ĐẠI HỌC QUỐC GIA TP. HCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



BÁO CÁO TỔNG KẾT

ĐỀ TÀI KHOA HỌC VÀ CÔNG NGHỆ SINH VIÊN NĂM 2018

*Tên đề tài tiếng Việt:*

Xây Dựng Xe Tự Hành Ứng Dụng Công Nghệ Xử Lý Ảnh

*Tên đề tài tiếng Anh:*

Building an Autonomous Car with Image Processing Technology

Khoa: Kỹ thuật Máy tính

Thời gian thực hiện: 06 tháng

Cán bộ hướng dẫn:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TT | Họ và tên, MSSV | Chịu trách nhiệm | Điện thoại | Email |
| 1 | Nguyễn Văn Phú Nhàn |  |  |  |
| 2 | Bùi Anh Khoa |  | 0772912970 | 15520364@gm.uit.edu.vn |
|  |  |  |  |  |

Thành phố Hồ Chí Minh – Tháng 12/2018

|  |  |  |  |
| --- | --- | --- | --- |
| LOGO DHCNTT -hinh.jpg | ĐẠI HỌC QUỐC GIA TP. HCM  TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN | Ngày nhận hồ sơ |  |
| Mã số đề tài |  |
| *(Do CQ quản lý ghi)* | |

BÁO CÁO TỔNG KẾT

*Tên đề tài tiếng Việt:*

Xây Dựng Xe Tự Hành Ứng Dụng Công Nghệ Xử Lý Ảnh

*Tên đề tài tiếng Anh:*

Building an Autonomous Car with Image Processing Technology

|  |  |
| --- | --- |
| *Ngày 17 tháng 12 năm 2018.*  Cán bộ hướng dẫn  *(Họ tên và chữ ký)* | *Ngày 17 tháng 12 năm 2018*  Sinh viên chủ nhiệm đề tài |

THÔNG TIN KẾT QUẢ NGHIÊN CỨU

# Thông tin chung:

- Tên đề tài:

Xây Dựng Xe Tự Hành Ứng Dụng Công Nghệ Xử Lý Ảnh

- Chủ nhiệm:

- Cơ quan chủ trì: Khoa Kỹ thuật Máy tính, Trường Đại học Công nghệ Thông tin.

- Thời gian thực hiện: 06 tháng

# Mục tiêu

# Lý do chọn đề tài

*Đánh giá tổng quan tình hình nghiên cứu thuộc lĩnh vực của đề tài*

*Ngoài nước*

Đã có nhiều hãng phát triển và thử nghiệm xe tự hành:

- Google: 23 chiếc.

- Tesla: 39 chiếc.

- Apple: 55 chiếc.

- …

Các hãng công nghệ lớn đã kể trên và nhiều hãng khác vẫn đang trong quá trình nghiên cứu và phát triển để ứng dụng xe tự hành vào đời sống thực tế. trong đó được tích hợp nhiều công nghệ như camera 360, stereo camera, vi xử lý, gpu tốc độ cao.

*Trong nước*

Hiện tại, ở Việt Nam cũng đã có nhiều công ty như FPT, Ikorn, Bosch, … đang có xu hướng nghiên cứu và phát triển xe tự hành. Mô hình nghiên cứu được triển khai và có tính lan tỏa mạnh, thu hút nhiều sự quan tâm của giới nghiên cứu khoa học công nghệ, với ước vọng tạo ra và ứng dụng xe tự hành trong tương lai ở đời sống xã hội Việt Nam.

Các nghiên cứu xe tự hành tại Việt Nam hiện nay đang tiến lên xe thực tế từ mô hình xe tự hành. Những mô hình demo này sử dụng cơ chế ngoại vi như servo để bẻ bánh lái và các động cơ điện 1 chiều để mô phỏng xe tự hành ngoài đời thực. Ứng dụng đang hiện thực hóa trên board chuyên biệt cho việc xử lý ảnh như board Nvidia và chỉ dừng lại ở phân đoạn nhận diện đường trắng liền, biển báo, vật cản.

# Vì vậy, các doanh nghiệp đang đẩy mạnh nghiên cứu, phát triển và sản xuất xe tự hành nhằm phục vụ nhu cầu người dùng, nâng cao kinh tế của công ty cũng như hướng đến thành phố thông minh.

# Thế nên quyết định chọn đề tài này để đón đầu công nghệ đang dần thịnh hành tại Việt Nam để đón đầu công nghệ

# Mục tiêu tổng quan

# Thiết kế mô hình xe tự hành dựa trên mô hình xe RC Car, ở đây sẽ dùng xe mang số hiệu: RGT ROCK HAMMER. Chiều dài khoảng 44cm và cân nặng khoảng 3-4kg. Thay đổi lò xo cho trục để độ nâng được tăng lên, giúp xe thăng bằng khi ôm cua những góc khá lớn.

