

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP HCM**

**BÁO CÁO TỔNG KẾT
ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN**

**NGHIÊN CỨU THIẾT KẾ THI CÔNG MODULE GƯƠNG CAMERA
CẢNH BÁO TIỀN VÀ CHẠM KHI XE CHUYỂN HƯỚNG THÔNG
QUA XỬ LÝ ẢNH**

SV2022-150

Chủ nhiệm đề tài: Mai Đức Tùng

TP Hồ Chí Minh, 11/2022

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐH SƯ PHẠM KỸ THUẬT TP HCM**

**BÁO CÁO TỔNG KẾT
ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN**

**NGHIÊN CỨU THIẾT KẾ THI CÔNG MODULE GƯƠNG CAMERA CẢNH BÁO
TIỀN VÀ CHẠM KHI XE CHUYỂN HƯỚNG THÔNG QUA XỬ LÝ ẢNH
SV2022-150**

Thuộc nhóm ngành khoa học: Công nghệ kỹ thuật ô tô

SV thực hiện: Mai Đức Tùng Nam, Nữ: Nam

Dân tộc: Kinh

Lớp, khoa: 19145CLA4

Năm thứ: 4 / Số năm đào tạo: 8

Ngành học: Công nghệ kỹ thuật ô tô

Người hướng dẫn: Th.S Nguyễn Thành Tuyên

TP Hồ Chí Minh, 11/2022

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC PHẠM VĂN THẠCH TP HCM

THÔNG TIN KẾT QUẢ NGHIÊN CỨU CỦA ĐỀ TÀI

1. Thông tin chung:

- Tên đề tài: Nghiên cứu thiết kế thi công module gương camera cảnh báo tiền va chạm khi xe chuyển hướng thông qua xử lý ảnh.

- Chủ nhiệm đề tài: Mai Đức Tùng

Mã số SV: 19145187

- Lớp: 19145CLA4

Khoa: Đào tạo chất lượng cao

- Thành viên đề tài:

Stt	Họ và tên	MSSV	Lớp	Khoa
1	Nguyễn Thành Luân	19145152	19145CLA3	Đào tạo chất lượng cao
2	Ngô Vũ Trường	19145001	19145CLA4	Đào tạo chất lượng cao

- Người hướng dẫn: Th.S Nguyễn Thành Tuyên

2. Mục tiêu đề tài: Nghiên cứu xây dựng thuật toán dựa trên công cụ chính là python để nhận diện phương tiện cơ giới và xác định tốc độ của chúng và cảnh báo đến người dùng nguy cơ bị phương tiện khác vượt.

3. Tính mới và sáng tạo:

Trên thị trường có rất nhiều hệ thống cảnh báo va chạm nhưng hầu hết đều sử dụng hệ thống radar và cảm biến dựa trên hiện tượng sóng, ví dụ như sóng âm, dội lại khi gặp vật cản. Do đó tại em mong muốn ứng dụng, thử nghiệm xử lý ảnh để giải quyết vấn đề trên.

4. Kết quả nghiên cứu:

Sau khi chạy thử chương trình hệ thống đã có thể nhận diện chiếc xe, phỏng đoán tốc độ lưu thông và phát ra tín hiệu cảnh báo tài xế

5. Đóng góp về mặt giáo dục và đào tạo, kinh tế - xã hội, an ninh, quốc phòng và khả năng áp dụng của đề tài:

Bước đầu dự án đã hoàn thiện quá trình nhận diện và phát ra cảnh báo. Có thể phát triển lên để hoàn thiện hơn

6. Công bố khoa học của SV từ kết quả nghiên cứu của đề tài (ghi rõ tên tạp chí nếu có) hoặc nhận xét, đánh giá của cơ sở đã áp dụng các kết quả nghiên cứu (nếu có):

Ngày tháng năm
SV chịu trách nhiệm chính
thực hiện đề tài
(kí, họ và tên)

Nhận xét của người hướng dẫn về những đóng góp khoa học của SV thực hiện đề tài *(phần này do người hướng dẫn ghi):*

Ngày tháng năm

Người hướng dẫn

(kí, họ và tên)

MỤC LỤC

MỤC LỤC.....	1
1. Đặt vấn đề	5
2. Tổng quan tài liệu.....	5
2.1. Mục tiêu nghiên cứu.....	5
2.2. Nghiên cứu trong và ngoài nước	5
2.2.1. Nghiên cứu trong nước	5
2.2.2. Nghiên cứu ngoài nước	6
2.3. Giới hạn đề tài và phạm vi ứng dụng	7
3. Kết quả nghiên cứu	8
3.1. Kết quả nghiên cứu cơ sở lý thuyết về ảnh và xử lý ảnh.....	8
3.1.1. Cơ sở lý thuyết về xử lý ảnh	8
3.1.2. Tổng quan về xử lý ảnh.....	21
3.1.3. Các quy trình xử lý ảnh	24
3.2. Kết quả nghiên cứu giải thuật nhận diện chiếc xe vượt ngưỡng tốc độ cho phép	28
3.3. Kết quả nghiên cứu thiết kế và lắp đặt phần cứng.....	34
3.3.1. Nghiên cứu thiết kế.....	34
3.3.2. Nghiên cứu lắp đặt phần cứng	36
3.4. Vận hành module về khả năng xử lý và cảnh báo bằng mô hình thực nghiệm	36
3.5. Kết quả nghiên cứu sơ đồ giải thuật	38
4. Kết luận và ý nghĩa của đề tài	42
4.1. Ý nghĩa về mặt khoa học.....	42
4.2. Đóng góp về phương diện xã hội & kinh tế.....	42
4.3. Định hướng phát triển trong tương lai.....	42
Tài liệu tham khảo	44
Phụ lục.....	46
Phụ lục 1: Chương trình python.....	46
import cv2	46
Phụ lục 2: Sơ lược về hệ điều hành, cách thiết lập và các thư viện.....	48
Phụ lục 3: Sơ lược về thiết bị.....	56
Phụ lục 4: Nguồn lấy Haar Cascade	60

MỤC LỤC HÌNH ẢNH

Hình 3.1. Cấu tạo nhiều điểm ảnh thành 1 bức ảnh hoàn chỉnh.....	8
Hình 3.2: Hai hệ màu RGB và CMYK	8
Hình 3.3: Một Pixel trên module màn hình LED.....	9
Hình 3.4: Khoảng cách giữa các màu	10
Hình 3.5: Độ phân giải ảnh ảnh hưởng đến chất lượng hiển thị ảnh	11
Hình 3.6: Bảng kích thước độ phân giải ảnh.....	12
Hình 3.7: Cách tính độ phân giải	13
Hình 3.8: Tầm hình màu khi chuyển thành ảnh xám	13
Hình 3.9: Ảnh nhị phân là dãy các số 0,1	14
Hình 3.10: Ảnh màu sau khi chuyển sang ảnh nhị phân.....	15
Hình 3.11: Hệ tọa độ RGB.....	16
Hình 3.12: Các pixel được phóng lớn, có các màu red-green-blue được lặp lại, cứ 3 màu được 1 pixel.....	16
Hình 3.13: Ảnh được tách ra 3 kênh màu	17
Hình 3.14: Không gian màu HSV	18
Hình 3.15: Không gian màu CMYK.....	19
Hình 3.16: Ảnh xám và ảnh nhị phân	20
Hình 3.17: Ảnh màu.....	21
Hình 3.18: Thị giác máy tính	22
Hình 3.19: Các lớp trên cùng của mạng thần kinh phát hiện các đặc trưng chung; các lớp sâu hơn phát hiện các đối tượng thực tế.....	23
Hình 3.20: Cách thị giác máy tính hoạt động.....	23
Hình 3.21: Các giai đoạn chính trong xử lý ảnh	24
Hình 3.22: Pixel	26
Hình 3.23: Mô hình vector	27
Hình 3.24: Mô hình Raster.....	28

Hình 3.25: Quy trình để nhận diện một khuôn mặt bằng phương pháp Haar Cascade	29
Hình 3.26: Dựa vào các góc cạnh để xác định hình chữ nhật	29
Hình 3.27: Cách thức hoạt động khi nhận diện mặt người	30
Hình 3.28: Kết quả nhận diện khuôn mặt với các ô chữ nhật khác nhau có nhiệm vụ nhận diện đối tượng mắt, mũi và miệng.....	30
Hình 3.29: Nhận diện phương pháp YOLOv5	31
Hình 3.30: Câu lệnh huấn luyện mẫu.....	32
Hình 3.31: Kết quả sau khi huấn luyện	33
Hình 3.32: Kết quả ở dạng biểu đồ	33
Hình 3.33: Bản thiết kế mô hình xe tải	34
Hình 3.34: Bản thiết kế mô hình xe ô tô con.....	35
Hình 3.35: Bản vẽ mô hình hóa hệ thống.....	35
Hình 3.36: Sơ đồ lắp ráp các máy tính nhúng và linh kiện	36
Hình 3.37: Mô hình thực tế đang vận hành.....	37
Hình 3.38: Sơ đồ giải thuật của việc nhận dạng xe	39

PHỤ LỤC BẢNG

Bảng 2.1: Mục tiêu và phương pháp nghiên cứu đề tài	5
Bảng 3.1: Bảng phân bố góc với từng màu	17

1. Đặt vấn đề

Trong thời đại công nghiệp ô tô ngày nay không còn mang khuynh hướng thuần cơ khí như trước đây nữa, mà nó đang dần chuyển dần qua xu thế kết hợp các lĩnh vực khác như điện, điện tử, nhúng, khoa học máy tính, AI. Việc đảm bảo an toàn khi di chuyển trên đường phố của các phương tiện giao thông ngày càng cải thiện để giảm thiểu thiệt hại. Hiện nay các vụ va chạm khi vào cua giữa xe máy và ô tô là rất phổ biến, dù ô tô đã xinhan (Vì thực tế với mắt thường qua kính hậu thì khó có thể xác định xe bên hông đang chạy với tốc độ đều hay đang vượt). Theo đà phát triển của gương hậu camera đề ra giải quyết vấn đề quay cổ quá nhiều để quan sát gương hậu. Trên thị trường có rất nhiều hệ thống cảnh báo va chạm nhưng hầu hết đều sử dụng hệ thống radar và cảm biến dựa trên hiện tượng sóng, ví dụ như sóng âm, dội lại khi gặp vật cản. Do đó tụi em mong muốn ứng dụng, thử nghiệm xử lý ảnh để giải quyết vấn đề trên.

2. Tổng quan tài liệu

2.1. Mục tiêu nghiên cứu

Mục tiêu tổng quát: Chế tạo module cảnh báo tiền va chạm thông qua xử lý ảnh

Mục tiêu cụ thể	Phương pháp nghiên cứu
1. Nghiên cứu cơ sở lý thuyết của xử lý ảnh	Phương pháp nghiên cứu lý thuyết
2. Nghiên cứu giải thuật nhận diện chiếc xe vượt ngưỡng tốc độ cho phép	Phương pháp nghiên cứu lý thuyết
3. Nghiên cứu các thư viện xử lý ảnh của Opencv trong ngôn ngữ lập trình Python	Phương pháp nghiên cứu lý thuyết
4. Thiết kế và lắp đặt các phần cứng	Phương pháp thực nghiệm
5. Vận hành module về khả năng xử lý và cảnh báo bằng những Video cao tốc và các mô hình xe cộ.	Phương pháp thực nghiệm

Bảng 2.1: Mục tiêu và phương pháp nghiên cứu đề tài

2.2. Nghiên cứu trong và ngoài nước

2.2.1. Nghiên cứu trong nước

Điểm mù của các phương tiện có kích thước lớn [1]

Xã hội càng phát triển thì nhu cầu của con người trong vận chuyển hàng hóa cũng trở nên nhiều hơn, việc các xe tải lớn, siêu trường, siêu trọng hay các container xuất hiện trên các tuyến đường cũng ngày một nhiều hơn. Song song với lợi ích kinh tế mà các loại xe có kích thước lớn này mang lại thì các nguy cơ tai nạn tiềm ẩn trong đó cũng ngày một lớn hơn. Điểm mù của các phương tiện là một trong những nguyên nhân gây ra những vụ tai nạn thảm khốc gần đây. Ở các xe có kích thước nhỏ thì điểm mù sẽ nhỏ và gần như không đáng kể so với kích thước con người cũng như các phương tiện khác. Tuy nhiên ở các phương tiện có tải trọng lớn và kích thước siêu khổng lồ thì điểm mù là một vùng rất rộng và tiềm ẩn nguy cơ gây ra các vụ tai nạn là rất cao bởi vì điểm mù tỉ lệ thuận với kích thước của xe. Đặc biệt là những trường hợp xe đổi hướng di chuyển, vào cua, quay đầu thì xác suất xảy ra tai nạn lại càng cao.

Nghiên cứu, thiết kế, chế tạo hệ thống xác định khoảng cách giữa ô tô và chướng ngại vật [2]

Từ nhu cầu an toàn của con người trên ô tô cũng như các phương tiện khác khi tham gia giao thông, ngoài các hệ thống an toàn đã có trên ô tô như hệ thống túi khí (air bag) hay hệ thống ABS... Bài báo cáo này nghiên cứu hệ thống mới cảnh báo đến người dùng thông qua việc xuất ra tín hiệu khi có vật cản trên đường. Bài báo cáo xoay sâu vào việc ứng dụng phương pháp phát và thu nhận tín hiệu thông qua cảm biến siêu âm (ultrasonic sensor) để phát ra tín hiệu cảnh báo đến người dùng. Nghiên cứu này xây dựng mô hình về hệ thống cảnh báo va chạm thông qua việc tính toán các thông số trong cảm biến chống va chạm. Từ đó, lên ý tưởng thiết kế và chế tạo một hệ thống hoàn thiện có thể xác định khoảng cách cụ thể giữa ô tô tới chướng ngại vật và điều khiển cảm biến để có thể phát hiện được chướng ngại vật ở khoảng cách xa hơn. Quy trình thực hiện hệ thống thông qua các giai đoạn nghiên cứu về phương pháp phát và thu nhận tín hiệu của cảm biến siêu âm, sau đó nghiên cứu tính toán các thông số từ xe đến chướng ngại vật để đề xuất hướng lập trình cho hệ thống cảnh báo và đưa vào phần cứng để có thể thực nghiệm và kiểm tra cũng như mô phỏng phiên bản hoàn chỉnh của hệ thống.

2.2.2. Nghiên cứu ngoài nước

A Blind-Zone Detection Method Using a Rear-Mounted Fisheye Camera with Combination of Vehicle Detection Methods [3]

Bài báo này đề xuất một phương pháp mới để phát hiện và theo dõi các phương tiện phía sau và trong vùng mù của phương tiện, sử dụng một camera mắt cá gắn phía sau và nhiều thuật toán phát hiện. Một thao tác là nguyên nhân chính gây ra tai nạn liên quan đến việc xe mục tiêu tiếp cận xe chủ từ phía sau và vượt vào làn đường liền kề. Khi phương tiện bị vượt di chuyển về phía rìa của hình ảnh và đi vào vùng mù, chế độ xem của phương tiện đó dần dần thay đổi từ chế độ xem trước sang chế độ xem bên. Hơn nữa, ảnh hưởng của biến dạng mắt cá thể hiện rõ nhất ở phần rìa của hình

ảnh, khiến việc phát hiện phương tiện mục tiêu đi vào vùng mù thậm chí còn khó khăn hơn. Hệ thống được đề xuất sử dụng bộ phân loại AdaBoost ở khoảng cách 10–40 m giữa máy chủ và phương tiện mục tiêu. Để phát hiện ở khoảng cách ngắn khi chế độ xem của phương tiện mục tiêu đã thay đổi thành chế độ xem bên và bộ phân loại AdaBoost kém hiệu quả hơn, nên đề xuất nhận dạng bánh xe. Hai phương pháp phát hiện bánh xe được sử dụng: ở khoảng cách từ 5 đến 15 m, một thuật toán mới có tên phát hiện đường viền vòm bánh xe được trình bày và đối với khoảng cách dưới 5 m, phát hiện vòng tròn Hough cung cấp khả năng phát hiện bánh xe đáng tin cậy. Một khung thử nghiệm cũng được trình bày, phân loại hiệu suất phát hiện theo chức năng của khoảng cách giữa máy chủ và phương tiện mục tiêu. Kết quả thử nghiệm chỉ ra rằng phương pháp được đề xuất mang lại tỷ lệ phát hiện lớn hơn 93% trong phạm vi tới hạn (vùng mù) của vật chủ.

Honda clarity plug-in hubrid 2021 With a Blind Spot Camera [4]

Cụ thể trên 2021 HONDA CLARITY PLUG-IN HYBRID. Honda Clarity plug-in hybrid sedan có tính năng Honda LaneWatch tiêu chuẩn. Camera này chỉ kích hoạt khi sử dụng tín hiệu rẽ phải và hiển thị nguồn cấp dữ liệu trực tiếp trên màn hình thông tin giải trí. LaneWatch đang dần bị loại bỏ, vì vậy nó có số lượng hạn chế trong dòng sản phẩm của Honda. Một điểm khác biệt đáng chú ý giữa hệ thống đang sử dụng tại Hyundai/Kia/Genesis là Lane Watch hiển thị nguồn cấp dữ liệu video trên màn hình thông tin giải trí thay vì cụm đồng hồ, yêu cầu người lái xe phải chuyển sự chú ý của họ ra khỏi đường nhiều hơn so với các hệ thống khác.

2.3. Giới hạn đề tài và phạm vi ứng dụng

Giới hạn nghiên cứu của đề tài chỉ dừng lại ở việc đưa ra tín hiệu cảnh báo đến người dùng (phát ra âm thanh báo hiệu).

Về tài chính: vấn đề hiện tại phần cứng xử lý chưa đủ mạnh để đáp ứng các bước xử lý ảnh nhưng với những phần cứng tốt hơn giá thành đội lên rất cao. Chính vì thế, đề tài bị hạn chế về tính tức thời, khả năng xử lý tình huống hay khả năng khắc phục sự cố khi sắp xảy ra va chạm.

Module camera có thể ứng dụng trên bất kì xe ô tô nào và đạt được hiệu quả cao ở các xe có kích thước lớn.

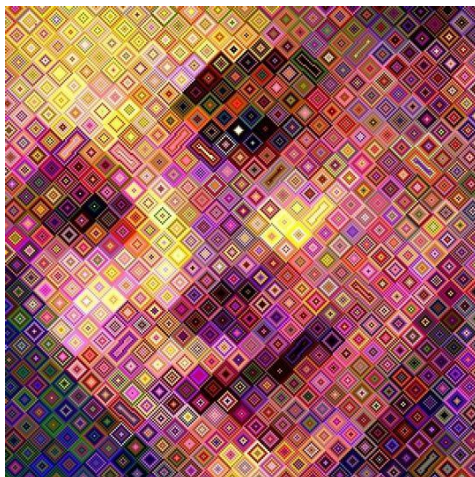
Hiện tại vấn đề xử lý cảnh báo thực sự hiệu quả khi sử dụng trong điều kiện đầy đủ ánh sáng, còn trong các điều kiện thiếu sáng hay thời tiết xấu vẫn còn nhiều hạn chế.

