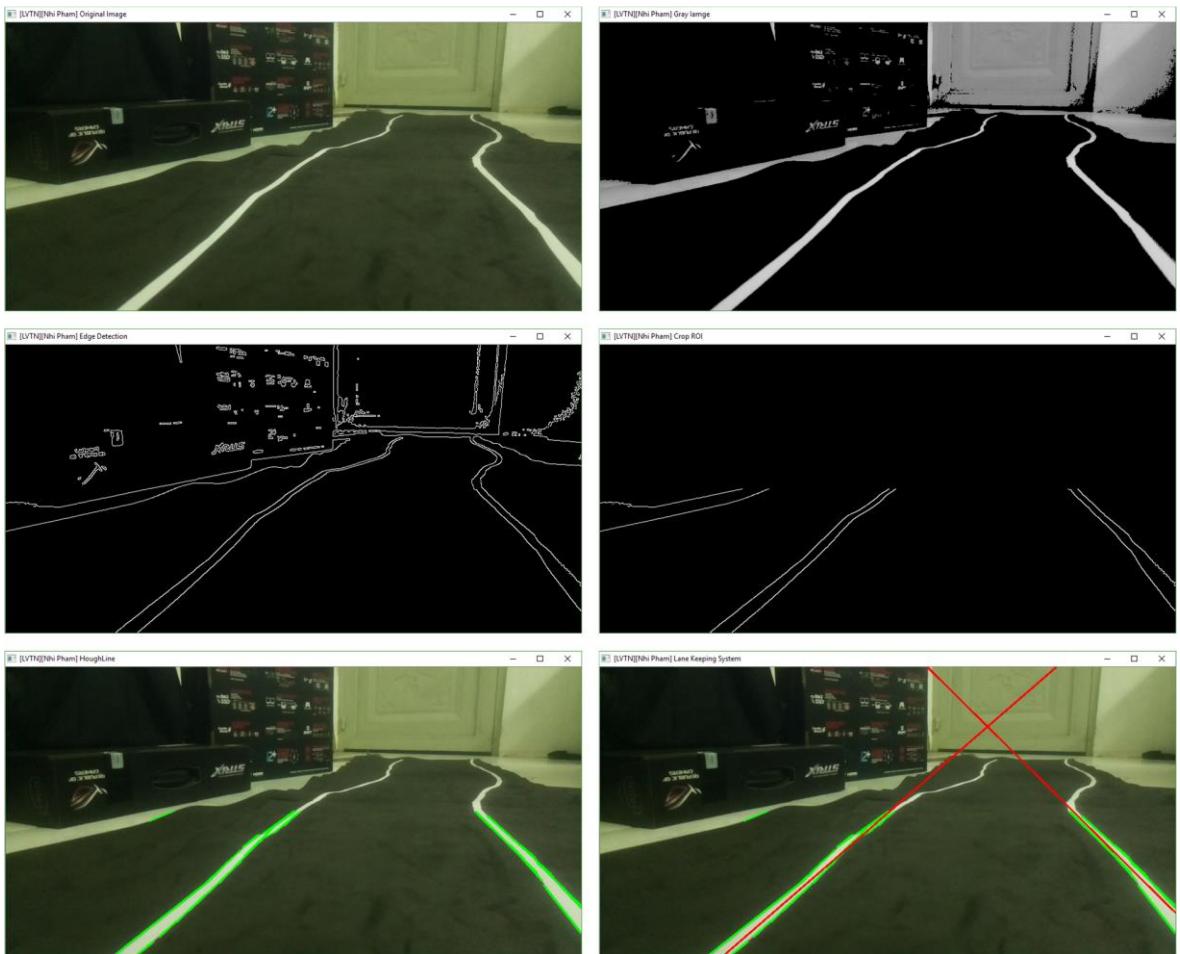
Hình 6.6 Thử nghiệm xử lý ảnh 1: Làn đường thẳng



Hình 6.7 Thử nghiệm xử lý ảnh 2: Làn đường cong



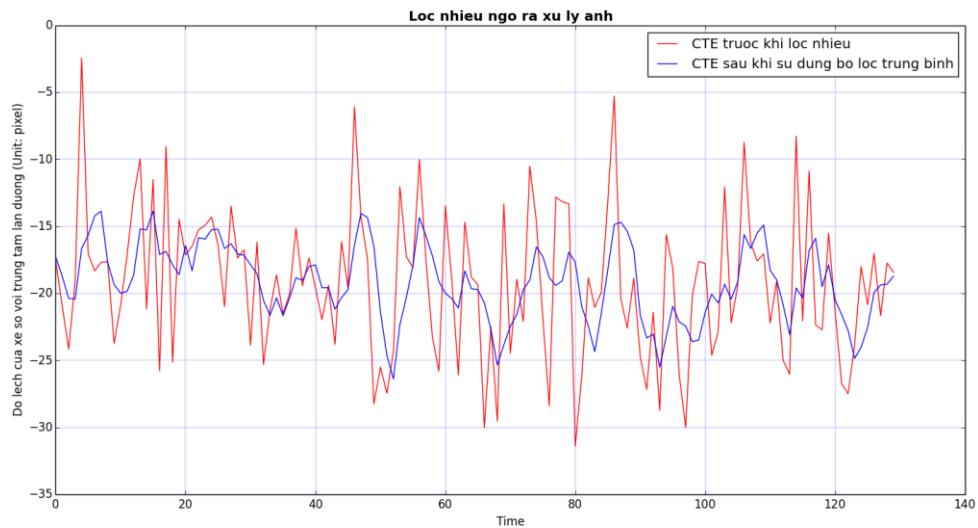
Hình 6.8 Thử nghiệm xử lý ảnh 3: Làn đường có một vạch biên đứt