# Xây dựng được các thuật toán, script, thư viện trên Nvidia Jetson TX2:

# Thuật toán nhận diện tâm giữa hai line nét đứt 2 bên:

# Nhận diện màu trắng, vàng, xanh lá (cỏ): Nhận diện 3 màu cùng một lúc.

# Lọc nhiễu với những vật thể khác: ví dụ bên ngoài line nhận thấy những vật thể lạ

# Dự đoán góc bẻ lái từ phương trình đường cong và lịch sử từ những frame ảnh trước đó: Phương trình bậc 2

# Lúc không có biển báo, tốc độ frame đạt tầm 30fps (thời gian thực).

# Thuật toán nhận diện biển báo (43 loại):

# Sử dụng công nghệ máy học Tensorflow của google

# Mạng neural VGG-16: Mạng với feature tốt làm cho kết quả trả về khá cao

# Độ chính xác với bộ test căn bản: 97%, độ chính xác này tạm đủ để nhận diện trên sa trường mô phỏng, chưa phải là tối ưu

# Lúc dự đoán biển báo, tốc độ frame đạt tầm 12fps

# Những script:

# Script chạy tự động dưới quyền admin

# Script cài đặt tự động

# Những thư viện:

# Thư viện Opencv bản 3.4.0 với GPU và CUDA – Kiểm tra mức độ chạy của GPU qua tergastat

# Thư viện Tensorflow 1.12 của Google: thư viện chuyên dùng cho mạng CNN

# Thư viện keras chính là backend của Tensorflow: Bao gồm những Sequential và Application built-in

# Thư viện SMBUS: sử dụng bus của I2C cho linux

# Thư viện I2C: Điều khiển board PCA9685

# Thư viện GPIO: Điều khiển input, output của led và nút bấm

# Xây dựng sa hình:

# Sa hình bao gồm:

# Line: bao gồm 2 lines đứt màu trắng, có khúc chia ra 2 đường phục vụ cho rẽ trái và rẽ phải

# Biển báo: bao gồm 2 biển báo căn bản là rẽ trái và rẽ phải

# Thử nghiệm thực trên sa hình:

# Xe chạy ổn với tốc độ 50-70%

# Có khả năng tăng giảm tốc khi nhận thấy biển báo

# Xe có khả năng tự ngắt chương trình và dừng động cơ khi lạc ra khỏi đường để giảm thiệt hại

# Mục tiêu cụ thể

# Thiết kế mô hình:

# Hình 1. Mô hình xe

# RGT ROCK HAMMER: RGT Rock Hammer là mẫu xe địa hình tỉ lệ 1/10 của hãng RGT Racing. Đây là dòng xe chuyên địa hình đầy sức mạnh, với thiết kế khung sườn kim loại bắt mắt và độ bền cao, hệ thống truyền động 4 bánh (4WD), servo nhông kim loại, bộ hộp số tạo lực kéo mạnh, bánh lớn, gầm cao giúp xe di chuyện ở những địa hình phức tạp (đá, cát, sình lầy hay cả nước...) với độ dốc lên tới 45 độ. Đây là mẫu xe phù hợp với những người thích thể loại cơ bắp, leo trèo khỏe, không cần tốc độ cao, thích chế độ các chi tiết trên xe cho giống xe thật.

# Thay đổi lò xo: dày hơn gấp đôi so với lò xo của nhà sản xuất

# Động cơ có chổi than

# Xây dựng thuật toán, script, thư viện:

# Thuật toán nhận diện lane đường:

# Đối với màu sắc (trắng, vàng, xanh lá):

# Sử dụng bộ chuyển đổi màu HSV

# Sử dụng bộ thư viện quét tìm những điểm thỏa yêu cầu 3 màu

# Tạo một frame mới với tất cả những phần tử đã quét được và gộp chung lại thành 1 frame với đủ 3 màu trên 1 frame

# Định vị tâm cho Lane đường

# Cân chỉnh camera:

# Bẻ line đường tìm được thành thẳng để phát hiện góc cong hay không

# Tính toán góc bẻ lái và dự đoán

# Đo từ điểm tâm tìm được đến trọng tâm của hai bánh xe sau

# Tính toán phương trình bậc 2 cho 2 lane đường

# Từ 2 phương trình tìm được tâm

# Tính góc lệch theo đường thẳng từ trọng tâm xe đến tâm và tự trọng tâm xe vẽ vuông góc lên

# Tổng hợp các tâm để dự đoán góc bẻ lái thông qua PID

# Lưu lịch sử cho những frame kế tiếp để dự đoán

# Khoảng cách 2 bên line đến rìa ảnh được lưu lại

# Dùng khoảng cách này để biết xe có bị lệch ra quá hay không

# Lọc biển báo trên frame ảnh

# Sử dụng màu sắc HSV

# Tính trung bình chênh lệch khoảng cách từ tâm (moment) tới các điểm rìa của contour để tìm ra 1 số và so sánh với mức mà hình tròn đạt được. Nhỏ hơn thì là hình tròn