3. Kết quả nghiên cứu

3.1. Kết quả nghiên cứu cơ sở lý thuyết về ảnh và xử lý ảnh

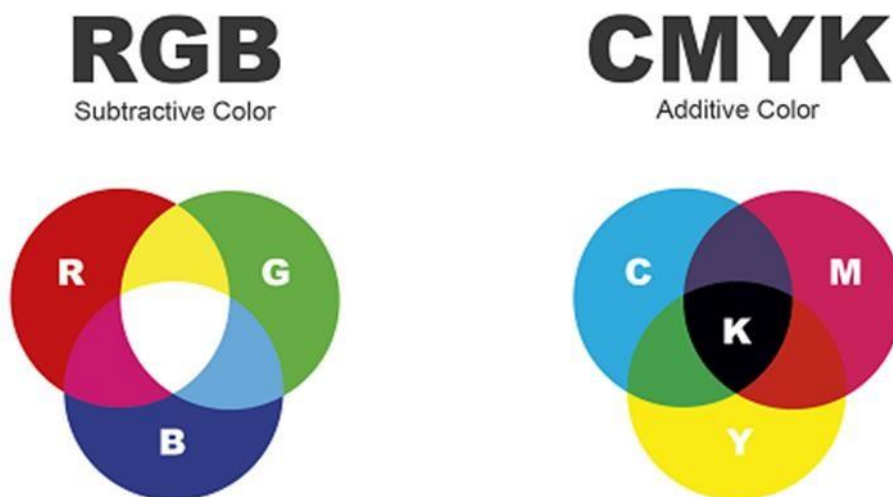
3.1.1. Cơ sở lý thuyết về xử lý ảnh

- Điểm ảnh: Điểm ảnh (hay còn gọi là Pixel – viết tắt của picture element) là một điểm màu rất nhỏ cấu tạo nên một bức ảnh kỹ thuật số. Mỗi Pixel đều có một tọa độ địa lý ITS riêng biệt. Mỗi Pixel tương ứng với một mảnh nhỏ của ảnh, bức ảnh càng nhiều Pixel thì càng rõ nét và hiển thị đầy đủ các chi tiết so với bản gốc. [5]



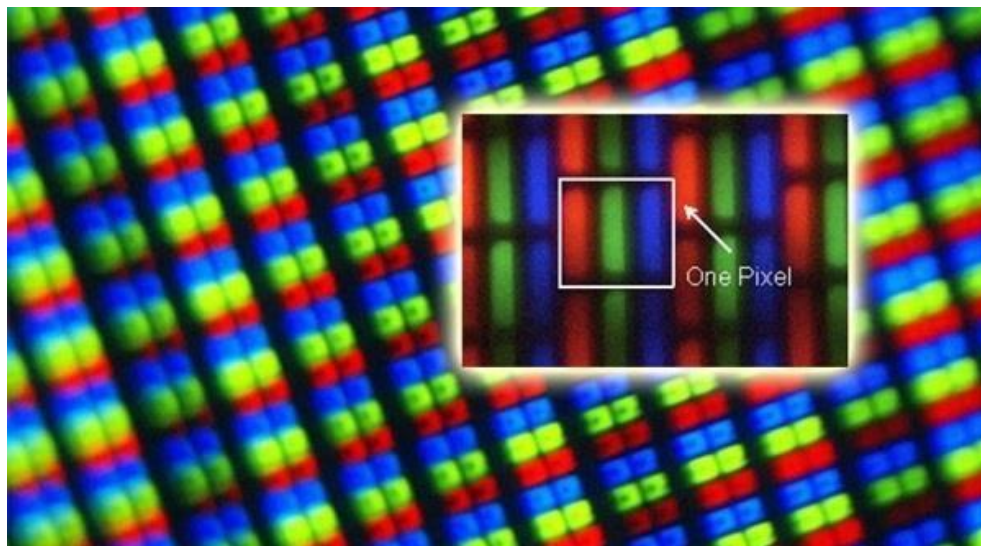
Hình 3.1: Cấu tạo nhiều điểm ảnh thành 1 bức ảnh hoàn chỉnh

Cường độ của mỗi điểm ảnh có thể thay đổi tùy thuộc vào các hệ màu sắc đại diện trong đo lường độ thành phần như RGB (Red – đỏ, Green – xanh lá, Blue – xanh dương) hay CMYK (Cyan – xanh lơ, Magenta – Hồng cánh sen, Yellow – vàng, Key – đen).



Hình 3.2: Hai hệ màu RGB và CMYK

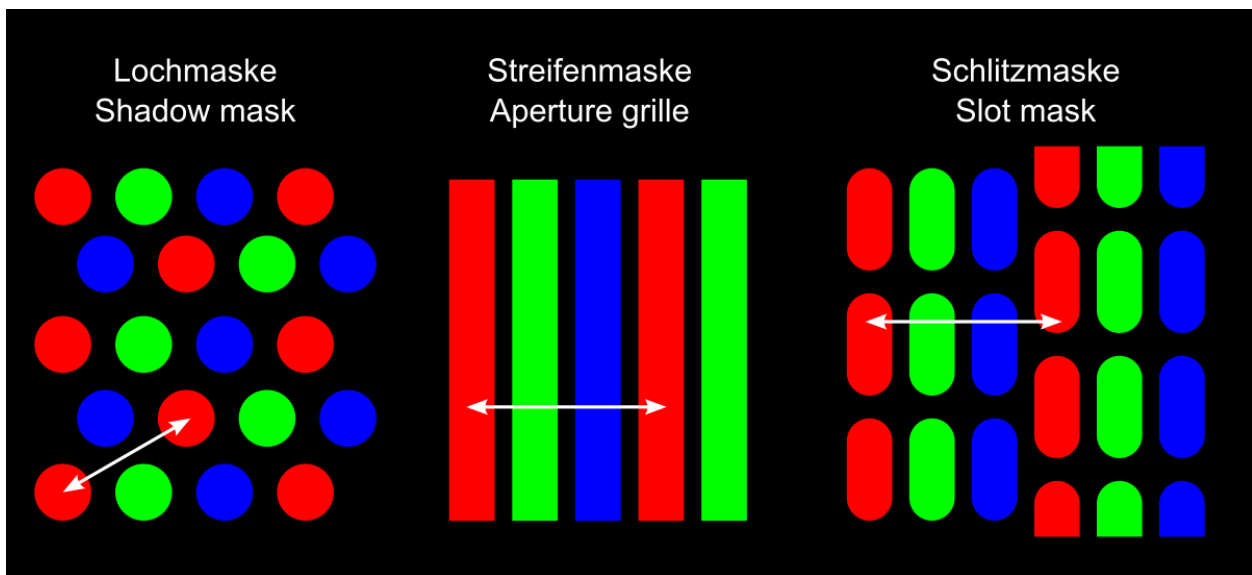
Độ cao của một Pixel được tính bằng milimet, chính là khoảng cách từ tâm điểm ảnh này đến tâm của điểm ảnh tiếp theo. Một bức ảnh chứa càng nhiều Pixel thì sẽ có dung lượng càng lớn.



Hình 3.3: Một Pixel trên module màn hình LED

Pixel Pitch, hay khoảng cách điểm ảnh ảnh hưởng rất lớn đến việc lựa chọn màn hình LED. Ví dụ, với một màn hình có kích thước 100inch, khoảng cách điểm ảnh lớn, thì các điểm ảnh Pixel lại ít đi, hình ảnh khi nhìn gần sẽ không nét như khi nhìn xa. [6]

Dot pitch (đôi khi được gọi là line pitch, stripe pitch hoặc phosphor pitch) là một đặc điểm kỹ thuật cho màn hình máy tính, máy in máy tính, máy quét hình ảnh hoặc các thiết bị dựa trên pixel khác mô tả khoảng cách, ví dụ, giữa các dấu chấm (pixel phụ) trên màn hình hiển thị. Trong trường hợp hiển thị màu RGB, đơn vị dẫn xuất của cao độ pixel là thước đo kích thước của bộ ba cộng với khoảng cách giữa các bộ ba. [7]



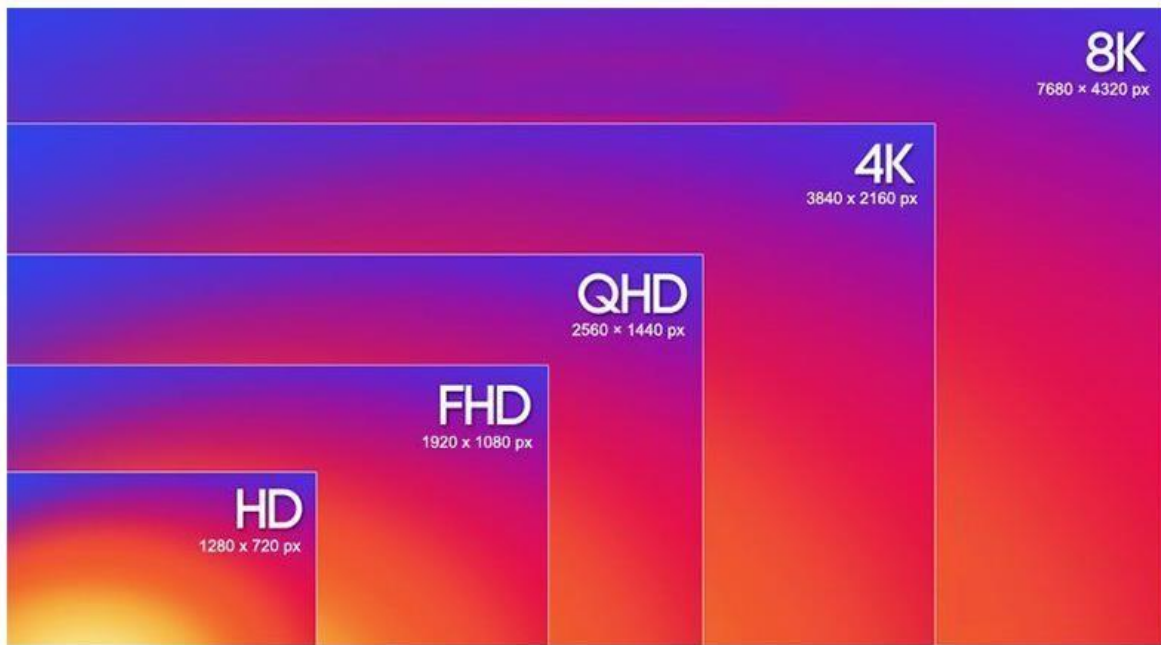
Hình 3.4: Khoảng cách giữa các màu

Cao độ chấm có thể được đo bằng đơn vị tuyến tính (với số nhỏ hơn có nghĩa là độ phân giải cao hơn), thường là milimét (mm), hoặc theo tốc độ, ví dụ, các chấm trên inch (với số lớn hơn có nghĩa là độ phân giải cao hơn). Khoảng cách gần hơn tạo ra hình ảnh sắc nét hơn (vì có nhiều chấm hơn trong một khu vực nhất định). Tuy nhiên, các yếu tố khác có thể ảnh hưởng đến chất lượng hình ảnh, bao gồm:

Phương pháp đo lường không có giấy tờ hoặc không được ghi chép đầy đủ, phức tạp do thiếu hiểu biết về sự tồn tại của các phương pháp khác nhau. Sự nhầm lẫn của pixel và subpixels. Khoảng cách phần tử khác nhau trên khu vực màn hình (ví dụ: mở rộng ở các góc so với trung tâm), hình dạng pixel khác nhau, tỷ lệ khung hình hình ảnh và pixel khác nhau, các yếu tố khác như yếu tố Kell hoặc video xen kẽ

Sự khác biệt chính xác giữa độ chấm ngang và chéo khác nhau tùy theo thiết kế của màn hình (xem hình học pixel và màn hình rộng), nhưng màn hình 0,28 mm (đường chéo) cấp nhập cảnh điển hình có khoảng cách ngang 0,24 hoặc 0,25 mm và đơn vị 0,26 mm (đường chéo) chất lượng tốt có khoảng cách ngang là 0,22 mm.

- Độ phân giải ảnh: Độ phân giải hình ảnh (Image resolution) cho biết lượng thông tin chứa đựng trong 1 tập tin ảnh hiển thị trên màn hình, chính là số lượng điểm ảnh chứa trên 1 màn hình hiển thị. [8]



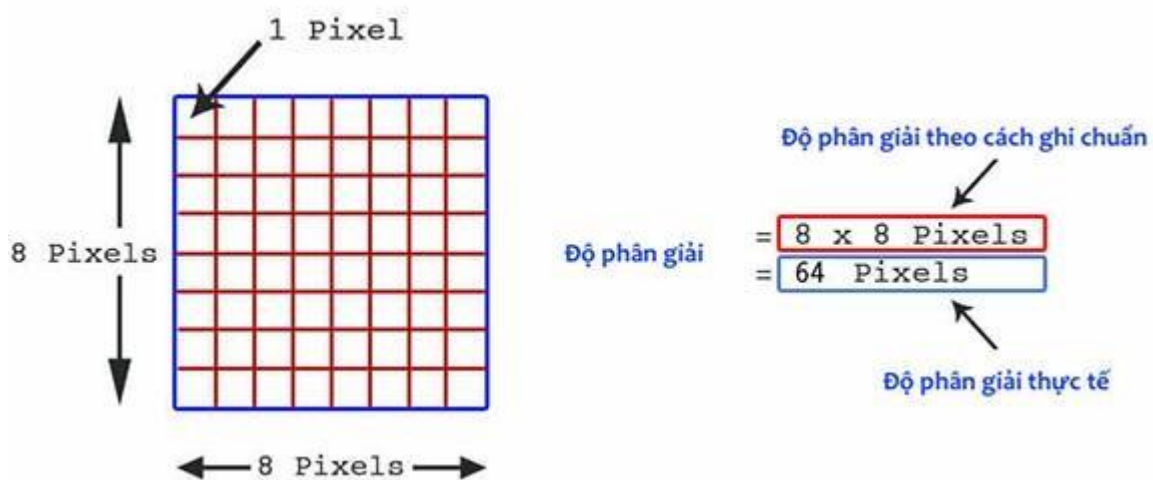
Hình 3.5: Độ phân giải ảnh ảnh hưởng đến chất lượng hiển thị ảnh

Đơn vị đo độ phân giải của hình ảnh thường là Pixel hoặc Megapixel (1 Megapixel = 1 triệu Pixel). Mỗi một Pixel có kích thước tương đối bằng $0,26 \times 0,35$. Pixel được hiển thị trên màn hình là PPI (pixels per inch).

<div>TUYỆT VỜI</div> <div>XUẤT SẮC</div> <div>RẤT TỐT</div> <div>TỐT</div> <div>TỆ</div>										
* Chỉ số trong khung là PPI (Pixel per inch), PPI càng cao thì hình ảnh càng chi tiết										
KÍCH THƯỚC IN ẤN										
ĐỘ PHÂN GIẢI MÃY ẢNH		8x12	11x14	16x20	16x24	20x30	24x36	30x45	40x60	50x75
	2MP	133	114	80	66	60	53	35	26	21
	3MP	193	165	115	96	77	64	51	38	30
	4MP	205	176	123	102	82	68	54	41	32
	5MP	216	185	129	121	86	72	57	43	34
	6MP	250	214	150	125	100	83	66	50	40
	7MP	256	219	153	128	102	85	68	51	40
	8MP	259	222	155	129	103	86	69	51	41
	9MP	290	249	174	145	116	96	77	58	46
	10MP	322	276	193	161	129	107	86	64	51
	11MP	338	290	203	169	135	112	90	67	54
	12.7MP	364	312	218	182	145	121	97	72	58
	16.6MP	416	356	249	208	166	138	110	83	66
	18MP	408	350	245	204	163	136	108	81	65
	21.1MP	468	401	280	234	187	156	124	93	74
	22MP	457	392	274	228	182	152	121	91	73
	31MP	541	464	324	270	216	180	133	108	86
	39MP	601	515	360	300	240	200	160	120	96
	63MP	812	696	487	406	324	270	216	162	129
	35mm*	590	515	354	295	236	196	157	118	94
	6x6cm	944	809	566	472	377	314	251	188	151
	6x7cm	956	820	574	478	382	318	255	191	153
	4x5	988	847	592	494	395	329	263	197	158
	8x10	1383	1186	830	691	553	461	368	276	221

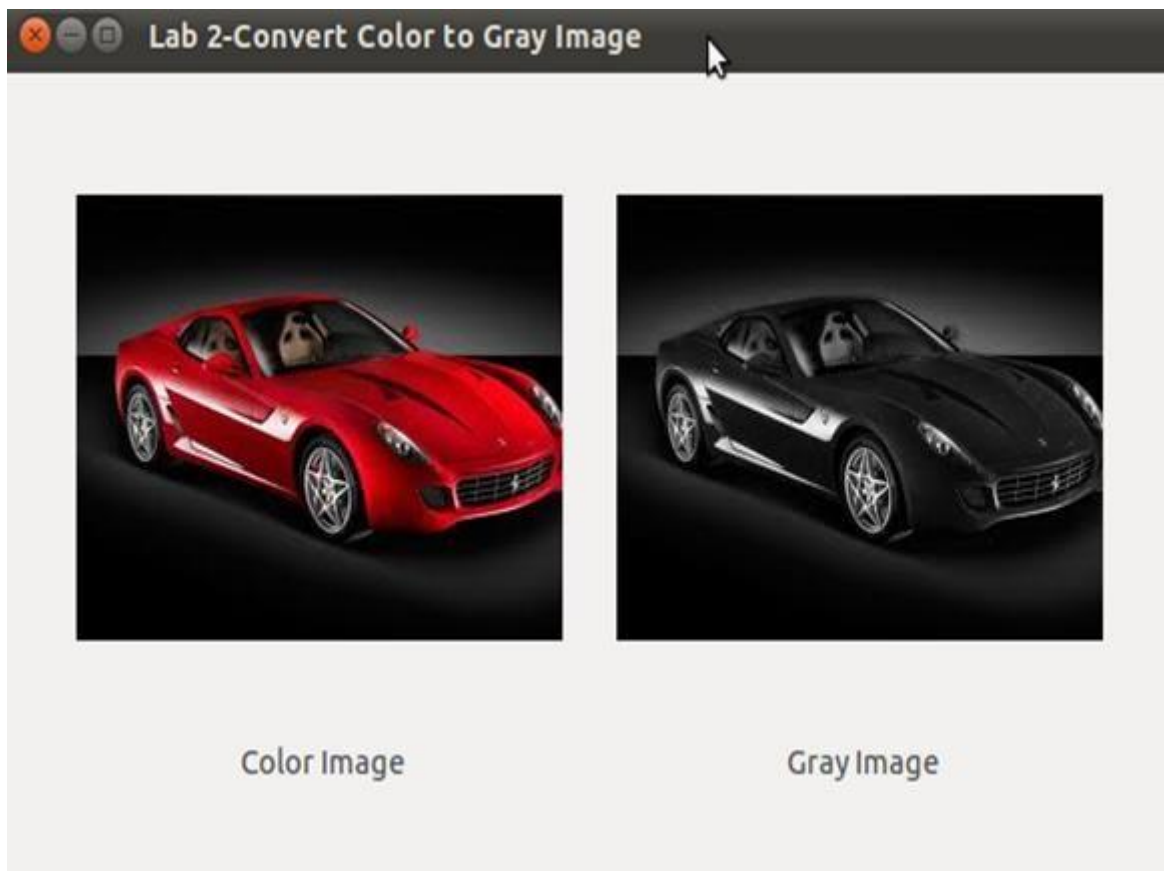
Hình 3.6: Bảng kích thước độ phân giải ảnh

Tổng số Pixel trên một bức ảnh sẽ được gọi là độ phân giải của bức ảnh. Cách tính độ phân giải của ảnh đó là nhân số cột điểm ảnh với hàng điểm ảnh rồi chia cho một triệu. Ví dụ hình ảnh có kích thước 1920x1080 thì sẽ có tổng cộng 2.073.600 Pixel tương ứng với xấp xỉ 2 Megapixel.