Hình 6.9 Thử nghiệm xử lý ảnh 4: Xe có hướng lệch sang trái làn đường

Qua các thử nghiệm xử lý ảnh 1-4, đề tài đã được được kết quả như mục tiêu ban đầu đặt ra, nhận dạng chính xác hai vạch biên của làn đường. Xe có thể xác định được làn đường cong, làn đường vạch đứt. Ở thử nghiệm 4, ta thấy có những đường thẳng được xác định bởi thuật toán HoughLine, nhưng bởi có bộ lọc slope của các đường thẳng trước khi nội suy ra đường thẳng biên nên những đường thẳng nhiễu đó được loại bỏ.

6.3.2. Kết quả của bộ lọc ngõ ra xử lý ảnh



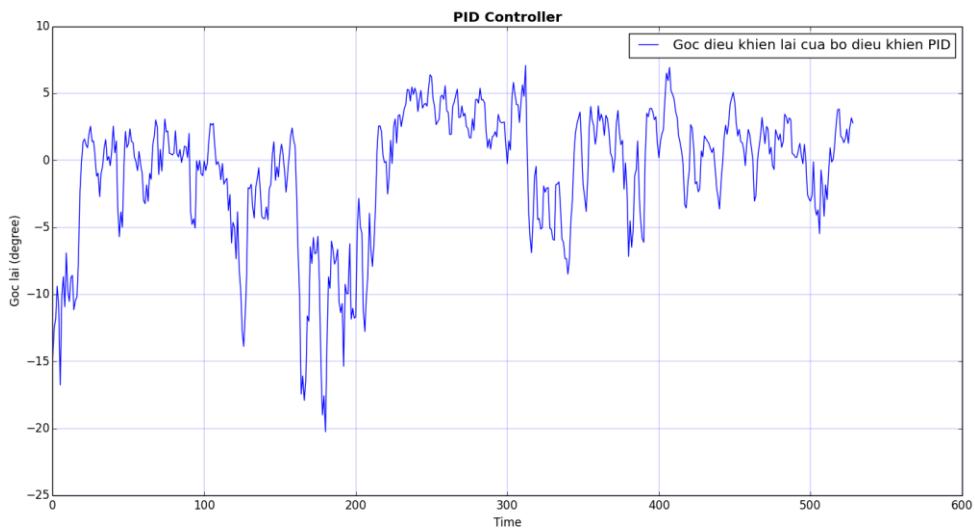
Hình 6.10 Đồ thị ngõ ra CTE của xử lý ảnh

Có nhiều xuất hiện trên ngõ ra của xử lý ảnh, nhưng nhiều nhỏ, không đáng kể. Để khắc phục, tôi sử dụng bộ lọc trung bình (average filter) để làm giảm nhiễu của ngõ ra xử lý ảnh. Một số lý do dẫn đến nhiễu của ngõ ra xử lý ảnh như sau:

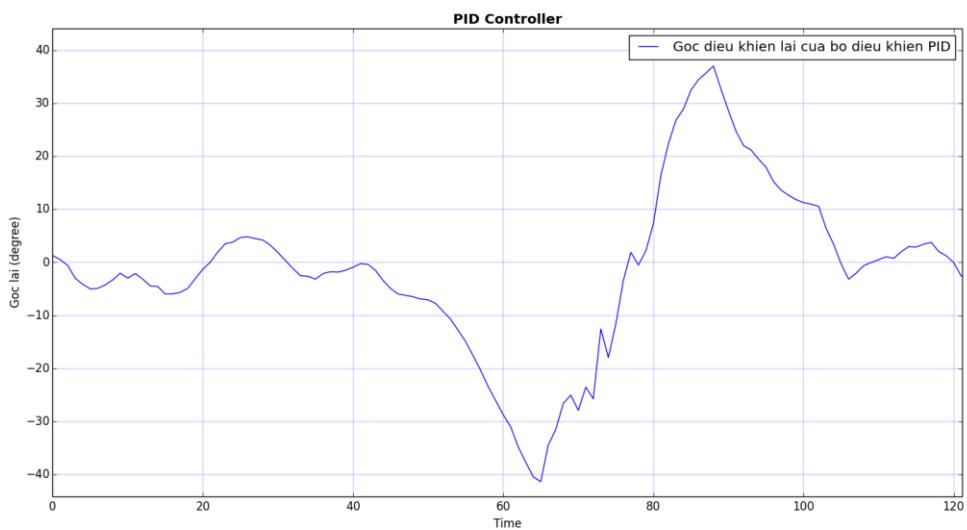
- Chất lượng cảm biến của camera.
- Độ sáng của đèn thay đổi, dẫn đến các frame ảnh khác nhau giữa các lần capture.
- Thuật toán nội suy đường thẳng *Polyfit*.