# Dự đoán biển báo là biển báo nào:

# Sử dụng tensorflow-backend, train với VGG-16 model (16 lớp)

# 4 loại biển báo để test

# Thư viện điều khiển GPIO

# Test những chân hoạt động tốt

# Tìm tài liệu về các GPIO cho Linux

# Tìm kiếm tên file và folder của GPIO trên Jetson TX2

# Thư viện điều khiển motor

# Sử dụng I2C bus 0 hoặc I2C bus 1

# Sư dụng board PCA9685 12 kênh PWM

# Cân chỉnh PWM cho board PCA9685 với mức PWM phù hợp cho ESC motor controller

# Thư viện điều khiển Servo

# Sử dụng I2C bus 0 hoặc I2C bus 1

# Sư dụng board PCA9685 12 kênh PWM

# Cân chỉnh PWM cho board PCA9685 với mức PWM phù hợp cho ESC motor controller

# Xây dựng sa hình:

# Đường cong góc nhỏ

# Đường con rắn

# Đường cong góc lớn so với góc bẻ lái tối đa của xe RC

# Đường nét đứt

# Ngã rẽ

# Biển báo rẽ trái và rẽ phải

# Thử nghiệm thực tế trên sa hình:

# Thử nghiệm xe chạy với tốc độ 50%, 60%, 70%, 80%

# Thử nghiệm xe chạy với đường không có biển báo

# Thử nghiệm xe chạy với đường có biển báo

# Tính mới và sáng tạo

# Phân tích hiện trạng

# Tình hình nghiên cứu trong nước

* *Trong nước* 
  + Hiện tại, ở Việt Nam cũng đã có nhiều công ty như FPT, Ikorn, Bosch, … đang có xu hướng nghiên cứu và phát triển xe tự hành. Mô hình nghiên cứu được triển khai và có tính lan tỏa mạnh, thu hút nhiều sự quan tâm của giới nghiên cứu khoa học công nghệ, với ước vọng tạo ra và ứng dụng xe tự hành trong tương lai ở đời sống xã hội Việt Nam.
  + Các nghiên cứu xe tự hành tại Việt Nam hiện nay đang tiến lên xe thực tế từ mô hình xe tự hành. Những mô hình demo này sử dụng cơ chế ngoại vi như servo để bẻ bánh lái và các động cơ điện 1 chiều để mô phỏng xe tự hành ngoài đời thực. Ứng dụng đang hiện thực hóa trên board chuyên biệt cho việc xử lý ảnh như board Nvidia và chỉ dừng lại ở phân đoạn nhận diện đường trắng liền, biển báo, vật cản.

# Tình hình nghiên cứu quốc tế

* *Ngoài nước* 
  + Đã có nhiều hãng phát triển và thử nghiệm xe tự hành:
    - Google: 23 chiếc.
    - Tesla: 39 chiếc.
    - Apple: 55 chiếc.
    - …

Các hãng công nghệ lớn đã kể trên và nhiều hãng khác vẫn đang trong quá trình nghiên cứu và phát triển để ứng dụng xe tự hành vào đời sống thực tế. trong đó được tích hợp nhiều công nghệ như camera 360, stereo camera, vi xử lý, gpu tốc độ cao.

# Phân tích các công nghệ

# Thiết kế mô hình:

# Hình 1. Mô hình xe

# RGT ROCK HAMMER: RGT Rock Hammer là mẫu xe địa hình tỉ lệ 1/10 của hãng RGT Racing. Đây là dòng xe chuyên địa hình đầy sức mạnh, với thiết kế khung sườn kim loại bắt mắt và độ bền cao, hệ thống truyền động 4 bánh (4WD), servo nhông kim loại, bộ hộp số tạo lực kéo mạnh, bánh lớn, gầm cao giúp xe di chuyện ở những địa hình phức tạp (đá, cát, sình lầy hay cả nước...) với độ dốc lên tới 45 độ. Đây là mẫu xe phù hợp với những người thích thể loại cơ bắp, leo trèo khỏe, không cần tốc độ cao, thích chế độ các chi tiết trên xe cho giống xe thật.