Hình 3.7: Cách tính độ phân giải

- Mức xám của ảnh: Ảnh mức xám (hay còn gọi là ảnh đơn sắc – monochromatic). Mỗi điểm ảnh trong ảnh mức xám sẽ có giá trị từ 0 cho đến 255. Giá trị của Pixel bằng 0 tương ứng với điểm ảnh tối (đen), giá trị của Pixel bằng 255 tương ứng với điểm ảnh sáng (trắng).



Hình 3.8: Tầm hình màu khi chuyển thành ảnh xám

Độ sáng của điểm ảnh được tính theo công thức [9]:

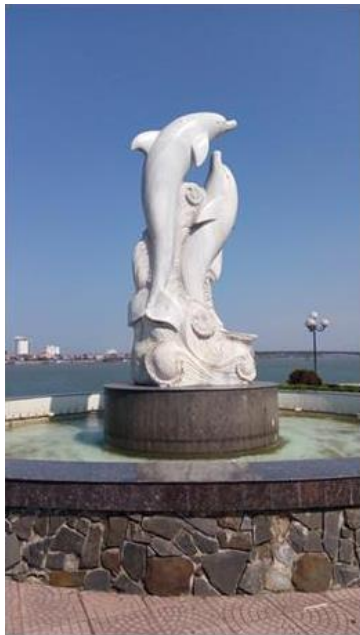
$$S = 0.2989R + 0.5870G + 0.1140B$$

Ảnh nhị phân là ảnh số, trong đó mỗi điểm ảnh được biểu diễn bởi giá trị là 0 (trắng) hoặc 1 (đen).



Hình 3.9: Ảnh nhị phân là dãy các số 0,1

Ảnh nhị phân được tạo ra bằng cách biến đổi ảnh xám dựa vào một ngưỡng xác định. Cách phổ biến nhất để tạo ra một ảnh nhị phân từ một ảnh xám là chọn một ngưỡng sao cho tất cả các điểm ảnh ít quan tâm có giá trị thấp hơn ngưỡng sẽ có màu trắng và các điểm ảnh được quan tâm có giá trị cao hơn ngưỡng sẽ có màu đen hoặc ngược lại.



Color



Grayscale



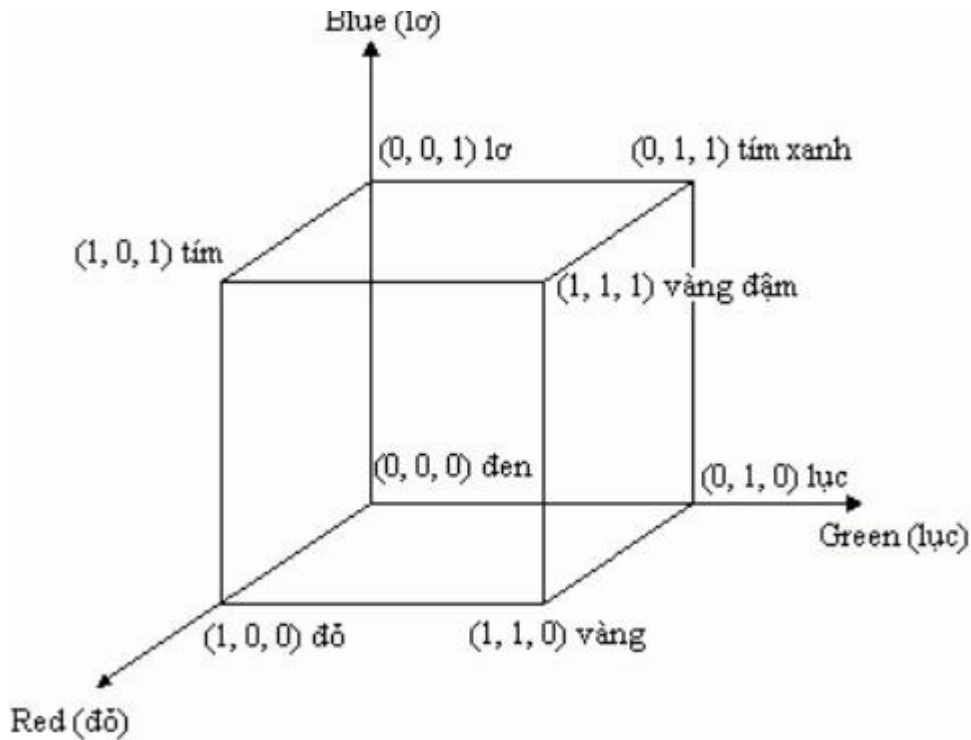
Black-and-white

Hình 3.10: Ảnh màu sau khi chuyển sang ảnh nhị phân

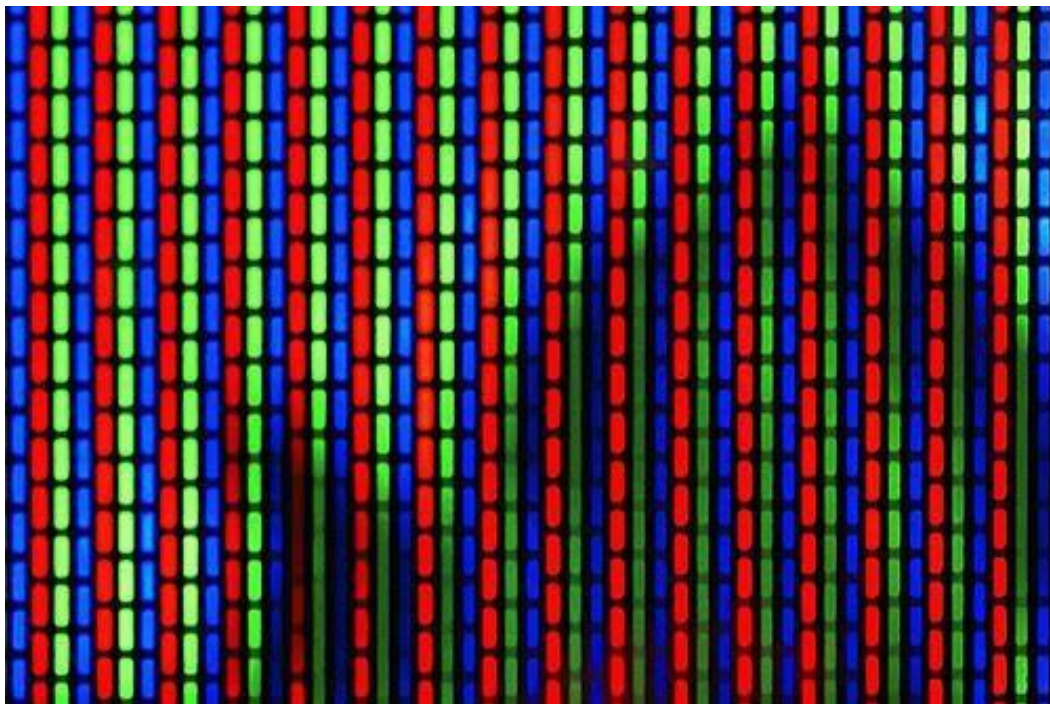
- Không gian màu [10]:

Có 3 không gian màu chính: Không gian màu RGB, không gian màu HSV, không gian màu CMYK.

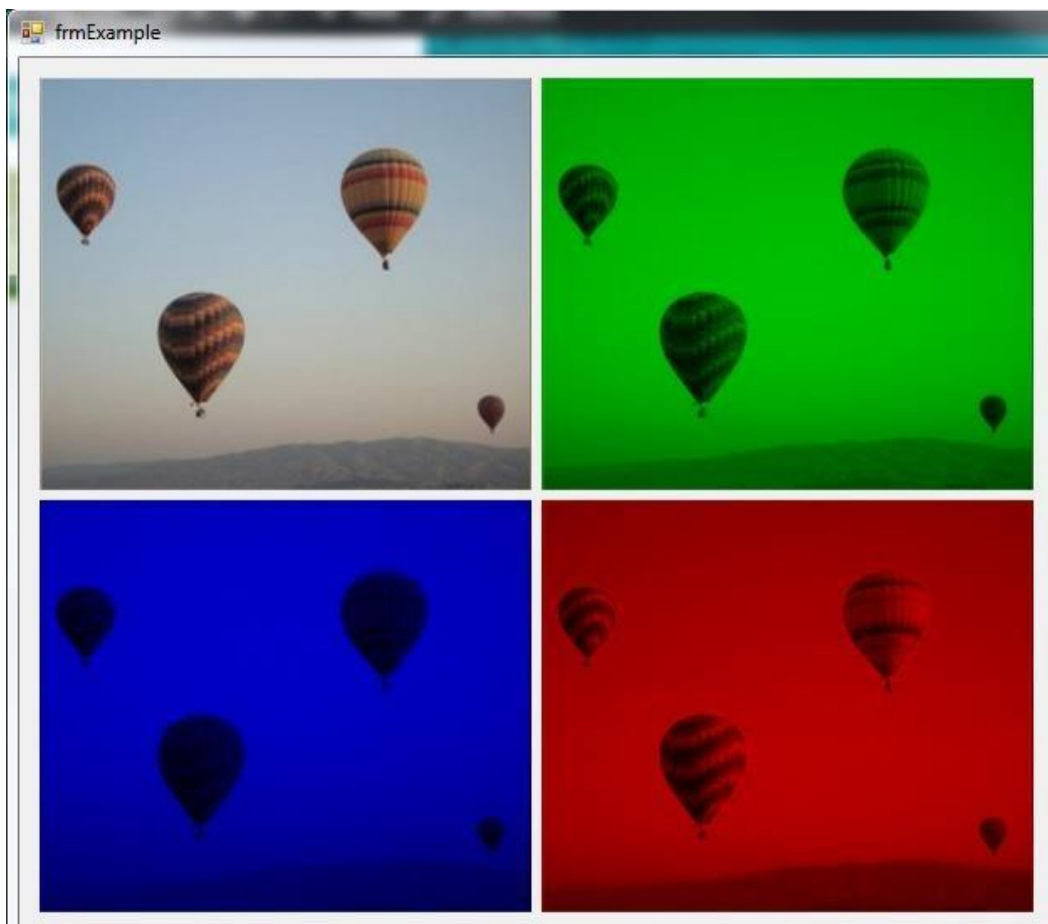
+ Không gian màu RGB: dùng 3 màu cơ bản R, G, B và ký hiệu RGBCIE để phân biệt với các chuẩn khác. Người ta dùng hệ tọa độ màu R-G-B (tương ứng với hệ tọa độ x-y-z) để biểu diễn màu như sau.



Hình 3.11: Hệ tọa độ RGB



Hình 3.12: Các pixel được phóng lớn, có các màu red-green-blue được lặp lại, cứ 3 màu được 1 pixel



Hình 3.13: Ảnh được tách ra 3 kênh màu

+ Không gian màu HSV: được ứng dụng nhiều trong việc chỉnh sửa ảnh, phân tích ảnh và thị giác máy tính. Hệ không gian này dựa vào ba thông số sau để mô tả màu sắc:

H(Hue): Vùng màu, là phần màu của mô hình màu và được biểu thị dưới dạng một số từ 0 đến 360 độ.

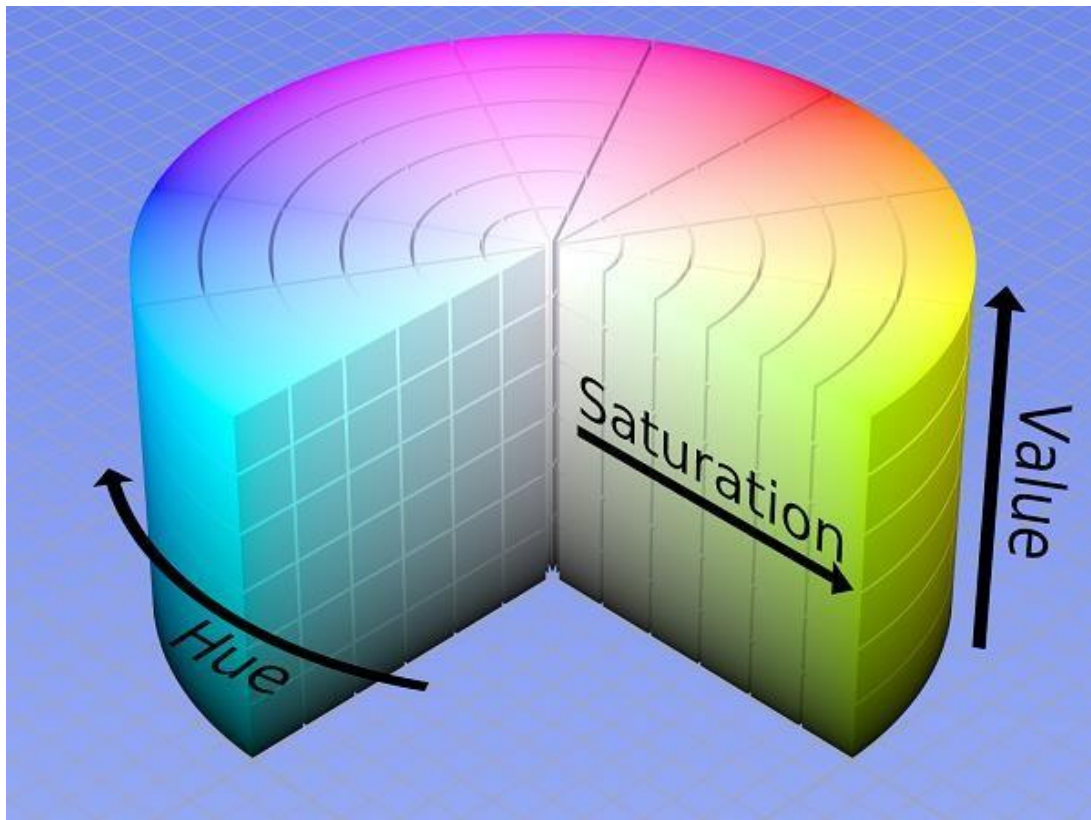
Màu	Góc
Màu đỏ	0-60
Màu vàng	60-120
Màu xanh lá	120-180
Cyan	180-240
Màu xanh da trời	240-300
Màu đỏ tươi	300-360

Bảng 3.1: Bảng phân bố góc với từng màu

S (Saturation): Độ bão hòa màu là lượng màu xám trong màu, từ 0 đến 100 phần trăm. Một hiệu ứng mờ nhạt có thể có được từ việc giảm độ bão hòa về không để giới thiệu nhiều màu xám hơn.

Tuy nhiên, độ bão hòa đôi khi được xem trên phạm vi từ 0-1, trong đó 0 là màu xám và 1 là màu chính.

V (Bright hay Value): độ sáng



Hình 3.14: Không gian màu HSV

+ Không gian màu CMYK: được ứng dụng phổ biến trong ngành in ấn thiết kế như poster, brochure, name card, catalogue, sách hoặc tạp chí, ...

- C(Cyan): xanh lơ
- M(Magenta): hồng x
- Y(Yellow): Vàng



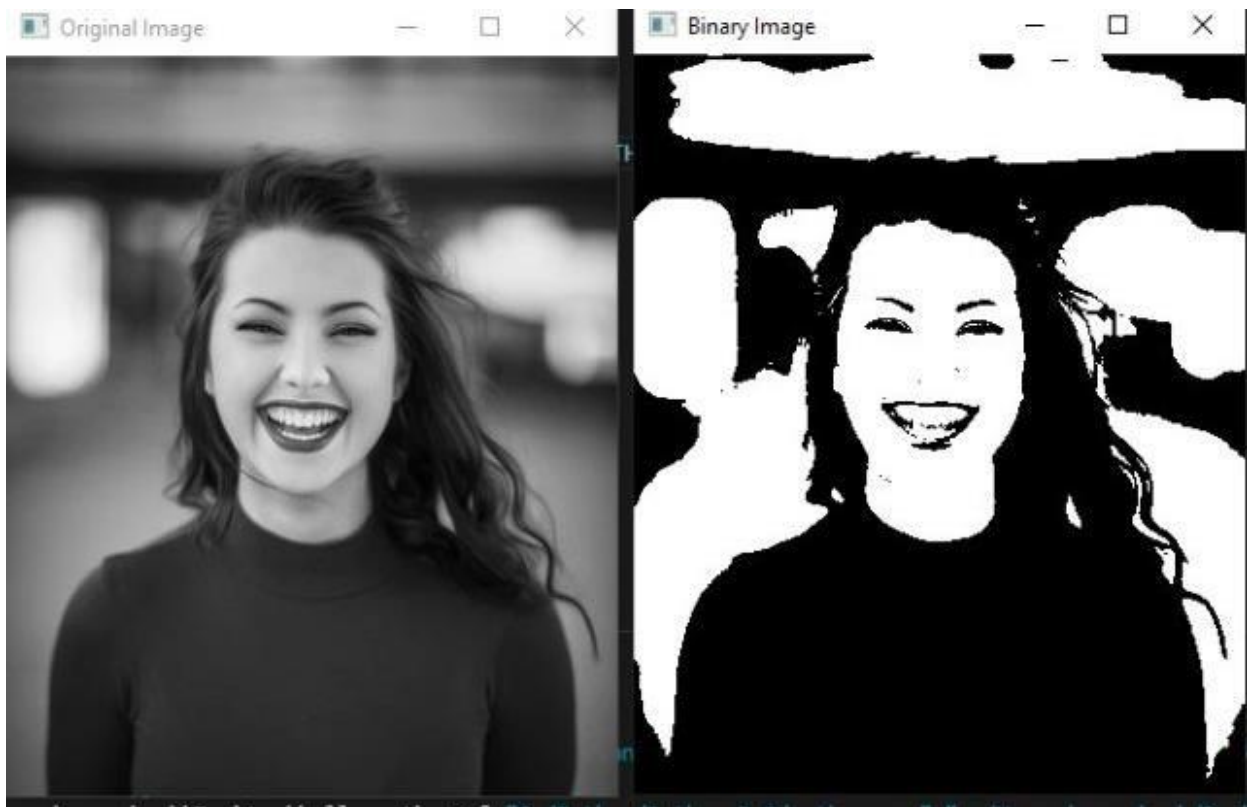
Hình 3.15: Không gian màu CMYK

- Định nghĩa ảnh số: Ảnh số là khái niệm dùng để biểu diễn số học của hình ảnh máy tính với mức xám phù hợp dùng để mô tả ảnh gần với ảnh thật
- Ảnh trắng đen: Ảnh đen trắng chỉ bao gồm 2 màu :màu đen và màu trắng .Người ta phân mức đen trắng đó thành L mức .Nếu dùng số bit $B = 8$ bit để mã hóa mức đen trắng ta có công thức :

$$L = 2^B \quad (L = 2^8 = 256)$$

Nếu L bằng 2, $B = 1$, nghĩa là chỉ có 2 mức: mức 0 và mức 1, còn gọi là ảnh nhị phân (binary image). Mức 1 ứng với màu sáng, còn mức 0 ứng với màu tối. Nếu L lớn hơn 2 ta có ảnh đa cấp xám. Nói cách khác, với ảnh nhị phân mỗi điểm ảnh được mã hóa trên 1 bit, còn với ảnh 256 mức, mỗi điểm ảnh được mã hóa trên 8 bit. Như vậy, với ảnh đen trắng: nếu dùng 8-bit (1 byte) để biểu diễn mức xám, số các mức xám có thể biểu diễn được là 256. Mỗi mức xám được biểu diễn dưới dạng là một số nguyên nằm trong khoảng từ 0 đến 255, với mức 0 biểu diễn cho mức cường độ đen nhất và 255 biểu diễn cho mức cường độ sáng nhất.