6.3.3. Kết quả bộ điều khiển PID

Hệ số PID được sử dụng trong thử nghiệm là: $K_p = 0.37$, $K_i = 0.0002701$, $K_d = 2.012$. Kết quả thử nghiệm PID trên làn đường thẳng và làn đường cong như sau:



Hình 6.11 Kết quả điều khiển PID khi xe di chuyển trên làn đường thẳng



Hình 6.12 Kết quả điều khiển PID khi xe di chuyển qua một đoạn đường cong (Hình 6.5c)

Nhận xét, các thông số bộ điều khiển chưa được tối ưu, vẫn còn dao động nhỏ ở trường hợp xe di chuyển trên làn đường thẳng.

Chương 7. KẾT QUẢ ĐẠT ĐƯỢC VÀ HƯỚNG PHÁT TRIỂN ĐỀ TÀI

7.1. Kết quả đạt được

Luận văn đã thực hiện được các nhiệm vụ đề ra:

- Xây dựng được mô hình phần cứng xe tự hành.
- Sử dụng và lập trình xử lý ảnh trên máy tính nhúng Raspberry Pi.
- Xây dựng thuật toán xác định làn đường nét liền và nét đứt.
- Áp dụng bộ điều khiển PID vào xe tự hành và điều chỉnh thông số cho bộ điều khiển.
- Xe tự hành có hoạt động được trên đường thẳng và đường cong < 30°.

Tuy nhiên, ngoài những mục tiêu đã đạt được, hệ thống cần phải cải thiện một số điểm để xe hoạt động tốt hơn:

- Đo đặc thông số xe chính xác để giảm tối thiểu các sai số do lắp đặt.
- Cần thêm bộ encoder để điều khiển vận tốc của xe, nhất là xe di chuyển ở các đoạn đường cong.
- Cần tối ưu thuật toán xử lý ảnh cũng như điều khiển PID để giảm điều chỉnh thông số khi thay đổi môi trường.
- Cải thiện quỹ đạo di chuyển của xe khi gặp các đoạn đường cong.

7.2. Hướng phát triển đề tài

Đề tài có thể mở rộng nghiên cứu sử dụng Deep Learning vào điều khiển bánh với ngõ vào hình ảnh lấy trực tiếp từ Camera. Ngoài ra, đề tài có thể mở rộng ứng dụng như nhận dạng biến báo giao thông, nhận dạng vật cản tĩnh và vật cản động, v.v...

Đề tài có thể nâng cấp và mở rộng để ứng dụng trên xe thực tế.

TÀI LIỆU THAM KHẢO

- [1] Nelson H. C. Yung and Andrew H. S. Lai, “*Effective Edge-Based Road Lane Detection*”, Vision Interface (VI1998), pp287-294 (1998).
- [2] Aharon Bar Hillel, “*Recent progress in road and lane detection: A survey*”, Machine Vision and Applications, pp727-745 (2014).
- [3] Sibel Yenikaya, Gokhan Yenikaya and Ekrem Duven, “*Keeping vehicle on the Road – A survey On-Road Detection Systems*”, ACM Computer Survey 46 (2013).
- [4] Sukriti Srivastava, Ritika Singal and Manisha Lumb, “*Efficient Lane Detection Algorithm using Different Filtering Techniques*”, International Journal Computer Applications (2014).
- [5] Pravin T. Mandlik and Prof A. B. Deshmukh, “*Raspberry-Pi Based Real Time Lane Departure Warning System using Image Processing*”, IJERT (2016).
- [6] Abdelhamid Mammeri, Guanqian Lu and Azzedine Boukerche, “*Design of Lane Keeping Assit Ssystem for Autonomous Vehicles*”, 2015.
- [7] Noor Cholis Basjaruddin, “*Lane Keeping Assit System Based on Fuzzy Logic*”, 2015.
- [8] Kyaw Ko Ko Htet, Tan Kok Kiong and Du Xinxin, “*Comprehensive Lane Keeping System with Mono Camera*”, 2015.
- [9] OpenCV Tutorial, website: docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html
- [10] Raspberry Pi, website: raspberrypi.org