* + **Thông số kỹ thuật**

Hãng: RGT Racing

Tên sản phẩm: Rock Hammer

Tỉ lệ: 1/10

Kích thước: 470x270x215mm

Chiều dài cơ sở: 350mm

Khoảng sáng gầm xe: 80mm

Bề rộng lốp: 52mm

Đường kính lốp: 135mm

Hex trục bánh: 12mm

Tần số điều khiển: 2.4GHz

ESC: Hobbywing WP1040 40A, chống nước

Motor: Brushed RC540-8020

Servo: 15Kg

Pin: Pin sạc 7.2V 2000 mAh

Khoảng cách điều khiển: khoảng 100-120m

Thời gian sạc: 3-4 tiếng

Thời gian chơi: khoảng 15-20 phút

Tốc độ: 25km/h

# Thay đổi lò xo: dày hơn gấp đôi so với lò xo của nhà sản xuất

# Động cơ có chổi than:

# Hình 2

# Mô-tơ chổi than là loại mô-tơ dùng chổi than chì. Mô-tơ chổi than sử dụng trong xe điện RC là loại Mô-tơ dùng điện một pha, gồm có hai dây đỏ và đen. Ưu điểm của Mô-tơ chổi than là giá xe điều khiển từ xa sản xuất thấp cho nên chúng ta có thể thấy những chiếc xe điện RC giá thành thấp có gắn động cơ chổi than.

# Nhược điểm của loại động cơ chổi than được làm bằng chổi than chì nên sau một khoảng thời gian sử dụng sẽ làm mòn chổi than, tuổi thọ kém, mô-tơ tiêu thụ điện lớn, công suất yếu hơn các loại mô-tơ không chổi than có cùng kích cỡ. Vì vậy những chiếc xe mô hình RC chạy loại mô-tơ này thường không có tốc độ cao.

# Vì vậy xe sẽ được nâng cấp bằng động cơ không chổi than

# Xây dựng thuật toán, script, thư viện:

# Thuật toán nhận diện lane đường:

# Đối với màu sắc (trắng, vàng, xanh lá):

# Sử dụng bộ chuyển đổi màu HSV

# Hình 3. HSV color wheel

# Sử dụng bộ thư viện quét tìm những điểm thỏa yêu cầu 3 màu

# Hình 4. Ví dụ của lấy vùng màu (In range)

# Tạo một frame mới với tất cả những phần tử đã quét được và gộp chung lại thành 1 frame với đủ 3 màu trên 1 frame

# Định vị tâm cho Lane đường

# Cân chỉnh camera: Thông thường camera sẽ đọc ảnh vào dưới dạng hơi cong vì thế cần bẻ thẳng lại. 🡺Dùng 1 loạt những ảnh bàn cờ vua phẳng để so sánh và bẻ frame ảnh từ camera

# Hình 5. Cân chỉnh frame ảnh

# Bẻ thẳng frame ảnh: Dùng hàm getPerspectiveTransform và hàm warpPerspective của OpenCV. Với nguồn là phần ảnh cần bẻ thẳng và đích là phần hình chữ nhật cẩn nới, bẻ thẳng ra

# Hình 6. Biến đổi ảnh

# Lấy biểu đồ những điểm thuộc làn đường:

# Theo chiều X, tính tổng những điểm thuộc làn đường

# Tổng hợp vào một biến

# Chia khung làn đường:

# Chia thành 9 khung chữ nhật bằng nhau

# Xác định số lượng điểm thuộc làn đường trong khung đó

# Từ đó xác định vị trí của từng khung chữ nhật

# Hình 7. Sliding window

# Tại đây ta sẽ lưu lại lịch sử các khung

# Tìm phương trình đường cong

# Dùng hàm polyfit để dùng tất cả những điểm của làn đường và thế vào phương trình để tìm ra ba hệ số a, b, c cho phương trình ax2 + bx + c = y

# Khi tính được 2 phương trình, fill làn đường để dùng cho sau này

# Hình 8. Skip sliding window

# Tính toán góc bẻ lái và dự đoán

# Đo khoảng cách từ camera tới trung bình tâm của 2 bánh sau

# Đo khoảng cách từ camera tới điểm trên ảnh ở thực tế

# Tính khoảng cách từ điểm đích tới trung bình tâm của 2 bánh sau

# Tính góc lệch bằng hàm tan trong lượng giác

# Hình 9. Tính góc

# Cân chỉnh góc bằng một hệ số nhân với hệ số tìm được do thực nghiệm trên sa hình thực tế, ở đây chọn 1.5 để góc lớn hơn

# Tính toán khoảng cách lệch so với 2 bên làn đường

# Sau khi lấy được phương trình 2 làn đường, thế vào công thức để tính lệch (curvation)

# Công thức:

# Hình 10. Công thức tính độ lệch 2 bên đường

# Hình 11. Kết quả

# Lưu lịch sử cho những frame kế tiếp để dự đoán

# Mỗi lần xử lý 1 frame ảnh, so sánh curvation

# Nếu khác nhau 🡺 Lấy kết quả của frame mới để xử lý và cho xe chạy

# Nếu giống nhau 🡺 Lấy kết quả của frame cũ để xử lý và cho xe chạy

# Sau mỗi lần tính toán góc lệch, tâm điểm thành công thì lưu vào lịch sử.

# Lọc biển báo trên frame ảnh

# Chuyển thành ảnh xám

# Hình 12. Grasyscale

# Làm mờ Gaussian

# Hình 13. Gaussian blur

# Lọc màu xanh ra riêng

# Hình 14. Lọc màu xanh

# Lọc ngưỡng trắng đen: Đưa về ảnh binary

# Hình 15. Threshold

# Tìm contours bằng hàm find contours

# Nếu diện tích lớn hơn 5000 🡺 dùng hàm approxPolyDP của opencv 🡺 Nếu mức độ xấp xỉ (approx) mà bé hơn 8, có nghĩa là những điểm của contour kết tinh ra một hình elipse