Ảnh nhị phân khá đơn giản, các phần tử ảnh có thể coi như các phần tử logic. Ứng dụng chính của nó được dùng theo tính logic để phân biệt đối tượng ảnh với nền hay để phân biệt điểm biên với điểm khác.



Hình 3.16: Ảnh xám và ảnh nhị phân

- Ảnh màu:

Ảnh mẫu theo lý thuyết của Thomas là ảnh tổ hợp từ 3 màu cơ bản: đỏ (R), lục (G), lơ (B) và thường thu nhận trên các dải băng tần khác nhau. Với ảnh màu, cách biểu diễn cũng tương tự như với ảnh đen trắng, chỉ khác là các số tại mỗi phần tử của ma trận biểu diễn cho ba màu riêng rẽ gồm: đỏ (red), lục (green) và lum (blue). Để biểu diễn cho một điểm ảnh màu cần 24 bit. 24-bit này được chia thành ba khoảng 8 bit. Mỗi màu cũng phân thành L cấp màu khác nhau (thường L-256). Mỗi khoảng này biểu diễn cho cường độ sáng của một trong các màu chính.



Hình 3.17: Ảnh màu

3.1.2. Tổng quan về xử lý ảnh

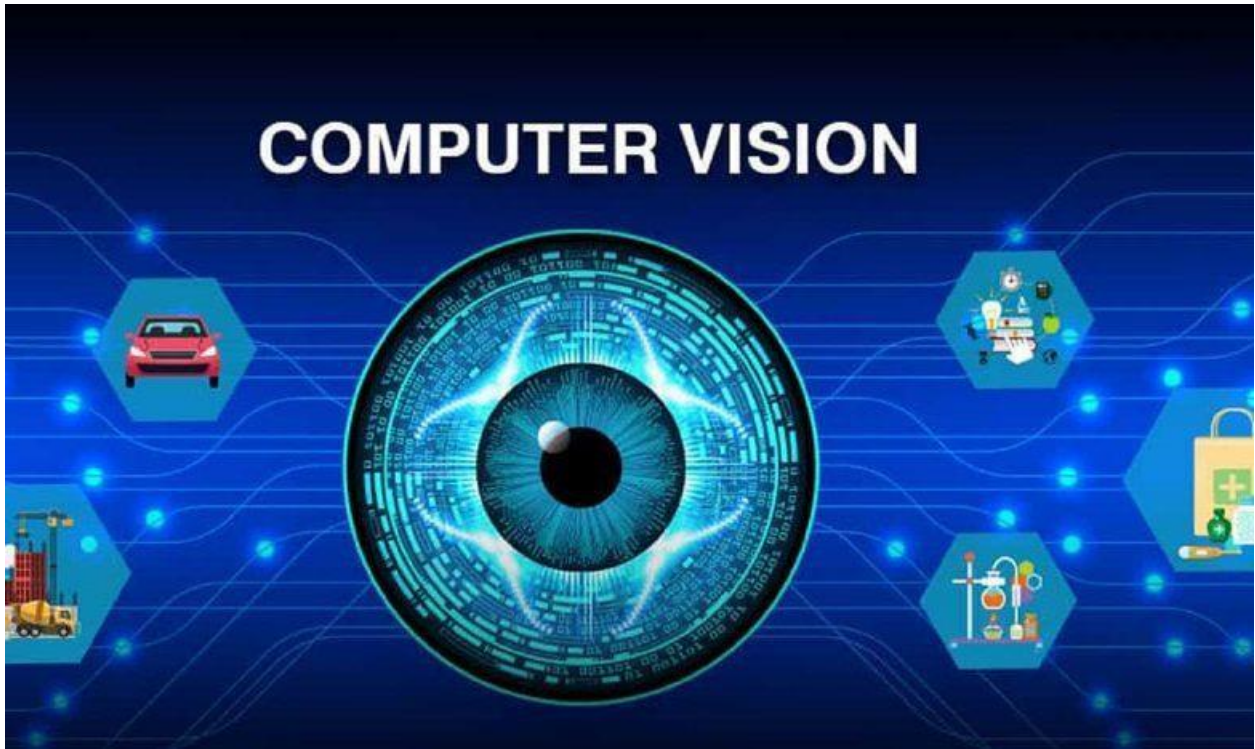
Xử lý ảnh là một chủ đề nghiên cứu trong lĩnh vực thị giác máy, nó là quá trình chuyển đổi một ảnh gốc thành một ảnh mới với các thuộc tính và theo sở thích của người dùng. Phân tích, phân loại tính năng, tăng cường, phân đoạn và tách cạnh, ghi nhãn vùng và thu thập thông tin hình ảnh đều là những ví dụ về xử lý hình ảnh. Xử lý ảnh kỹ thuật số, giống như xử lý dữ liệu với đồ họa, là một nhánh của tin học ứng dụng. Hình ảnh nhân tạo được coi là một cấu trúc dữ liệu và do máy tính tạo ra được gọi là xử lý dữ liệu đồ họa. Để truyền hoặc mã hóa ảnh tự nhiên, xử lý ảnh kỹ thuật số bao gồm các phương pháp và quy trình biến đổi. Sau đây là một số mục tiêu của xử lý hình ảnh:

- + Chỉnh sửa hình ảnh cải thiện chất lượng của hình ảnh.
- + Đánh giá nội dung hình ảnh, nhận dạng hình ảnh, và nhận dạng hình ảnh tự động.

Việc chia hình ảnh thành các phần có ý nghĩa để xác định mục này với mục khác, dựa trên đó cấu trúc của hình ảnh gốc có thể được đặc trưng, được gọi là nhận dạng và đánh giá nội dung hình ảnh. Có thể mô tả nhận dạng ảnh của các đối tượng trên ảnh, tách cạnh, phân đoạn ảnh và các phương pháp nhận dạng cơ bản khác. Công nghệ này nhận dạng các chữ cái trong văn bản và thường được sử dụng trong y tế (quy trình tế bào, nhiễm sắc thể, v.v.).

- Thị giác máy tính [11]

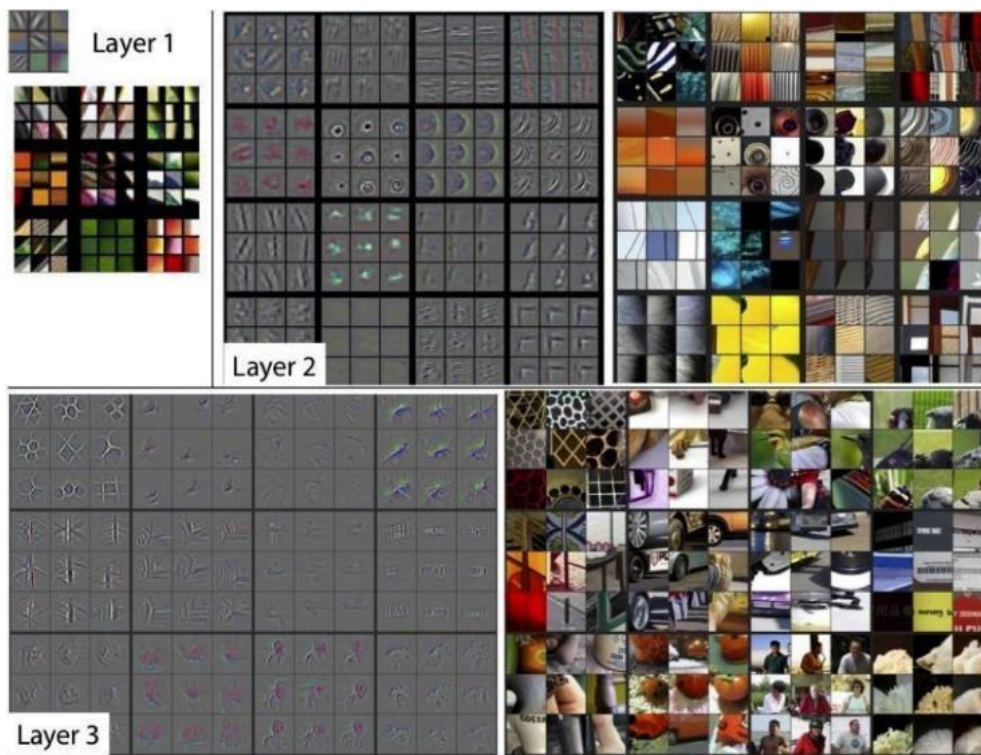
Thị giác máy tính là quá trình máy tính tự động phân tích ảnh và video để có được kiến thức tốt hơn về môi trường.



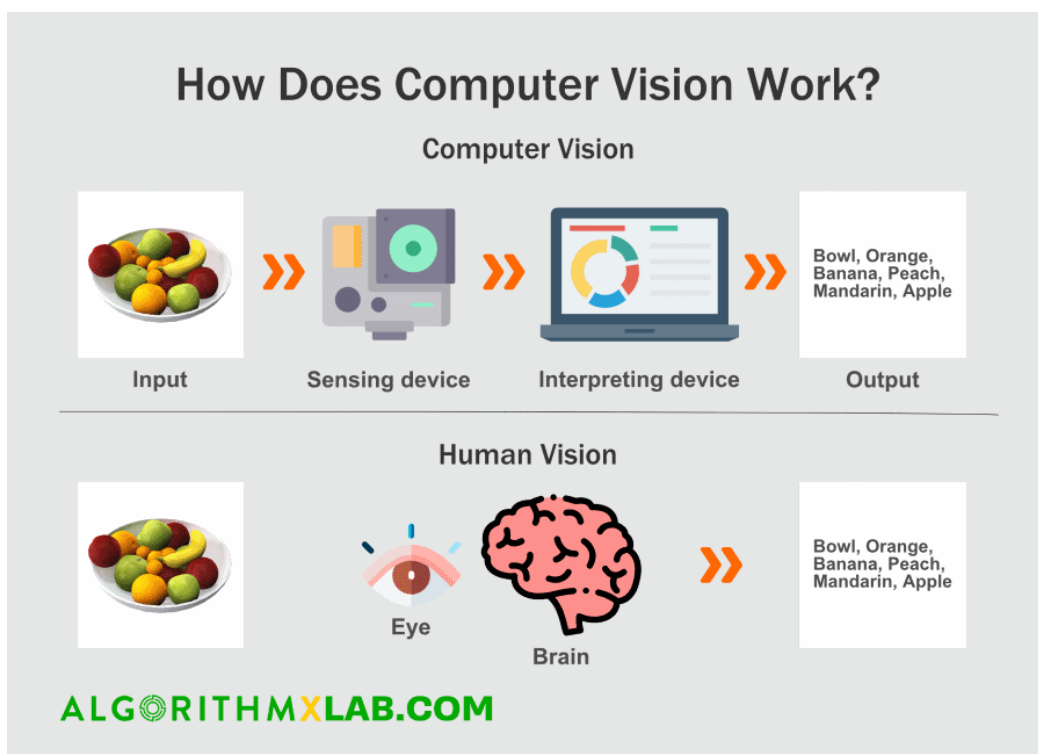
Hình 3.18: Thị giác máy tính

Thị giác máy tính được lấy cảm hứng từ khả năng của hệ thống thị giác con người và nó được coi là một vấn đề tương đối đơn giản để giải quyết khi nó lần đầu tiên được giải quyết vào những năm 1960 và 1970.

Vào những năm 1980s, nhà khoa học máy tính người Pháp Yan LeCun đã giới thiệu mạng thần kinh tích chập (convolutional neural network, CNN), một hệ thống AI lấy cảm hứng từ neocognitron của Fukushima. Một CNN bao gồm nhiều lớp tế bào thần kinh nhân tạo, các thành phần toán học mô phỏng gần giống hoạt động của các phiên bản sinh học của chúng.



Hình 3.19: Các lớp trên cùng của mạng thần kinh phát hiện các đặc trưng chung; các lớp sâu hơn phát hiện các đối tượng thực tế

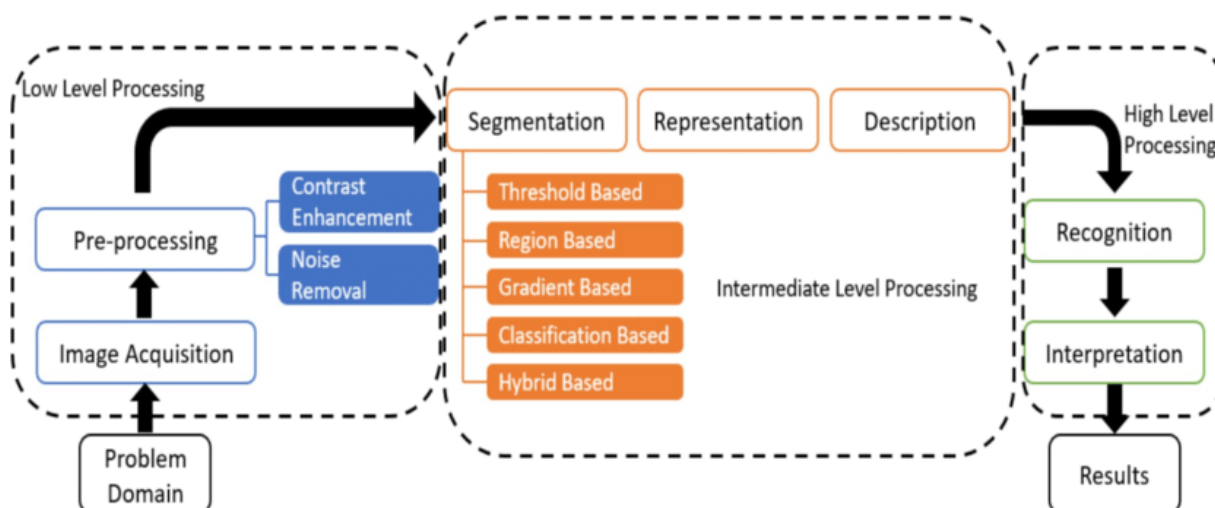


Hình 3.20: Cách thị giác máy tính hoạt động

Tuy nhiên, chúng tôi tin / cho rằng việc nhìn thấy đơn giản bởi vì chúng tôi có hệ thống thị giác của riêng mình, giúp công việc hiển thị tự nhiên đối với tâm trí có ý thức của chúng tôi. Trên thực tế, hệ thống thị giác ở người vô cùng phức tạp, và ước tính lượng não tham gia vào quá trình xử lý thị giác nằm trong khoảng từ 25% đến hơn 50%.

3.1.3. Các quy trình xử lý ảnh

Chụp ảnh là giai đoạn đầu tiên và quan trọng nhất trong quy trình. Hình ảnh đầu vào sẽ được nhận qua camera, cảm biến, máy quét và các thiết bị khác, sau đó các tín hiệu này sẽ được số hóa. Chất lượng của các đối tượng được phân tích sẽ xác định thiết bị thu nhận hình ảnh nào được sử dụng [12]. Độ phân giải, chất lượng màu sắc, dung lượng bộ nhớ và tốc độ thu nhận hình ảnh của thiết bị đều là những thông số quan trọng trong bước này.



Hình 3.21: Các giai đoạn chính trong xử lý ảnh

Tiền xử lý: Hình ảnh sẽ được cải thiện về độ tương phản, giảm nhiễu, loại bỏ bóng, độ lệch và các yếu tố khác nhằm nâng cao chất lượng hình ảnh và chuẩn bị cho các bước xử lý. Sau đó trong quá trình xử lý hình ảnh, mọi thứ trở nên phức tạp hơn rất nhiều. Bộ lọc thường được sử dụng để thực hiện thao tác này.

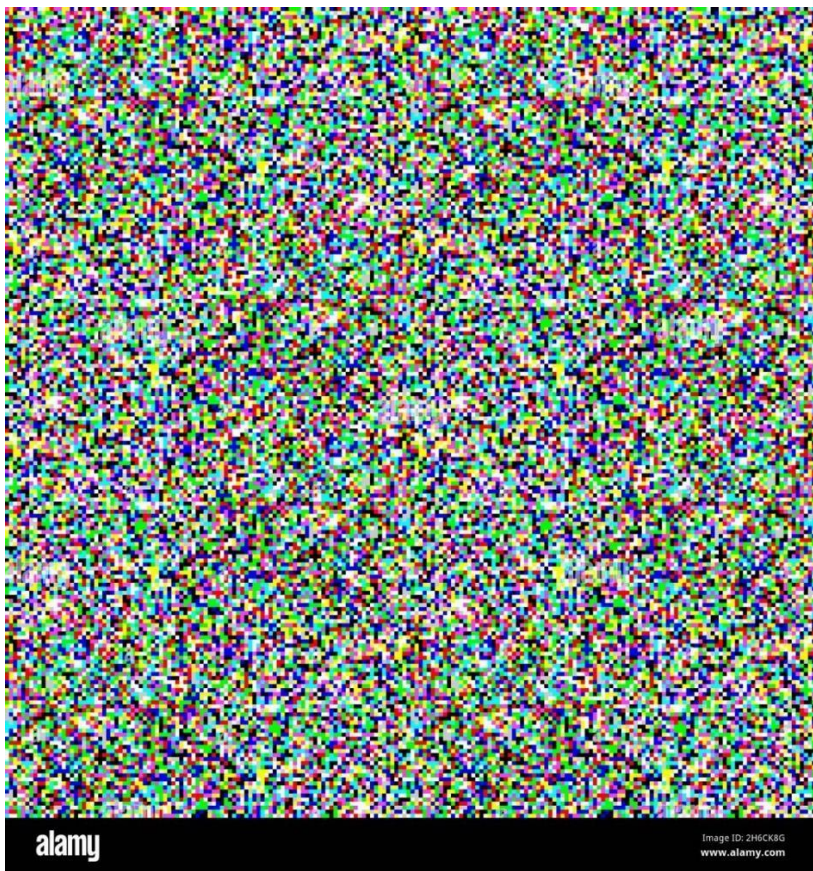
Phân đoạn ảnh là một giai đoạn quan trọng trong quá trình xử lý ảnh. Dựa trên đường viền hoặc các vùng liên quan, giai đoạn này chia hình ảnh thành các thành phần có các thuộc tính tương tự. Có thể sử dụng cùng một màu sắc, mức xám và các tiêu chí khác để biểu thị các vùng liên quan. Phân đoạn hình ảnh được sử dụng để cung cấp mô tả tổng hợp về các yếu tố khác nhau tạo nên một hình ảnh thô. Vì lượng thông tin có trong hình ảnh rất lớn và chúng ta chỉ cần trích xuất một vài tính năng trong hầu hết các ứng dụng, nên cần phải có một quy trình để giảm lượng dữ liệu không cần thiết. Thủ tục này chủ yếu yêu cầu phân đoạn hình ảnh và trích xuất đối tượng địa lý.

Tách các đối tượng: Đầu ra của bước phân đoạn thường là dữ liệu pixel thô, chứa ranh giới của vùng ảnh hoặc tập hợp tất cả các pixel thuộc vùng đó. Trong cả hai trường hợp, dữ liệu thô phải được chuyển đổi thành một định dạng mà máy tính có thể xử lý được. Câu hỏi đầu tiên cần trả lời để chuyển đổi chúng là liệu một vùng hình ảnh nên được biểu diễn dưới dạng ranh giới hay là một vùng hoàn chỉnh chứa tất cả các pixel thuộc về nó. Biểu diễn ranh giới cho một khu vực thích hợp cho các ứng dụng tập trung vào các khía cạnh hình thái bên ngoài của đối tượng, chẳng hạn như các cạnh và điểm uốn biên. Biểu diễn vùng thích hợp cho các ứng dụng sử dụng chất lượng bên trong của một đối tượng, chẳng hạn như kết cấu hoặc cấu trúc xương. Việc chọn biểu diễn phù hợp cho một vùng hình ảnh chỉ là một bước trong quá trình chuyển đổi dữ liệu hình ảnh thô thành một định dạng có thể được xử lý thêm. Chúng ta cũng phải nghĩ ra một phương tiện xác định dữ liệu đã thay đổi theo cách làm nổi bật các phẩm chất quan tâm, giúp cho việc xử lý dễ dàng hơn.

Giai đoạn cuối cùng trong xử lý ảnh là nhận dạng và giải thích. Việc ghi nhận các mục trong hình ảnh là một định nghĩa đơn giản về nhận dạng hình ảnh. Ví dụ, các đối tượng trong ảnh cần được phát hiện để nhận dạng chữ viết tay là các mẫu chữ cái; chúng ta phải tách các mẫu chữ cái đó ra và tìm ra phương pháp để gán các chữ cái liên quan trong bảng chữ cái cho các mẫu chữ cái đã thu thập một cách chính xác. xuất hiện trong bức ảnh Quá trình truyền đạt ý nghĩa cho một tập hợp các mục đã biết được gọi là diễn giải. Do đó, có thể thấy rằng không phải mọi ứng dụng xử lý ảnh đều cần thiết để hoàn thành tất cả các bước xử lý nói trên; chẳng hạn, phần mềm chỉnh sửa ảnh chỉ dừng lại ở giai đoạn tiền xử lý. Nói chung, các tác vụ xử lý như nhận dạng và diễn giải chỉ được tìm thấy trong các hệ thống phân tích hình ảnh tự động hoặc bán tự động trích xuất thông tin quan trọng từ hình ảnh, chẳng hạn như nhận dạng ký tự quang học, nhận dạng chữ viết tay, v.v.

- Hình ảnh và biểu diễn hình ảnh:

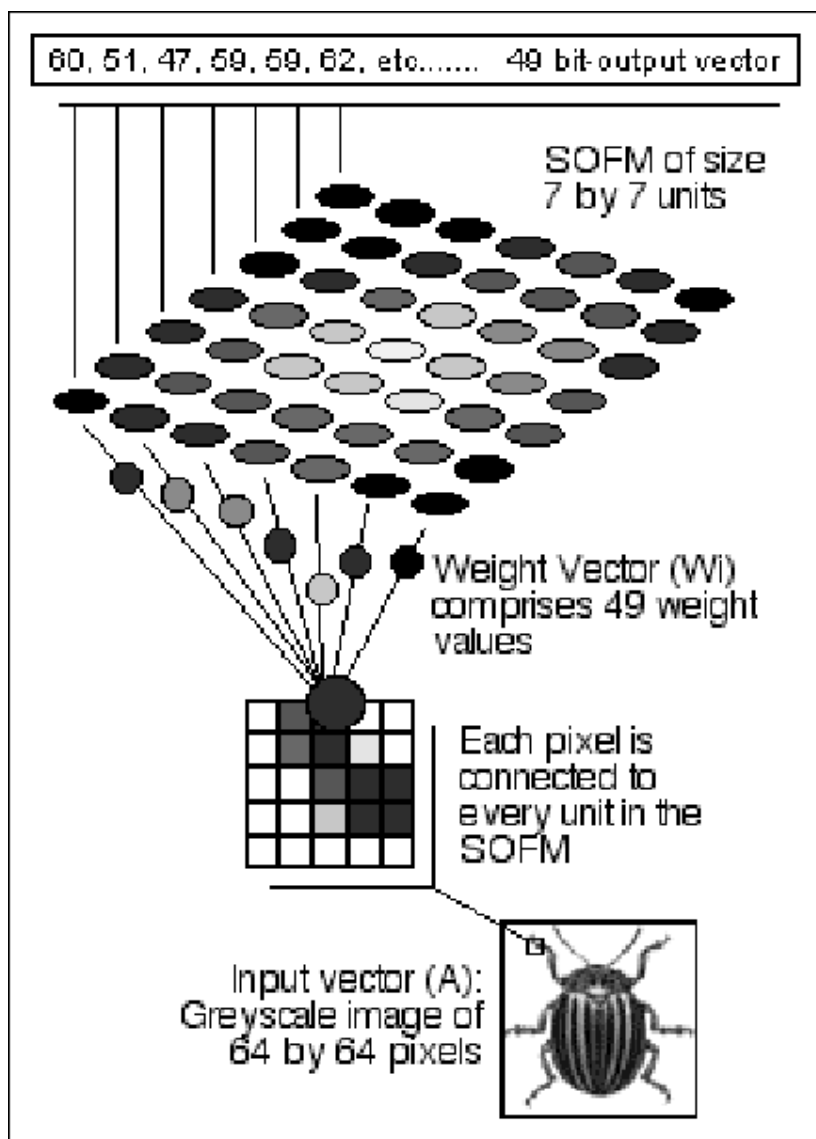
Về cả giá trị không gian và độ sáng, ảnh là ảnh liên tục. Số hóa hình ảnh là bắt buộc để xử lý hình ảnh bằng máy tính. Quá trình chuyển đổi tín hiệu liên tục thành tín hiệu rời rạc bằng cách lấy mẫu (phân biệt không gian) và lượng tử hóa các thành phần giá trị mà về nguyên tắc không thể phân biệt được bằng mắt thường. hai điểm gần nhau Điểm ảnh (Picture Elements), yếu tố hình ảnh hoặc pixel là tên được đặt cho những điểm này. Điều quan trọng là phải phân biệt giữa các pixel và tham chiếu trong các hệ thống đồ họa máy tính. Để tránh hiểu lầm, chúng tôi gọi ý tưởng pixel này là pixel thiết bị. Khái niệm pixel thiết bị có thể được giải thích như sau: khi chúng ta nhìn vào một màn hình (ở chế độ đồ họa), chúng ta thấy rằng nó được tạo thành từ các chấm nhỏ gọi là pixel chứ không phải là một hình ảnh liên tục. Mỗi pixel được tạo thành từ một tập hợp các tọa độ x, y và màu.



Hình 3.22: Pixel

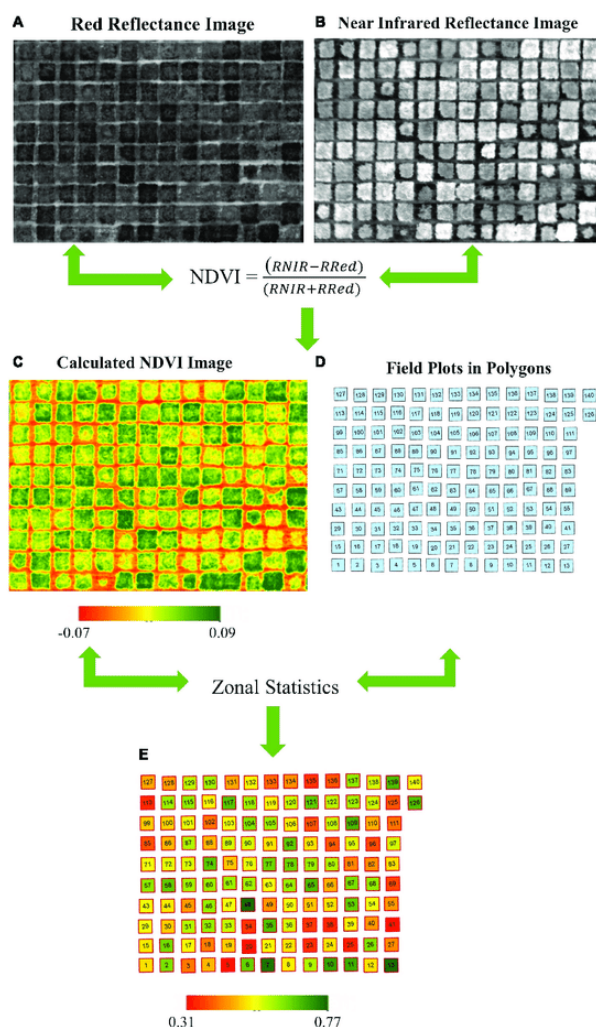
Hình ảnh có thể được biểu diễn bằng mô hình Vector hoặc mô hình Raster [13].

+ Mô hình vector: Ngoài việc giảm không gian lưu trữ và dễ hiển thị và in ấn, ảnh vector còn cho phép dễ dàng chọn, sao chép, di chuyển và tìm kiếm. Hình ảnh gốc được mã hóa và tái tạo lại bằng cách sử dụng hướng vector của các pixel lân cận trong mô hình này. Hình ảnh vector có thể được lấy trực tiếp từ các thiết bị kỹ thuật số như Digitalize hoặc chuyển đổi từ ảnh raster bằng phần mềm vector hóa.



Hình 3.23: Mô hình vector

+ Mô hình raster hiện là mô hình biểu diễn hình ảnh được sử dụng rộng rãi nhất. Hình ảnh được biểu diễn bằng ma trận pixel. Mỗi pixel có thể được biểu diễn bằng một hoặc nhiều bit, tùy thuộc vào nhu cầu. Để chụp, hiển thị và in, mô hình Raster là lý tưởng. Những hình ảnh được sử dụng trong cuộc điều tra này cũng là những hình ảnh mà mô hình Raster đại diện.



Hình 3.24: Mô hình Raster

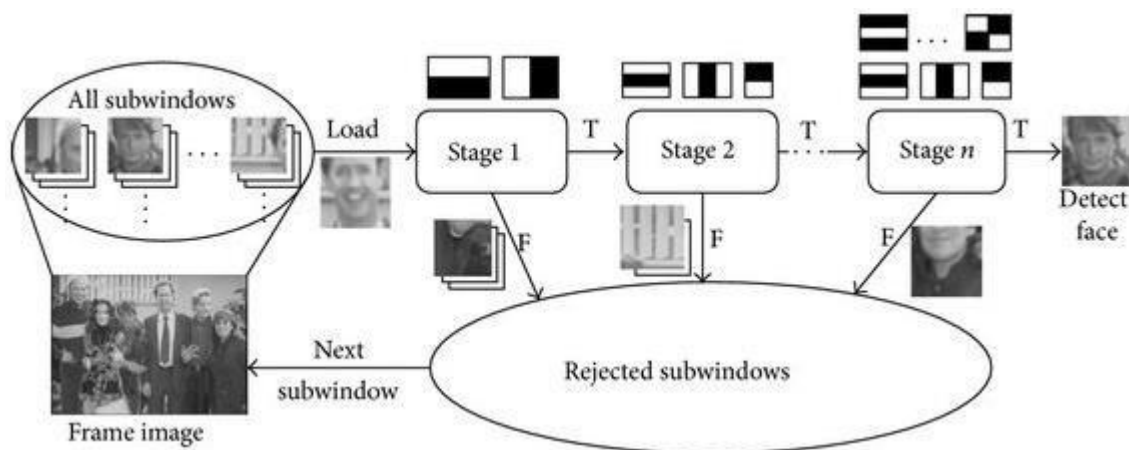
3.2. Kết quả nghiên cứu giải thuật nhận diện chiếc xe vượt ngưỡng tốc độ cho phép

Có 2 phương pháp để nhận diện chiếc xe

- Phương pháp Haar Cascade [14]:

Là một kỹ thuật dựa trên tính năng để phát hiện mọi thứ (khuôn mặt, mắt, tay, đồ vật, v.v.) được trình bày bởi Paul Viola và Michael Jones trong bài báo của họ vào năm 2001. "Phát hiện đối tượng nhanh chóng bằng cách sử dụng Cascades đơn giản được nâng cao tính năng", nghiên cứu nói.

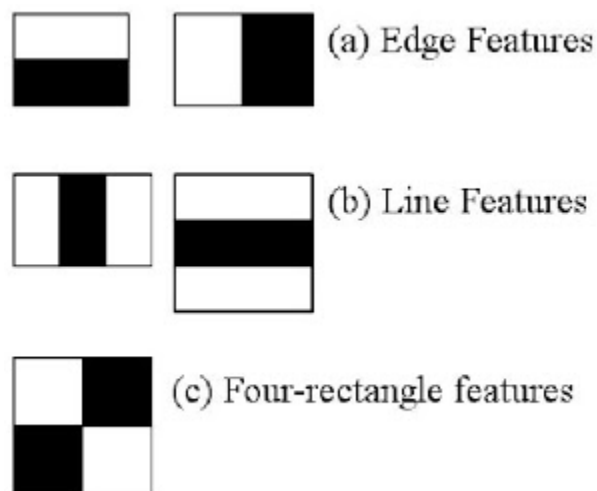
Hệ thống lần đầu tiên được sử dụng để phát hiện khuôn mặt trực diện và các đặc điểm như mắt, mũi và miệng. Tuy nhiên, trong GitHub của họ, có rất nhiều tính năng Haar đã được học trước đây cho các đối tượng khác nhau, như toàn bộ cơ thể, thân trên, thân dưới, nụ cười và nhiều tính năng khác.



Hình 3.25: Quy trình để nhận diện một khuôn mặt bằng phương pháp Haar Cascade

Về cơ bản nó là sử dụng các tính năng kiểu Haar và sau đó sử dụng càng nhiều các tính năng đó qua nhiều lượt (thác) để tạo thành một công cụ nhận dạng hoàn chỉnh. Nói một cách dễ hiểu, các dòng thác này đều có sẵn trên các trang web và bạn chỉ cần tải về máy và gọi điện là có thể sử dụng.

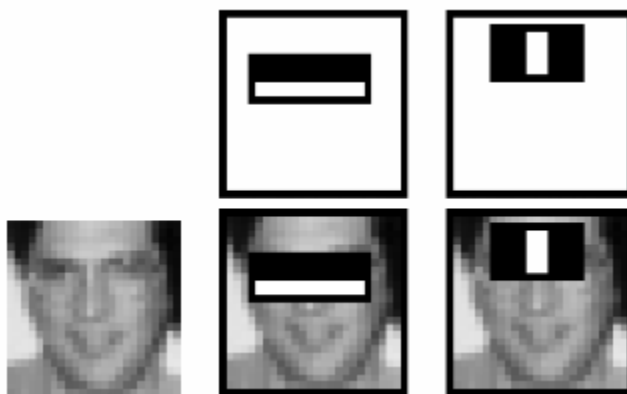
Cách thức hoạt động và các bước thực hiện: Như được chỉ ra trong hình, các đặc điểm hình chữ nhật tương tự như hạt nhân được sử dụng để phát hiện các đặc điểm khuôn mặt khác nhau như mắt và nốt sần được áp dụng cho hình ảnh.



Hình 3.26: Dựa vào các góc cạnh để xác định hình chữ nhật

- a) Chúng có phải là các bộ lọc thu được các cạnh hình ảnh không
- b) Chúng có phải là bộ lọc để lấy các đường thẳng trong hình ảnh không
- c) Chúng có phải là bộ lọc về đặc điểm của 4 hình vuông không

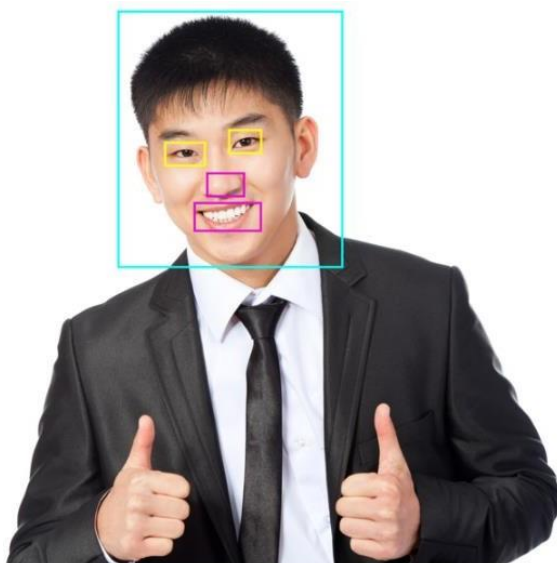
Tính năng hình chữ nhật đầu tiên trong hình dưới đây tính toán sự khác biệt về cường độ giữa các vùng mắt và má trên khuôn mặt. Sự khác biệt về cường độ giữa mắt và sống mũi được đo bằng đặc điểm hình chữ nhật thứ hai. Để nhận dạng, bộ lọc Haar chỉ có thể nhìn vào một khu vực nhất định trong cửa sổ.



Hình 3.27: Cách thức hoạt động khi nhận diện mặt người

Tạo ra hàng nghìn đặc điểm cho một bức tranh bằng cách sử dụng các đặc điểm hình chữ nhật này. Mặt khác, tính toán tổng số pixel trong các phần trắng và đen của ảnh có thể là một quá trình tốn nhiều thời gian, đặc biệt là đối với những bức ảnh lớn.

Chỉ với bốn pixel để làm việc, hình ảnh tích phân có thể hoàn thành cùng một phép tính. Hàng nghìn bức ảnh có chú thích có thể được sử dụng để xây dựng bộ phân loại nhận dạng khuôn mặt bằng cách chuyển đổi chúng thành bản đồ tính năng lớp HAAR và đào tạo nó bằng mô hình học máy.



Hình 3.28: Kết quả nhận diện khuôn mặt với các ô chữ nhật khác nhau có nhiệm vụ nhận diện đối tượng mắt, mũi và miệng

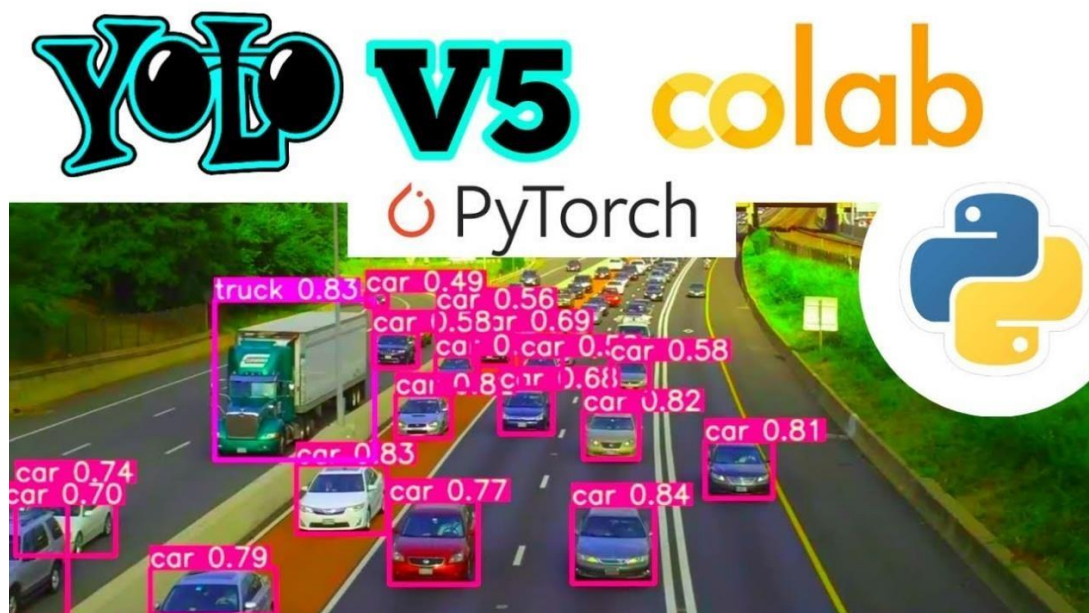
Bước 1: Chia nhỏ hình ảnh (đã được cung cấp cho bộ phân loại) thành các phần con (hoặc các cửa sổ con như trong hình minh họa).