# Bắt đầu đưa vào máy học tensorflow để dự đoán

# Hình 16. Detect Elipse Object

# Dự đoán biển báo là biển báo nào:

# Tiền xử lý ảnh:

# Chuyển ảnh thành dạng mảng 1 chiều với đặc trưng mong muốn.

# Huấn luyện data:

# Thu thập khoảng 300 tấm ảnh về biển báo

# Dùng model VGG-16 (16 layers) để tạo ra mạng CNN và sau đó training cho bộ data

# Thử nghiệm với máy i7 4th generation, ram 8Gb, không có GPU thời gian train với 4 loại biển báo khoảng 45ph, và 43 loại biển báo khoảng 8h

# Train 10 thế hệ

# Hình 17. Training

# Dự đoán biển báo

# Gọi lại model VGG-16 🡺 Để chính xác hơn lưu model vào file Json, và lấy ra config lại từ file Json bên chương trình dự đoán.

# Gọi compiler

# Bắt đầu dự đoán từng frame ảnh sau khi đã qua bước tiền xử lý

# Hình 18. Prediction

# Thư viện điều khiển GPIO

# Những GPIO sử dụng:

# Jetson TX2:

# GPIO393

# GPIO394

# GPIO297

# GPIO388 🡺 Trở kéo xuống

# GPIO298 🡺 Trở kéo xuống

# GPIO467 🡺 Trở kéo xuống

# GPIO255 🡺 Trở kéo xuống

# Jetson TX1:

# GPIO10

# GPIO9

# GPIO36

# GPIO187 🡺 Trở kéo xuống

# GPIO186 🡺 Trở kéo xuống

# GPIO163 🡺 Trở kéo xuống

# GPIO511 🡺 Trở kéo xuống

# Nguyên tắc hoạt động:

# Gửi những đoạn text vào file GPIO tương ứng trong hệ điều hành Ubuntu 16.04 do nhà sản xuất cung cấp

# Thư viện điều khiển motor và servo

# Jetson TX2: Sử dụng I2C bus 0 🡺 địa chỉ 0x70

# Jetson TX1: Sử dụng I2C bus 1 🡺 địa chỉ 0x40

# Điều khiển bằng xung PWM: Khi gửi những giá trị vào bus I2C, board PCA9685 sẽ build ra xung PWM gửi cho motor

# 2 Thông số maximum và minimum được tìm qua thực nghiệm, đo đạc

# Board PCA9685 hoạt động với mức điện áp 5V

# Xây dựng sa hình:

# Đường nét đứt

# Đường cong góc nhỏ: Hình 20. Đường cong góc nhỏ

# Đường con rắn: Hình 21. Đường con rắn

# Đường cong góc lớn với độ lớn nhất so với mức độ bẻ lái tối đa của servo: Hình 22. Đường cong góc lớn

# Ngã rẽ và biển báo: Hình 23. Traffic sign

# Thử nghiệm thực tế trên sa hình:

# <https://www.youtube.com/watch?v=uwkWJDGMpa8>

# https://www.youtube.com/watch?v=YKzGS5qxAIQ

# Tính mới và sáng tạo của đề tài

# Ưu điểm:

# Xe được thiết kế dạng vượt địa hình, nên leo qua được vật cản chiều cao khoảng 30cm

# Phuộc nhún có lò xo được thay 🡺 Mức chịu nặng lên hơn 5kg.

# Pin được thay từ lithium sang lipo 🡺 Thời gian sử dụng động cơ lâu hơn

# Thư viện được viết trên C++ và Python 🡺 Đa dạng hóa ngôn ngữ 🡺 Dễ kế thừa và phát triển

# Với python, dễ dàng liên kết với tensorflow-backend, scikit-learn, Skimage, Openni 2

# Xe sử dụng tối đa GPU nên tốc độ xử lý được giảm thiểu đáng kể

# Camera của xe gắn cao 🡺 Mức độ nhìn đường được tăng lên, đưa vào những trường hợp giúp kỹ sư dễ dàng test, kiểm tra, viết mã lệnh.