Bước 2: Chúng tôi tạo một dòng gồm N không có bộ dò, mỗi bộ phát hiện sự kết hợp của một số loại đặc trưng từ các hình ảnh được truyền (ví dụ: đường thẳng, cạnh, hình tròn, hình vuông). Giả sử rằng mỗi tiểu mục được cấp phát một giá trị tin cậy trong quá trình trích xuất đối tượng địa lý.

Bước 3: Khuôn mặt được nhận diện là ảnh (hoặc ảnh phụ) với độ tin cậy cao nhất và nộp cho bộ tích lũy, trong khi phần còn lại bị loại. Cascade sau đó tìm nạp khung / hình ảnh tiếp theo nếu có và lặp lại quá trình.

- Phương pháp YOLOv5:

Mô hình phát hiện đối tượng YOLOv5 là một phần của dòng mô hình YOLO. Joseph Redmon đã tạo ra ba phiên bản đầu tiên của YOLO. Sau đó, Alexey Bochkovskiy đã sản xuất YOLOv4, một phiên bản nhanh hơn và chính xác hơn của YOLO. Sau đó, có YOLOv5, mới được phát hành và có các so sánh sơ bộ cho thấy nó chính xác như YOLOv4 đồng thời nhanh hơn khi đưa ra dự đoán (mặc dù vẫn còn nhiều hoài nghi về độ tin cậy của các so sánh). Điều này là do YOLOv5 chỉ mới được tải lên GitHub gần đây và vẫn chưa có tài liệu chính thức).



Hình 3.29: Nhận diện phương pháp Yolov5

Không giống như những người tiền nhiệm, YOLOv5 được xây dựng bằng PyTorch chứ không phải DarkNet. Đây là một điểm cộng đáng kể cho YOLOv5 vì PyTorch phổ biến hơn rất nhiều, do đó sẽ có nhiều tài liệu và hướng dẫn hơn để chúng ta chuyển sang.

Cách thức hoạt động và các bước thực hiện:

** Bộ sưu tập các bức ảnh*

Có một số lựa chọn để thu thập ảnh để đào tạo người mẫu:

+ Tìm kiếm Tập dữ liệu ảnh

mở(<https://storage.googleapis.com/openimages/web/index.html>), chứa hơn 9 triệu hình ảnh được sắp xếp thành 6000 lớp khác nhau.

+ Sử dụng các công cụ tìm kiếm (Google, Bing...) kèm theo ghi chú để kiểm tra kỹ bản quyền của hình ảnh.

+ Xóa khung hình khỏi video

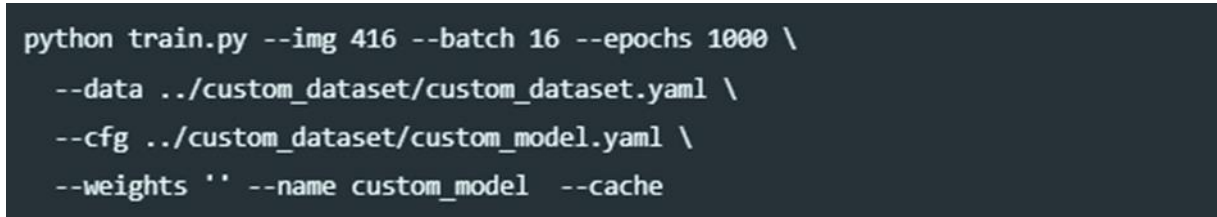
+ Tự chụp ảnh

** Dán nhãn*

Sử dụng thư viện labelImg để dán nhãn và tôi mất hơn 1 giờ để gắn nhãn cho khoảng 130 hình ảnh.

** Huấn luyện mẫu*

```
python train.py --img 416 --batch 16 --epochs 1000 \  
  --data ../custom_dataset/custom_dataset.yaml \  
  --cfg ../custom_dataset/custom_model.yaml \  
  --weights '' --name custom_model --cache
```

Hình 3.30: Câu lệnh huấn luyện mẫuimg: kích thước hình ảnh

(độ phân giải)

batch: số lượng hình ảnh được sử dụng để huấn luyện trong mỗi lượt

epochs: Số lần huấn luyện tất cả các hình ảnh trong tập dữ liệu đào tạo

data: đường dẫn đến tệp cấu hình của bộ dữ liệu

cfg: đường dẫn đến tệp cấu hình của mô hình

weights: đường dẫn đến tệp weight có chứa các kết nối giữa các nơ-ron (vì vậy "" là để đào tạo từ đầu)

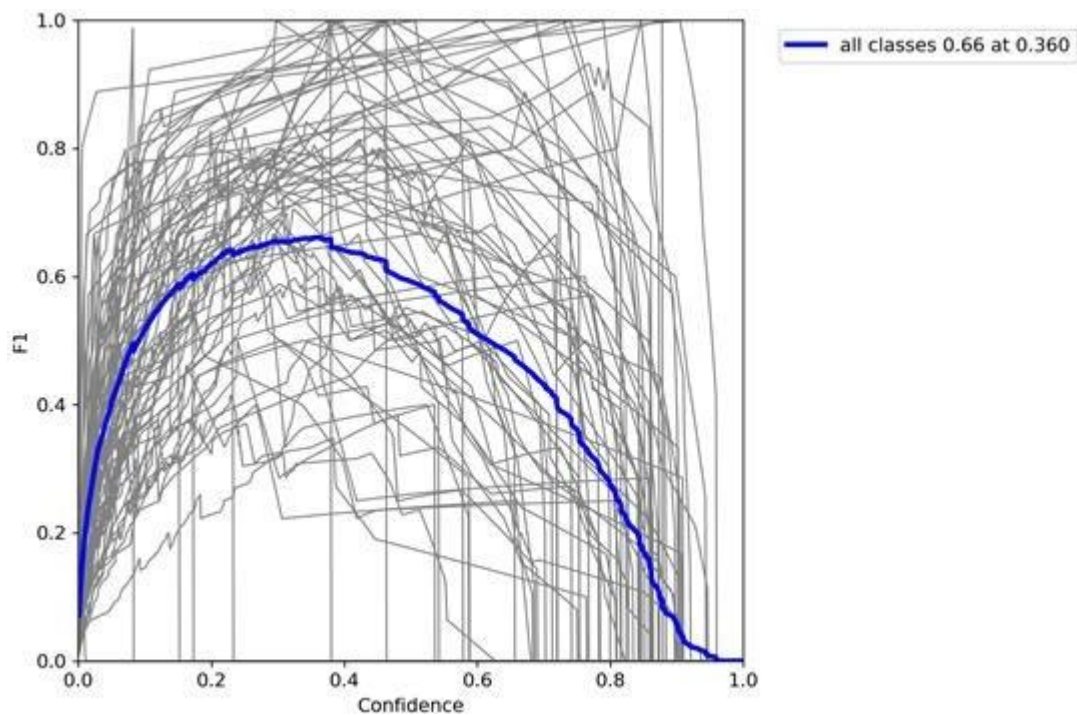
name: tên thư mục để lưu mô hình

cache: sử dụng cache để đào tạo nhanh hơn

- Kết quả



Hình 3.31: Kết quả sau khi huấn luyện

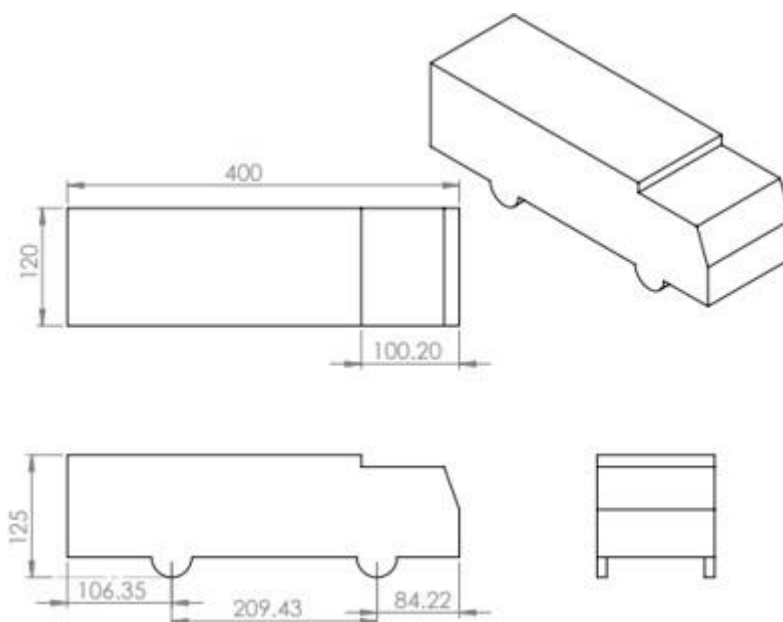


Hình 3.32: Kết quả ở dạng biểu đồ

3.3. Kết quả nghiên cứu thiết kế và lắp đặt phần cứng

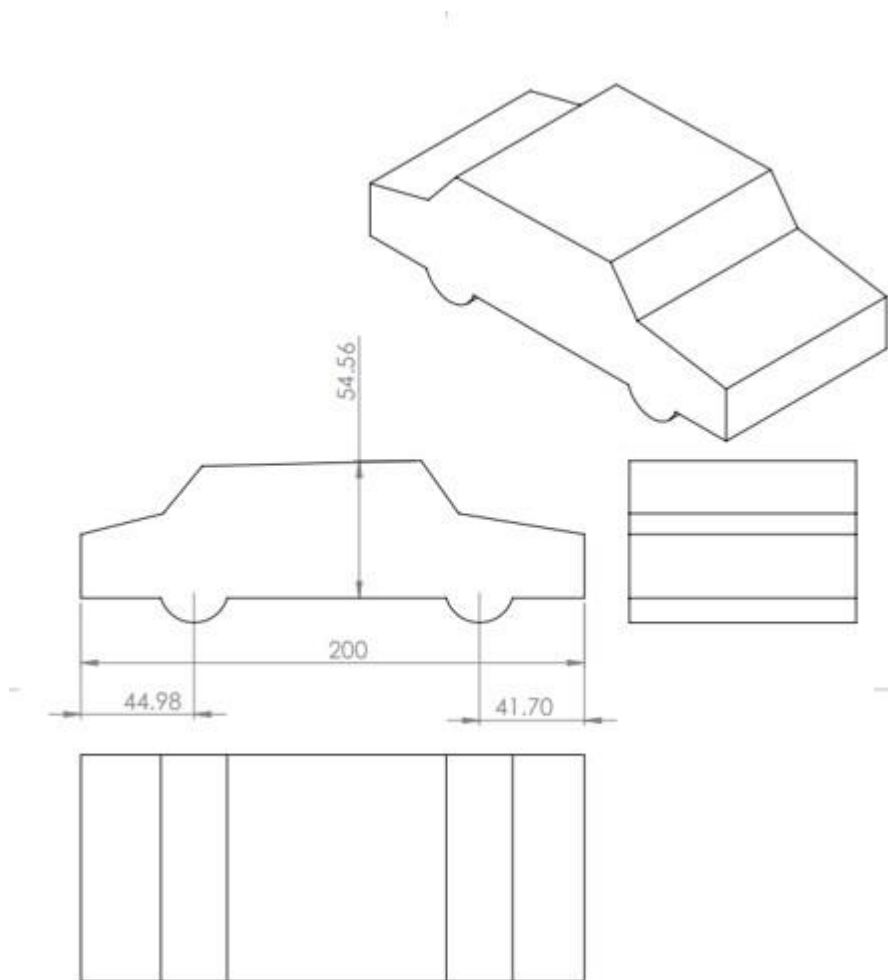
3.3.1. Nghiên cứu thiết kế

- Bản thiết kế mô hình xe tải:



Hình 3.33: Bản thiết kế mô hình xe tải

- Bản thiết kế ô tô con:

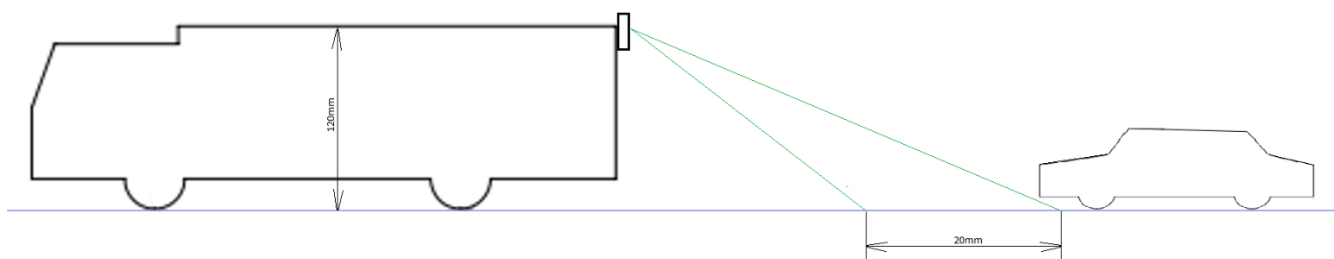


Hình 3.34: Bản thiết kế mô hình xe ô tô con

Mô hình hóa:

-Khoảng cách 2 lane $S = 200\text{mm}$, chiều cao camera = 120mm ,

$$V = \frac{S}{(T2-T1)}$$



Hình 3.35: Bản vẽ mô hình hóa hệ thống

3.3.2. Nghiên cứu lắp đặt phần cứng

Sơ đồ lắp đặt:

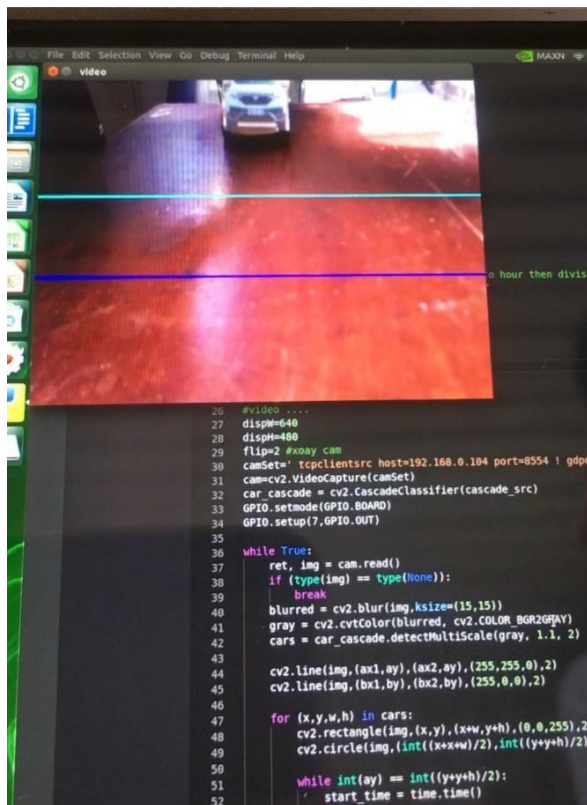


Hình 3.36: Sơ đồ lắp ráp các máy tính nhúng và linh kiện

Mô hình thực hiện với raspberry pi 4 kết nối với camera, jetson nano liên kết với màn hình và còi, 2 cục nguồn cấp điện cho raspberry và jetson. Raspberry và jetson giao tiếp với nhau bằng mạng wifi. Tín hiệu được thu thập trực tiếp bằng camera và raspberry chuyển dữ liệu đầu vào đến jetson. Tại jetson, dữ liệu được xử lý bằng chương trình python để xuất ra tín hiệu thông qua còi.

3.4. Vận hành module về khả năng xử lý và cảnh báo bằng mô hình thực nghiệm.

Để hệ thống có thể hoạt động tốt môi trường thực nghiệm cần đáp ứng một số điều kiện về độ sáng hay tốc độ xe cần nhận dạng. Ánh sáng của môi trường thực nghiệm không được quá tối hay quá sáng, nếu thực nghiệm ở môi trường quá sáng có thể làm chói cho xe ô tô con, điều này sẽ gây khó khăn trong việc nhận dạng ô tô con. Vận tốc di chuyển của ô tô con cũng là yếu tố then chốt quyết định sự thành công trong quá trình thử nghiệm. Để thí nghiệm đạt kết quả nhận diện chính xác nhất thì tốc độ của phương tiện phải ổn định. Bên cạnh đó, tốc độ phương tiện thực nghiệm không được quá cao vì hệ thống vẫn còn bị ràng buộc rất nhiều vào chất lượng của phần cứng. Hệ thống vẫn còn bỏ sót một số trường hợp không nhận dạng được trên mô hình, tỉ lệ nhận dạng phương tiện ở khoảng 8/10 số lần thực nghiệm. Tỉ lệ này có thể thay đổi tùy thuộc vào các yếu tố độ sáng môi trường thực nghiệm hay tốc độ di chuyển của xe con.