# Pin có dung lượng lớn 🡺 xài hơn được 8h

# Sử dụng thuật toán lấy vùng màu, giảm bớt trường hợp bị mất làn nếu xài ngưỡng cứng

# VGG-16 là model với 16 lớp, khá nhiều và lượng ảnh train là hơn 300 ảnh 🡺 Độ chính xác khác cao (98%).

# Chân input với nút bấm có gắn tụ không phân cực, giảm hiện tượng dội nút, double click

# Thuật toán có lưu lịch sử, phòng trù cho những trường hợp vô định, out làn.

# Nhược điểm:

# Tốc độ chậm do là loại xe vượt địa hình

# Góc đánh lái nhỏ 🡺 Những đường cong thực tế khó xử lý.

# Xe chưa chạy được trong môi trường bất ổn định.

# Tóm tắt kết quả nghiên cứu:

# Đặc tả kỹ thuật

# Camera Orbbec Astra:

# Thông số kỹ thuật:

# Kích thước: 165 x 30 x 40 mm

# Cân nặng: 0.3 kg

# Khoảng cách: 0.6 – 8.0m (Optimal 0.6 – 5.0m)

# Kích thước ảnh chiều sâu:

# 640\*480 (VGA) @ 30FPS

# 320\*240 (QVGA) @ 30FPS

# 160\*120 (QQVGA) @ 30FPS

# RGB Image Size

# 1280\*960 @ 7FPS

# 640\*480 @ 30FPS

# 320\*240 @ 30FPS

# Field of View

# 60° horiz x 49.5° vert. (73° diagonal)

# Data Interface

# USB 2.0

# Microphones

# 2

# Operating Systems

# Windows 7/8/10, Linux, Android

# Power

# USB 2.0

# Software

# Astra SDK or OpenNI 2 or 3rd Party SDK

# Không gian màu sắc HSV:

# Không gian màu sắc HSV là một vòng màu sắc và được tạo nên từ việc dựa vào 3 số liệu chính của nó chính vùng màu, độ bão hòa của màu sắc và độ sáng của màu sắc

# HSV hay còn gọi là không gian màu HSB là một không gian màu dựa trên 3 thông số chính của không gian màu:

# H viết tắt của từ HUE có nghĩa là vùng màu

# S Viết tắt của từ SATURATION có nghĩa là độ bảo hòa màu.

# V viết tắt của chữ VALUE có nghĩ là giá trị hay độ sáng của màu sắc

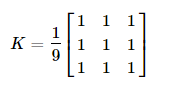
# Perspective Transform: Biến đổi phối cảnh

# Công thức: \texttt{dst} (x,y)= \texttt{src} (f_x(x,y), f_y(x,y))

# Phép ngoại suy của những điểm không tồn tại tạo ra một ảnh mới được “kéo dãn”.

# Hình 24. Ví dụ perspective transform

# Histogram: Biểu đồ phân bố tần số Histogram

* + Biểu đồ phân bố tần số (Biểu đồ phân bố mật độ, biểu đồ cột) dùng để đo tần số xuất hiện của một vấn đề nào đó, cho ta thấy rõ hình ảnh sự thay đổi, biến động của một tập dữ liệu. Đây là một khái niệm rất phổ biến, được sử dụng rộng rãi từ lĩnh vực kỹ thuật tới kinh tế.
  + Biểu đồ phân bố tần số (Biểu đồ phân bố mật độ, biểu đồ cột) dùng để đo tần số xuất hiện của một vấn đề nào đó, cho ta thấy rõ hình ảnh sự thay đổi, biến động của một tập dữ liệu. Đây là một khái niệm rất phổ biến, được sử dụng rộng rãi từ lĩnh vực kỹ thuật tới kinh tế.
  + Biểu đồ phân bố tần số (Biểu đồ phân bố mật độ, biểu đồ cột) dùng để đo tần số xuất hiện của một vấn đề nào đó, cho ta thấy rõ hình ảnh sự thay đổi, biến động của một tập dữ liệu. Đây là một khái niệm rất phổ biến, được sử dụng rộng rãi từ lĩnh vực kỹ thuật tới kinh tế.
  + Hình 25. Histogram
  + Trục tung (Oy) biểu diễn số lượng điểm ảnh (Pixel) của mức xám.
  + Trục hoành (Ox) biểu diễn mức xám.
  + Giá trị lớn nhất của trục hoành chính là số lượng điểm ảnh (Pixel) có trong một bức ảnh.
  + Với ảnh màu như RGB thì có tới 3 biểu đồ Histogram thể hiện từng kênh màu.
  + Một biều đồ tốt à biểu đồ có số lượng điểm ảnh nhiều nhất ở vùng giữa (Độ sáng trung bình) và ít dần ra 2 vùng sáng tối (Ngọn núi).
  + Dựa vào biều đồ Histogram mà bạn có thể biết được hình ảnh sáng tối như thế nào.
  + Áp dụng cho các xử lý ảnh cao cấp khác.
  + Hình 26. Phân tích Histogram
  + Ảnh tối là ảnh có tập trung quá nhiều điểm ảnh bên vùng tối.
  + Ảnh sáng là ảnh có tập trung quá nhiều điểm ảnh bên vùng sáng.
  + Ảnh có độ tương phản cao là ảnh có điểm ảnh tập trung nhiều ở 2 vùng sáng và tối và tập trung ít ở vùng giữa.
  + Ảnh có độ tương phản thấp là ảnh có điểm ảnh tập trung nhiều ở vùng giữa và tập trung rất ít ở vùng hai vùng sáng và tối.
* Polyfit:
  + Công thức: E = \sum\_{j=0}^k |p(x\_j) - y\_j|^2
  + Giải thích:
    - x[0]\*\*n \* p[0] + ... + x[0] \* p[n-1] + p[n] = y[0]
    - x[1]\*\*n \* p[0] + ... + x[1] \* p[n-1] + p[n] = y[1]
    - ...
    - x[k]\*\*n \* p[0] + ... + x[k] \* p[n-1] + p[n] = y[k]
  + Kết quả trả về phương trình bậc 2: ax^2 + bx + c = 0
* Gaussian blur: làm mờ
  + 
  + Những pixel ảnh sẽ nhân với K để ra giá trị mới là giá trị đã được làm mờ.
  + Với input nhập vào càng lớn, ảnh sẽ càng mờ.
  + Hình 27. Làm mờ Gaussian
* Machine learning:
  + Dưới góc nhìn của Machine Learning, mọi bài toán đều có input và output, ở giữa của quy trình đó là một hàm. Hàm là một khái niệm toán học khá quen thuộc với chúng ta. Hàm là một tập hợp các quy tắc được dùng để các phần tử ở tập nguồn đến các phần tử ở tập đích sao cho mỗi một phần tử ở tập nguồn chỉ có một và chỉ một phần tử ở tập đích. Trong đó, một bộ phận nhỏ của hàm, khi tất cả các phần tử của cả hai tập đều là dạng số học, người ta gọi đó là hàm số.
  + Hình 28. Machine learing – Hàm số
  + Cùng với sự phát triển của Computer Vision[Wiki], nhiệm vụ của người làm Machine Learning ở một mức cơ bản nào đó là giúp máy tính điện tử tìm ra được hàm quan hệ giữa tập input và tập output để từ đó máy tính có khả năng cải tiến hiệu xuất P của nhiệm vụ T thông qua trải ngiệm E. Hàm này được gọi là Hypothesis[Wiki] hoặc Model[Wiki]
  + Supervised Learning:
    - Học có giám sát là việc có sẵn một tập nguồn và một tập đích tương ứng để làm cơ sở xây dựng ra model mong muốn. Tập hợp kết hợp bởi hai tập này được gọi là tập Train[Wiki]. (Test sets và validate sets không được đề cập đến trong khuôn khổ bài viết này). Theo thời gian, khi đưa bài toán vào áp dụng thực tế, dữ liệu mới lại được sử dụng để trainHuấn luyện lại nhằm cải tiến model hiện tại. Điều này chính là mô tả cho khái niệm học có giám sát.
    - Các thuật toán nằm trong nhóm các bài toán học có giám sát được chia nhỏ thành hai phần dựa trên đặc tính của tập đích trong tập train.
  + Classification:
    - Khi dữ liệu của tập đích là một nhóm hữu hạn và có thể labledGắn nhãn được, bài toán được xếp vào dạng classification Bài toán phân loại. Ví dụ hay được sử dụng cho bài toán dạng này là bài toán gắn nhãn xem một email có phải là spam hay không, hoặc phân loại giữa cam và táo hay phân biệt chữ viết tay...
    - Hình 29. Classification
  + Regression:
    - Dạng còn lại được gọi là regressionBài toán hồi quy. Hồi quy là một thuật ngữ rất khó hiểu. Khi tìm hiểu cả bằng tiếng Trung và tiếng Nhật thì từ này trùng hợp là đều được dịch là hồi quy回帰 cả. Thì ra là những cái gì có tính liên tục và tiếp nối với nhau thì người ta gọi là là hồi quy. Vậy khi tập đích trong tập train là một tập dữ liệu có dạng liên tục không thể phân thành nhóm mà là một dữ liệu cụ thể thì bài toán được xếp vào dạng hồi quy. Bài toán tiêu biểu cho dạng này thường được ví dụ là tính toán giá cả của sản phẩm dựa trên thông số trước đó, hoặc dự đoán biến động tài chính...

# Thiết kế hệ thống

# <Trình bày 3 đến 5 trang, vẽ sơ đồ khối của hệ thống, nêu cách thức hoạt động của hệ thống, trình bày mối liên hệ giữa các khối nhỏ trong hệ thống, chú ý không để hình vẽ mà chỉ giới thiệu hình vẽ, hình vẽ sẽ đặt ở mục 7>

# 

# Thiết kế chi tiết

# Board chính được sử dụng trong xe tự hành sẽ là NVIDIA Jetson TX1. Sau khi thu thập và xử lý các thông tin thu thập được từ camera, board Jetson TX1 sẽ gởi tín hiệu điều khiển đến mạch điều khiển hệ thống xe (Shield). Shield sẽ có chức năng chuyển đổi tín hiệu điều khiển từ board TX1 thành các xung PWM để truyền tải đến các hệ thống điều khiển tương ứng như bẻ lái (Servo) và động cơ chạy xe (Motor 20A).