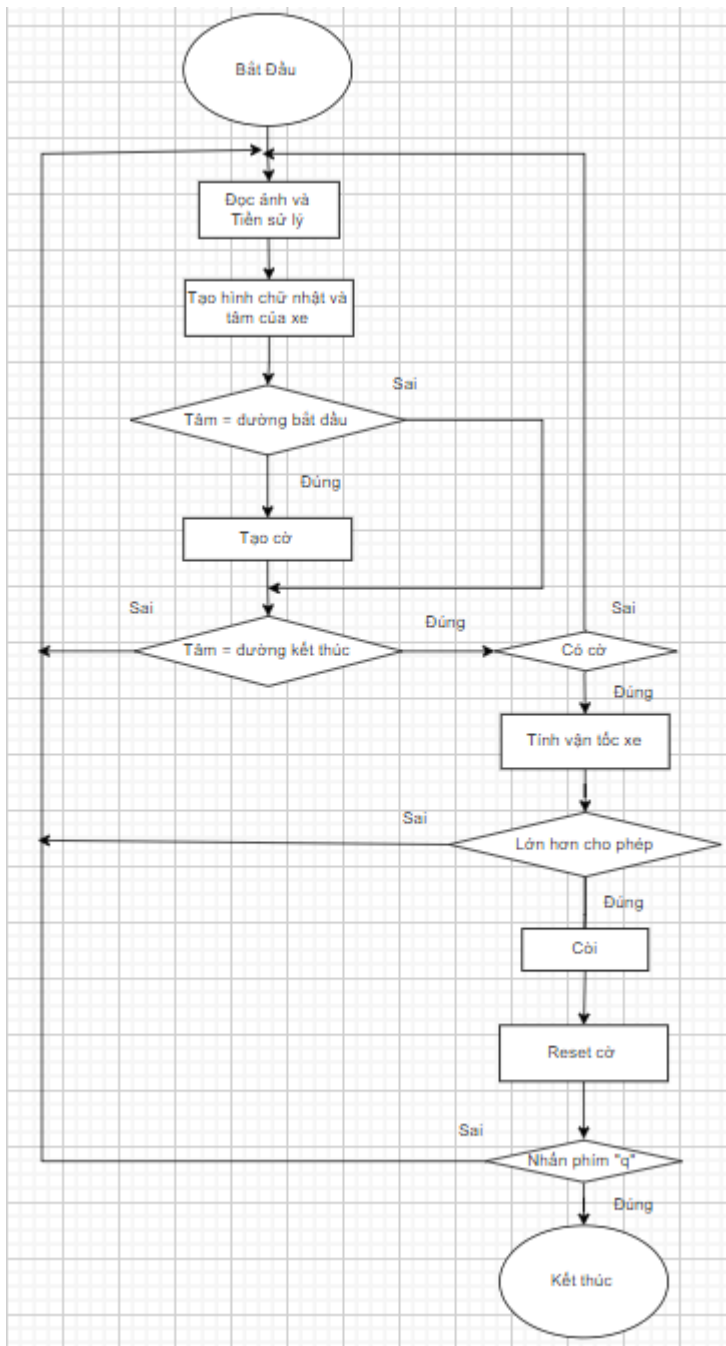
Hình 3.37: Mô hình thực tế đang vận hành

Đánh giá về khả năng vận hành của hệ thống:

Việc thực nghiệm trên mô hình thực tế với camera thu hình xe ô tô con có tỉ lệ nhận diện thấp hơn khi nạp trực tiếp một video vào chương trình mà không sử dụng đến camera hay sử dụng mô hình nhưng đặt một điện thoại có mở video phương tiện lưu thông trên đường. Vì hệ thống sử dụng cascade nhận diện phương tiện thật nên việc sử dụng các xe mô hình có thể không chính xác như việc sử dụng để nhận dạng trên xe thực. Mặc dù, xe mô hình có thể nhìn thấy giống xe thật bằng mắt người nhưng việc xử lý trên máy tính có thể gặp khó khăn vì xe mô hình có thể không đáp ứng đủ các đặc điểm thực tế của xe thật.

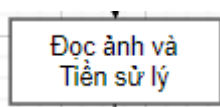
Hệ thống tính được tốc độ của phương tiện di chuyển ở mức chính xác tương đối cao. Do camera được lắp với raspberry nên chỉ cần raspberry được cấp nguồn thì camera có thể hoạt động được và sự truyền dẫn dữ liệu được cải thiện thành không dây nên đã tách biệt hoàn toàn với bộ phận xử lý thông tin và phát tín hiệu. Vì vậy sản phẩm có thể lắp đặt ở bất cứ đâu trên xe và chỉ cần chỉnh sửa lại một vài thông số là đã có thể hoạt động. Bên cạnh đó bộ phận xử lý là jetson nano chỉ cần kết nối với loa và màn hình nên có thể được cấp nguồn ở khu vực bảng điều khiển là có thể hiển thị hình ảnh và cảnh báo âm thanh đến người dùng mà không cần dùng giao thức kết nối qua dây cáp truyền thông như HDMI và dây nguồn giữa raspberry và jetson. Điều này giúp đơn giản hóa hệ thống cũng như tính di động cao của hệ thống. Ngoài ra việc tách camera ra khỏi jetson và lắp đặt trên raspberry giúp giảm tải lượng công việc xử lý đồng thời cùng lúc của jetson nên cải thiện đáng kể được độ trễ của camera.

3.5. Kết quả nghiên cứu sơ đồ giải thuật



Hình 3.38: Sơ đồ giải thuật của của việc nhận dạng xe

Chương trình



Hình ảnh thu được từ camera sẽ đi qua bộ lọc để giảm lượng thông tin xử lý so với việc để màu nguyên bản

```

while True:
ret, img = cam.read() #Thu nhận hình ảnh
if (type(img) == type (None)):
break
blurred = cv2.blur(img,ksize=(15,15)) #Bộ lọc làm mờ ảnh
gray = cv2.cvtColor(blurred, cv2.COLOR_BGR2GRAY) #Chuyển ảnh thành
màu xám
cars = car_cascade.detectMultiScale(gray, 1.1, 2) #Sử dụng cascade để nhận diện
xe

```

Tạo hình chữ nhật và
tâm của xe

Sau khi nhận diện được chiếc xe ta đóng khung hình chữ nhật bao quanh chiếc xe và tạo thêm vòng tròn ở trung tâm hình chữ nhật

```

cv2.rectangle(img, (x,y),(x+w,y+h),(0,0,255),2) #Tạo hình chữ nhật
cv2.circle(img, (int((x+x+w)/2), int((y+y+h)/2)),1, (0,255,0), -1) #Tạo hình
tròn

```

Tâm = đường bắt đầu

```

while int(ay-10) <= int((y+y+h)/2) and int(ay+10) >= int((y+y+h)/2): start_time =
time.time() #Sau đó khi tâm vòng tròn chạm đường bắt đầu ta sẽ
có T1

```

Tạo cờ

$z=1$ #Tiếp đến tạo 1 cờ $z = 1$. Cờ có nhiệm vụ đánh dấu xe đã chạm vạch bắt đầu hay chưa

Tâm = đường kết thúc

```

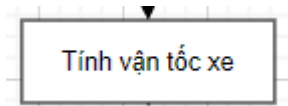
while int(ay) <= int((y+y+h)/2): #Sau khi tâm vòng tròn chạm vào đường kết thúc ta
sẽ có T2

```

Có cờ

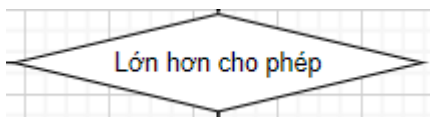
Cờ lúc này đã có

if int(by-10) <= int((y+y+h)/2) and int(by+10) >= int((y+y+h)/2) and z==1: #Có còi mới cho vào câu điều kiện if, trong câu điều kiện if có lấy thời gian lúc chạm vạch kết thúc. Nếu không có còi tức là không có thời gian chạm vạch kết thúc thì không thể tính được vận tốc.



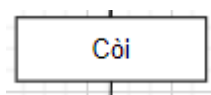
Tính tốc độ xe vừa nhận được

Speed = (10*3600)/(time1*1000) + 30 #tốc độ xe bằng khoảng cách thiết lập chia cho khoảng thời gian ghi nhận, đơn vị của thời gian là giây nên đổi sang giờ, đơn vị của độ dài là m đổi sang km. Đơn vị cuối cùng của vận tốc là km/h.



Câu điều kiện

if Speed > 60: #Nếu vận tốc lớn hơn vận tốc thiết lập thì xuất ra tín hiệu cảnh báo



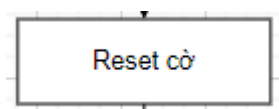
Xuất ra tín hiệu cảnh báo

GPIO.output(7,1) #còi hoạt động t1=

time.time()

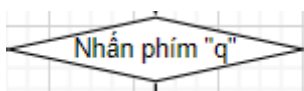
while (time.time() - t1)<1: #còi hoạt động duy trì trong 1 giây pass

GPIO.output(7,0) #còi dừng hoạt động



Sau khi xuất ra tín hiệu qua còi thì còi được reset để chuẩn bị cho lần hoạt động tiếp theo

z=0 #reset lại biến z



Nếu muốn thoát ra khỏi chương trình thì nhấn phím “q” để dừng vận hành hệ thống

if cv2.waitKey(5) == ord('q'): #Cần nhấn phím “q”

```
break #Dừng hoạt động chương trình chính  
cam.release() #Tắt camera  
cv2.destroyAllWindows() #Kết thúc chương trình
```

4. Kết luận và ý nghĩa của đề tài

4.1. Ý nghĩa về mặt khoa học

Đề tài nghiên cứu xử lý ảnh thông qua camera để cảnh báo người dùng đã đưa ra giải pháp là sử dụng 2 máy tính nhúng là Jetson Nano và Raspberry pi nhằm tính toán tốc độ xe lưu thông cảnh báo tài xế để gián tiếp giải quyết một phần nguyên nhân dẫn đến tai nạn ô tô. Đây là một trong những đề tài mới trong lĩnh vực công nghệ ô tô hầu hết các ứng dụng xử lý ảnh chỉ dừng lại ở điểm số lượng xe và nhận diện xe.

Hiện nay hầu hết việc ứng dụng xử lý ảnh chỉ toàn tập trung vào đếm số lượng xe lưu thông và các ứng dụng trên xe để tránh lệch làn đường. Ngoài ra, phương pháp xử lý bằng sóng siêu âm đã được ứng dụng và nghiên cứu nhiều trong việc cảnh báo an toàn va chạm điểm mù nhưng việc xử lý bằng hình ảnh thì vẫn còn tương đối hiếm và mới mẻ. Tính mới của đề tài là ứng dụng xử lý ảnh là nhận diện phương tiện và tính toán tốc độ phương tiện.

4.2. Đóng góp về phương diện xã hội & kinh tế

Giải quyết được phần nào đó vấn đề an toàn điểm mù, đây là nghiên cứu có tính cấp thiết góp phần giảm thiểu tai nạn giao thông ở các xe có kích thước lớn trong những trường hợp xác suất xảy ra tai nạn cao như xe đổi hướng di chuyển, vào cua hay quay đầu. Đây là hệ thống có thể trang bị thêm trên bất cứ phương tiện nào có kích thước lớn nên có tính thương mại hóa cao.

4.3. Định hướng phát triển trong tương lai

- Dựa trên những nghiên cứu và thí nghiệm thu được, hệ thống vẫn còn rất nhiều hạn chế cần khắc phục, cải tiến và phát triển hơn để có thể tạo ra một hệ thống hoạt động với độ chính xác và ưu việt cao cũng như giảm về mặt giá thành để sản phẩm có thể tiếp cận với nhiều người sử dụng hơn.
- Có thể cải tiến để nhận diện được cả xe máy và người ở tất cả các điểm mù mà tài xế không thể quan sát tới điều này mang lại sự tiện lợi và góp phần giúp tài xế tập trung hơn trong việc lái xe góp phần giảm thiểu nguyên nhân dẫn đến tai nạn ô tô.
- Đồng thời, cải tiến việc tắt mở cảnh báo thông qua giọng nói, giúp tài xế có thể tập trung hơn trong việc lái xe mà không cần phải bấm cái nút vật lý.
- Ngoài ra nếu thiết bị này được nhiều xe sử dụng từ đó tại em có thể kết nối giữa các thiết bị với nhau thành một mạng lưới duy nhất, sử dụng thông tin đó ứng dụng GPS để theo dõi tất cả các xe và đo lường khoảng cách thông qua GPS nâng cao tỉ lệ chính xác hơn và đồng bộ hơn.

- Hiện tại trên thị trường có rất nhiều thiết bị cảnh báo tiền va chạm nếu có thể em sẽ kết hợp hệ thống của em với những thiết bị đó làm tăng độ chính xác từ đó làm giảm hầu hết các vụ tai nạn xảy ra trên đường.

Tài liệu tham khảo

[1] Minh Vũ, “Điểm 'mù' chết người của xe container khi vào cua?”, 19 05 2016.

<https://vnexpress.net/diem-mu-chet-nguoi-cua-xe-container-khi-vao-cua-3405362.html?gidzl=NaDSE4TRYnHSMriwHqM85rbTMpXRIjz3JW86Q5eKqnH12L0vL4RL5XnR1Zu158L355XLR3CiBk4OG5sF40>

[2] Đinh Hải Lâm, Đoàn Dương Quý, “Báo cáo nghiên cứu, thiết kế, chế tạo hệ thống xác định khoảng cách giữa ô tô với chướng ngại vật”, Biên Hòa, 06 2010

[3] Damien Dooley; Brian McGinley; Ciarán Hughes; Liam Kilmartin; Edward Jones; Martin Glavin, A Blind-Zone Detection Method Using a Rear-Mounted Fisheye Camera with Combination of Vehicle Detection Methods, IEEE, 2016.

[4] Nicole Wakelin, 7 Cars with a Blind Spot Camera, 17 03 2022.

[5] “Pixel là gì? Megapixel là gì?”, <https://www.thichcongnghe.com/2014/10/pixel-la-gi-megapixel-la-gi.html>

[6] “What is Pixel Pitch and Why Does It Matter?”, <https://www.planar.com/blog/2018/2/23/what-is-pixel-pitch-and-why-does-it-matter/>

[7] “Dot pitch”, <https://www.techopedia.com/definition/2697/dot-pitch>

[8] “Độ phân giải ảnh”, <https://binhminhdigital.com/tin/nhung-dieu-can-biet-ve-do-phan-giai-hinh-anh.html>

[9] “Độ sáng”, <https://nttuan8.com/bai-5-gioi-thieu-ve-xu-ly-anh/>

[10] “Color space (không gian màu) là gì?”, <https://quantrimang.com/cong-nghe/color-space-la-gi-182975>

[11] “Thị giác máy tính”, <https://medium.com/m/global-identity?redirectUrl=https%3A%2F%2Ftowardsdatascience.com%2Feverything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e>

[12] “Các bước xử lý ảnh”, http://tailieudientu.lrc.tnu.edu.vn/Upload/Collection/brief/brief_29212_32625_265201210184573.pdf

[13] “Mô hình vector và Raster”, <https://anhvientham.com/du-lieu-vector-va-raster-phan-tich-va-so-sanh-co-diem-gi-khac-nhau/>

[14] “Phương pháp Haar Cascade”, <https://www.analyticsvidhya.com/blog/2022/04/object-detection-using-haar-cascade-opencv/#:~:text=What%20are%20Haar%20Cascades%3F,%2C%20buildings%2C%20fruits%2C%20etc.>

[15] “Thư viện OpenCV”, <https://topdev.vn/blog/opencv-la-gi-hoc-computer-vision-khong-kho/>

[16] “Thư viện Matplotlib”, <https://viblo.asia/p/gioi-thieu-ve-matplotlib-mot-thu-vien-rat-huu-ich-cua-python-dung-de-ve-do-thi-yMnKMN6gZ7P>

[17] “Tensorflow”, <https://topdev.vn/blog/tensorflow-la-gi/>

Phụ lục

Phụ lục 1: Chương trình python

```
import cv2
import time
from RPi import GPIO

cascade_src = 'cars1.xml'
#Line a
ax1=0
ay=180
ax2=640
#Line b
bx1=0
by=300
bx2=640
#Car numi
= 1
t1=0
z=0
def Speed_Cal(time1):
#Here i converted m to Km and second to hour then division to reach Speed in this
form (KM/H)
try:
Speed = (10*3600)/(time1*1000) + 30
print ("Car Number "+str(i)+" Speed: "+str (Speed)+" KM/H")if
Speed > 60:
#Print("Trigger")
GPIO.output(7,1)      t1=
time.time()
while (time.time() - t1)<1:pass
GPIO.output(7,0)      except
ZeroDivisionError:
print (5)

start_time = time.time()
#Video ....
dispW=640
dispH=480 flip=2
#xoay cam
camSet='tcpclientsrc host=172.20.10.2 port=8554! gdpdepay! rtph264depay!
h264parse! nvv4l2decoder! nvvidconv flip-method='+str(flip)+'! video/x-
```

```

raw,format=BGRx ! videoconvert ! video/x-raw, width='+str(displW)+' ,
height='+str(displH)+' ,format=BGR ! appsink drop=true sync=false '
cam=cv2.VideoCapture(camSet)
car_cascade = cv2.CascadeClassifier(cascade_src)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7, GPIO.OUT)

while True:
    ret, img = cam.read()
    if (type(img) == type (None)):break
    blurred = cv2.blur(img,ksize=(15,15))
    gray = cv2.cvtColor(blurred, cv2.COLOR_BGR2GRAY)
    cars = car_cascade.detectMultiScale(gray, 1.1, 2)

    cv2.line(img, (ax1, ay), (ax2, ay), (255,255,0),2)
    cv2.line(img, (bx1, by), (bx2, by), (255,0,0),2)

    for (x,y,w,h) in cars:
        cv2.rectangle(img, (x,y),(x+w,y+h),(0,0,255),2)
        cv2.circle(img, (int((x+x+w)/2), int((y+y+h)/2)),1, (0,255,0),-1)

    while int(ay-10) <= int((y+y+h)/2) and int(ay+10) >= int((y+y+h)/2):
        start_time = time.time()
        z=1
        print ("1")break

    while int(ay) <= int((y+y+h)/2):print
    ("2")
    if int(by-10) <= int((y+y+h)/2) and int(by+10) >= int((y+y+h)/2) and z==1:print
    ("3")
    Speed_Cal(time.time() - start_time)z=0
    break cv2.imshow('video',
    img)

    if cv2.waitKey(5) == ord('q'):break
    cam.release()
    cv2.destroyAllWindows()

```

Phụ lục 2: Sơ lược về hệ điều hành, cách thiết lập và các thư viện

- Hệ điều hành Ubuntu:

Ubuntu là một hệ điều hành mã nguồn mở với ưu điểm miễn phí, cung cấp đủ chương trình cần thiết, hệ điều hành được cập nhật tự động mỗi 6 tháng. Giao diện của Ubuntu đơn giản.

Các thiết bị cần chuẩn bị để cài hệ điều hành:

- + 1 Board Jetson Nano Developer Kit
- + 1 Thẻ nhớ MicroSD Class 10 dung lượng tối thiểu 16GB
- + Đầu đọc thẻ nhớ
- + Màn hình HDMI, cáp HDMI, bàn phím, chuột, dây LAN
- + Nguồn 5V-2A hoặc 5V-5A
- + USB wifi hoặc dây ethernet

- Ngôn ngữ python dùng để viết code:

là ngôn ngữ lập trình phổ biến, được sử dụng rộng trong các ứng dụng web, phát triển phần mềm, khoa học dữ liệu và máy học (ML).

Ngôn ngữ Python ra đời với ưu điểm đơn giản, dễ hiểu, dễ đọc và có thể chạy trên nhiều nền tảng khác nhau:

- + Ngôn ngữ Python gần giống với tiếng anh giúp người sử dụng dễ dàng đọc và hiểu một chương trình Python.
- + Câu lệnh ngắn gọn giúp cải thiện năng suất của người lập trình.
- + Python có một thư viện tiêu chuẩn lớn, chứa nhiều dòng mã có thể tái sử dụng cho hầu hết mọi tác vụ. Vì vậy, người dùng không cần phải viết mã từ đầu.
- + Dễ dàng sử dụng Python với các ngôn ngữ lập trình phổ biến khác như Java, C và C++.
- + Cộng đồng người sử dụng Python lớn rộng và nhiệt tình hỗ trợ trên toàn thế giới. Người dùng có thể nhận được sự hỗ trợ nhanh chóng.
- + Tài liệu học Python có sẵn trên các nguồn trên Internet bao gồm video và tài liệu, thuận tiện cho người mới học cũng như những người cần tài liệu.
- + Python có thể được sử dụng trên nhiều hệ điều hành máy tính khác nhau, chẳng hạn như Windows, macOS, Linux và Unix.