# Yêu cầu hệ thống: Nguồn ổn định: 20 Voltage, dòng ổn định ~ 6 Ampe

# Thư viện GPIO: tần số led 60Hz, có hệ thống chống dội ngược nút.

# Thư viện Servo: đảm bảo xung tín hiệu cung cấp liên tục, chia xung tỉ lệ với góc 30 độ.

# Thư viện ESC Motor: cung cấp tốc độ một lần duy nhất & đảm bảo tín hiệu truyền liên tục.

# Thư viện nhận diện đường đi: sử dụng Opencv.

# Thư viện nhận diện biển báo: sử dụng Tensorflow.

# Tích hợp hệ thống và kiểm thử

# Làm một sa trường thử nghiệm, độ lớn khoảng 15 mét vuông, đường đi con rắn, cong, thẳng, có ngã rẽ, biển báo.

# Yêu cầu: xe chạy được tốc độ tối thiểu 60%.

# Kết quả nghiên cứu

# 2 clip demo:

# <https://www.youtube.com/watch?v=uwkWJDGMpa8>

# <https://www.youtube.com/watch?v=YKzGS5qxAIQ>

# Tên sản phẩm

# Opencv: Thư viện xử lý ảnh open source

# Tensorflow: Thư viên hỗ trợ máy học

# Keras: Tensorflow Backend

# SMBUS: Thư viện I2C cho linux

# OpenNI: Thư viện hỗ trợ stereo camera

# Hiệu quả, phương thức chuyển giao kết quả nghiên cứu và khả năng áp dụng

# Xe chạy được mức độ ổn định là hơn 10 vòng lớn

# Đường có gờ vẫn xử lý ổn

# Hình ảnh, sơ đồ minh họa chính

# <Trình bày các hình vẽ đã được tham khảo trong các mục trên có kèm theo đánh số hình>

# C:\Users\Anh Khoa\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\10583BA6.tmp

# Hình 1. Mô hình xe

# mo-to-choi-than

# Hình 2. Động cơ có chổi than

# Image result for HSV opencv

# Hình 3. HSV color wheel

# Image result for opencv in range

# Hình 4. Ví dụ của lấy vùng màu (In range)

# Image result for camera calibration opencv

# Hình 5. Cân chỉnh frame ảnh

# Image result for opencv find lane with perspective transform

# Hình 6. Biến đổi ảnh

# Hình 7. Sliding window

# output_19_0.png

# Hình 8. Skip sliding window

# Image result for math tan

# Hình 9. Tính góc

# https://cdn-images-1.medium.com/max/800/1*y1qspIhVgBU41l7L_AZj9A.png

# Hình 10. Công thức

# https://cdn-images-1.medium.com/max/1000/1*dCtE2JeCmdjGZ6ulw6I71w.png

# Hình 11. Kết quả

# Image result for convert to grayscale opencv

# Hình 12. Grayscale

# Image result for gaussian blur opencv

# Hình 13. Gaussian blur

# https://www.pyimagesearch.com/wp-content/uploads/2014/07/color_detection_blue_version.jpg

# Hình 14. Lọc màu xanh

# Image result for threshold opencv

# Hình 15. Threshold

# Image result for opencv detect round objects

# Hình 16. Detect Elipse Object

# Image result for keras training

# Hình 17. Training

# Image result for road traffic signs recognition

# Hình 18. Prediction

# Image result for pca9685

# Hình 19. PCA9685

# 

# Hình 20. Đường cong góc nhỏ

# 

# Hình 21. Đường con rắn

# 

# Hình 22. Đường cong góc lớn

# 

# Hình 23. Traffic sign

# getperspective_transform_01

# Hình 24. Ví dụ Perspective transform

# ss_3

# Hình 25. Histogram

# ss_4

# Hình 26. Phân tích Histogram

# gaussian.jpg

# Hình 27. Làm mờ Gaussian

# https://s3-ap-southeast-1.amazonaws.com/kipalog.com/s5r5p56xhz_image.png

# Hình 28. Machine learning – Hàm số

# https://s3-ap-southeast-1.amazonaws.com/kipalog.com/yosp8xl7ys_image.png

# Hình 29. Classification

# https://s3-ap-southeast-1.amazonaws.com/kipalog.com/57f2tkc0pd_image.png

# Hình 30. Regression

|  |  |
| --- | --- |
| *Cơ quan Chủ trì*  *(ký, họ và tên, đóng dấu)* | *Chủ nhiệm đề tài*  *(ký, họ và tên)* |