- Thư viện OpenCV dùng để xử lý ảnh [15]:

OpenCV (Open-Source Computer Vision Library) là thư viện nguồn mở hàng đầu cho Computer Vision và Machine Learning. Thư viện này giống như kho lưu trữ các mã nguồn mở được dùng để xử lý hình ảnh, phát triển các ứng dụng đồ họa trong thời gian thực.

OpenCV cung cấp một số lượng lớn các mã xử lý phục vụ cho quy trình của thị giác máy tính hay các learning machine khác. Đồng thời, OpenCV giúp cải thiện tốc độ của GPU khi thực hiện các hoạt động real time.

Cài đặt thư viện Numpy: mở Command Prompt và gõ lệnh: `pip install opencv-python`

Khai báo thư viện Numpy: `import cv2`

- Thư viện Numpy dùng để làm việc với ma trận:

Numpy (Numeric Python) là một thư viện toán học của Python. Nó cho phép làm việc hiệu quả với ma trận và mảng, tốc độ xử lý dữ liệu ma trận và mảng lớn nhanh hơn nhiều lần khi chỉ sử dụng “core Python” đơn thuần.

Cài đặt thư viện Numpy: mở Command Prompt và gõ lệnh: `pip install numpy`

Khai báo thư viện Numpy: `import numpy as np`

- Thư viện Matplotlib [16]:

Matplotlib là một thư viện vẽ đồ thị cho Python. Nó được sử dụng cùng với NumPy để cung cấp môi trường thay thế nguồn mở hiệu quả cho MatLab. Nó cũng có thể được sử dụng với các bộ công cụ đồ họa như PyQt và wxPython.

Một Matplotlib figure có thể được phân loại thành nhiều phần như dưới đây:

+ Figure: Như một cái cửa sổ chứa tất cả những gì bạn sẽ vẽ trên đó.

+ Axes: Thành phần chính của một figure là các axes (những khung nhỏ hơn để vẽ hình lên đó). Một figure có thể chứa một hoặc nhiều axes. Nói cách khác, figure chỉ là khung chứa, chính các axes mới thật sự là nơi các hình vẽ được vẽ lên.

+ Axis: Chúng là dòng số giống như các đối tượng và đảm nhiệm việc tạo các giới hạn biểu đồ.

+ Artist: Mọi thứ mà bạn có thể nhìn thấy trên figure là một artist như Text objects, Line2D objects, collection objects. Hầu hết các Artists được gắn với Axes.

Cài đặt thư viện Numpy: mở Command Prompt và gõ lệnh: `pip install matplotlib`

Khai báo thư viện Numpy:

```
import numpy as np
```

```
from matplotlib import pyplot as plt
```

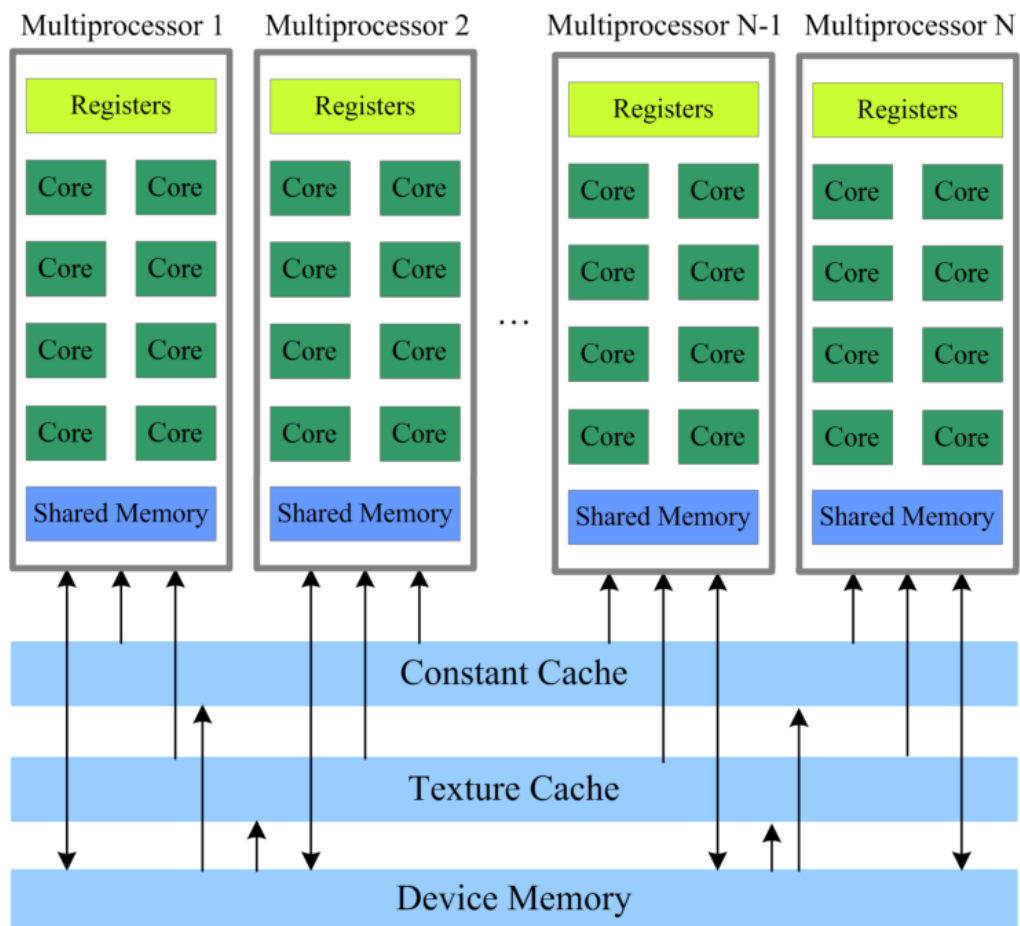
- Nền tảng Anaconda Python:

Nền tảng khoa học dữ liệu Python nguồn mở được sử dụng rộng rãi nhất ở thời điểm hiện tại là Anaconda. Với hơn 11 triệu người dùng, Anaconda Sử dụng Anaconda là phương pháp nhanh nhất và đơn giản nhất để nghiên cứu khoa học dữ liệu, quản trị các truy vấn nguồn mở, môi trường điều khiển và phân phối Python và R trên Windows, Linux và Mac OS X. Ưu điểm của anaconda

- + Tải xuống gói 1500+ dễ dàng cho Python / R để nghiên cứu dữ liệu.
- + Dễ dàng quản lý thư viện, môi trường và sự phụ thuộc liên thư viện.
- + Scikit-learning, Tensorflow và Keras giúp việc tạo các mô hình học máy và học sâu trở nên đơn giản.
- + Xử lý dữ liệu nhanh chóng bằng cách sử dụng cấu trúc và numpy.
- + Sử dụng Matplotlib và Bokeh để hiển thị kết quả.

- Cuda:

NVIDIA đã tạo ra kiến trúc tính toán song song được gọi là CUDA (Compute Unified Device Architecture). Công cụ tính toán của GPU (Đơn vị xử lý đồ họa) của NVIDIA được gọi là CUDA và các lập trình viên có thể truy cập nó bằng các ngôn ngữ lập trình được sử dụng rộng rãi. Các thuật toán cho GPU được các lập trình viên thực hiện bằng trình biên dịch PathScale Open64 C và ngôn ngữ C cho CUDA. Ngôn ngữ lập trình C được hỗ trợ bởi kiến trúc CUDA cho tất cả các hoạt động tính toán. Ngoài ra, hỗ trợ CUDA đã được phát triển bởi các bên thứ ba cho Python, Fortran, Java và MATLAB.



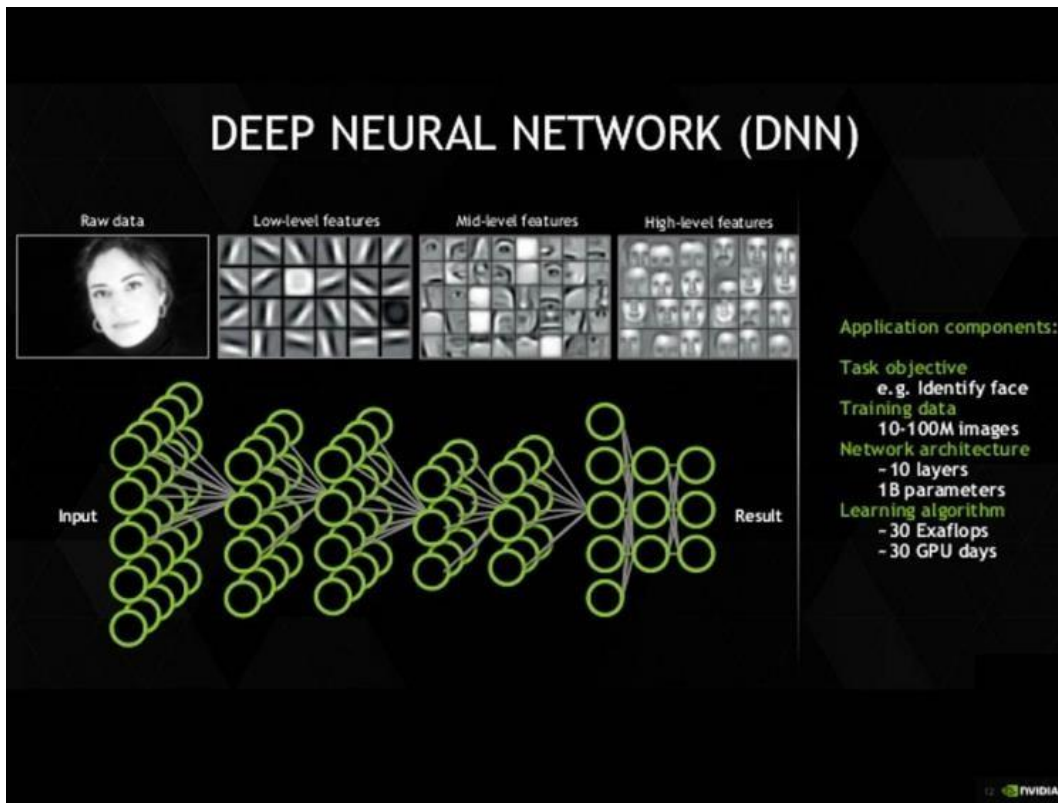
Quy trình xử lý của Cuda

Các nhà phát triển có thể truy cập đơn vị bộ xử lý đồ họa của tập lệnh ảo CUDA và bộ nhớ ảo (GPU CUDA). Các GPU NVIDIA gần đây nhất có khả năng thực hiện các phép tính tương tự như CPU một cách dễ dàng nhờ CUDA. Nhưng không giống như CPU, GPU có kiến trúc song song hoàn toàn tập trung vào khả năng xử lý nhiều luồng dữ liệu chậm cùng một lúc thay vì chỉ một luồng nhanh chóng. GPGPU là một phương pháp có mục đích chung để giải quyết vấn đề dựa trên GPU.

Mục đích của việc thiết lập môi trường Cuda là cho phép card đồ họa xử lý hình ảnh, tốc độ này nhanh hơn nhiều so với CPU.

- CuDNN:

Thư viện Mạng thần kinh sâu NVIDIA CUDA®, còn được gọi là CuDNN, là một thư viện cơ bản giúp tăng tốc GPU cho các mạng thần kinh sâu. Đối với mạng nơ-ron, CuDNN cung cấp các điều chỉnh đáng kể đối với các quy trình phổ biến như gộp, chuẩn hóa, chuẩn hóa và kích hoạt lớp.

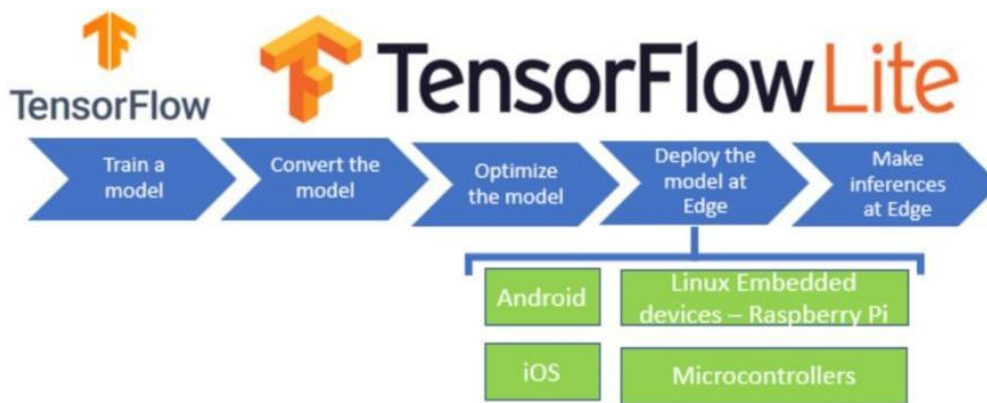


Cuda Deep Neural Network

Trên khắp thế giới, cuDNN thường được sử dụng bởi các nhà nghiên cứu học sâu và các nhà phát triển khung công tác để tăng hiệu suất GPU. Thay vì dành thời gian điều chỉnh hiệu suất GPU cấp thấp, họ có thể tập trung vào việc xây dựng các ứng dụng phần mềm và đào tạo mạng nơ-ron. Một số khung học sâu phổ biến, bao gồm Caffe, Caffe2, Chainer, Keras, MATLAB, MxNet, TensorFlow và PyTorch, được cuDNN tạo ra nhanh hơn. Điều quan trọng là phải kết hợp cuDNN vào các khung để sử dụng các khung học sâu được tối ưu hóa của NVIDIA.

- Tensorflow [17]:

Google đã phát triển TensorFlow, còn được gọi là Google TensorFlow, là một thư viện phần mềm miễn phí tập trung vào học máy. TensorFlow ban đầu được tạo ra bởi các kỹ sư và nhà nghiên cứu của Google Brain Team, chủ yếu để sử dụng nội bộ. TensorFlow ban đầu được cung cấp như một phần của giấy phép mã nguồn mở Apache 2.0. Google hiện đang sử dụng TensorFlow cho cả nghiên cứu và sản xuất, và nó được coi là ứng dụng thay thế nguồn đông DistBelief. Dự án học sâu tích hợp sâu đầu tiên được gọi là TensorFlow.



Quy trình xử lý Tensorflow

TensorFlow lấy tên của nó từ các mảng đa chiều được gọi là tenxơ, được sử dụng bởi các mạng thần kinh cho các hoạt động khác nhau. Theo Google, so với DistBelief, TensorFlow nhanh hơn, thông minh hơn và linh hoạt hơn và có thể dễ dàng thích ứng với các khu vực và sản phẩm mới. Nó chủ yếu được tạo ra để nghiên cứu mạng lưới thần kinh sâu sắc và để tạo điều kiện cho việc học máy, mặc dù TensorFlow đã được sử dụng trong một loạt các lĩnh vực khác.

Là một phần của việc học, TensorFlow sắp xếp thông qua các lớp dữ liệu (còn được gọi là các nút). Hệ thống chỉ định các đặc điểm cơ bản của đối tượng trong lớp đầu tiên. Nó tìm kiếm thông tin sắc thái hơn về vấn đề khi các chuyển động sâu hơn diễn ra. Hình ảnh được sắp xếp nhanh hơn, cho phép người tiêu dùng truy cập vào dữ liệu sâu sắc hơn. TensorFlow có thể được sử dụng với nhiều hệ điều hành khác nhau, bao gồm Linux, Windows, MacOS và các nền tảng di động như iOS và Android. Chạy trên một số CPU và GPU là một trong những khả năng độc đáo của TensorFlow. TensorFlow báo cáo các phép tính dưới dạng biểu đồ luồng dữ liệu trạng thái. Hiện tại, hơn 6.000 kho lưu trữ trực tuyến miễn phí sử dụng TensorFlow.

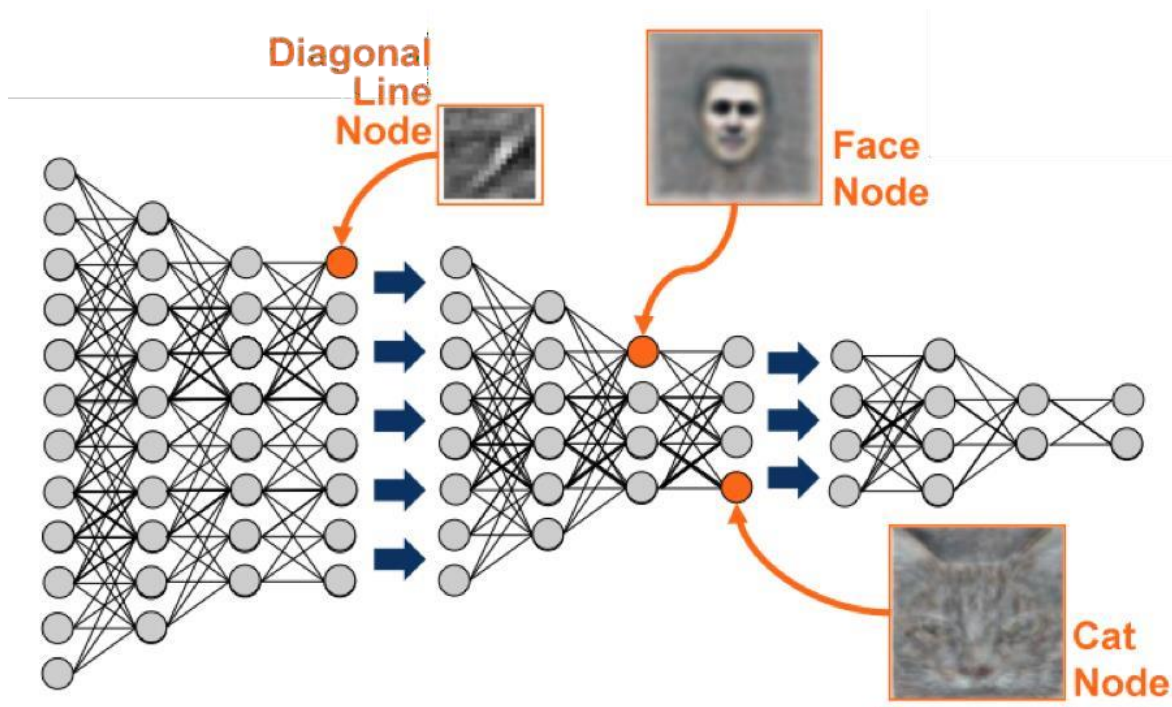
Windows, MacOS và các nền tảng di động như iOS và Android. Chạy trên một số CPU và GPU là một trong những khả năng độc đáo của TensorFlow. TensorFlow báo cáo các phép tính dưới dạng biểu đồ luồng dữ liệu trạng thái. Hiện tại, hơn 6.000 kho lưu trữ trực tuyến miễn phí sử dụng TensorFlow.

Ứng dụng của TensorFlow: Mặc dù TensorFlow có nhiều ứng dụng, nhưng có năm ứng dụng chính:

+ Nhận dạng giọng nói/âm thanh: Nhận biết và phân loại các tiếng ồn khác nhau. Điều này bao gồm phát hiện lỗi hỏng, phân tích tình cảm, tìm kiếm giọng nói và nhận dạng giọng nói.

+ Các chương trình dựa trên văn bản: phân tích Cảm xúc, xác định mối đe dọa và phát hiện gian lận là ba ứng dụng chính của TensorFlow. Tóm tắt văn bản, Trả lời tự động, và nhận dạng ngôn ngữ và dịch là những công dụng khác của nó.

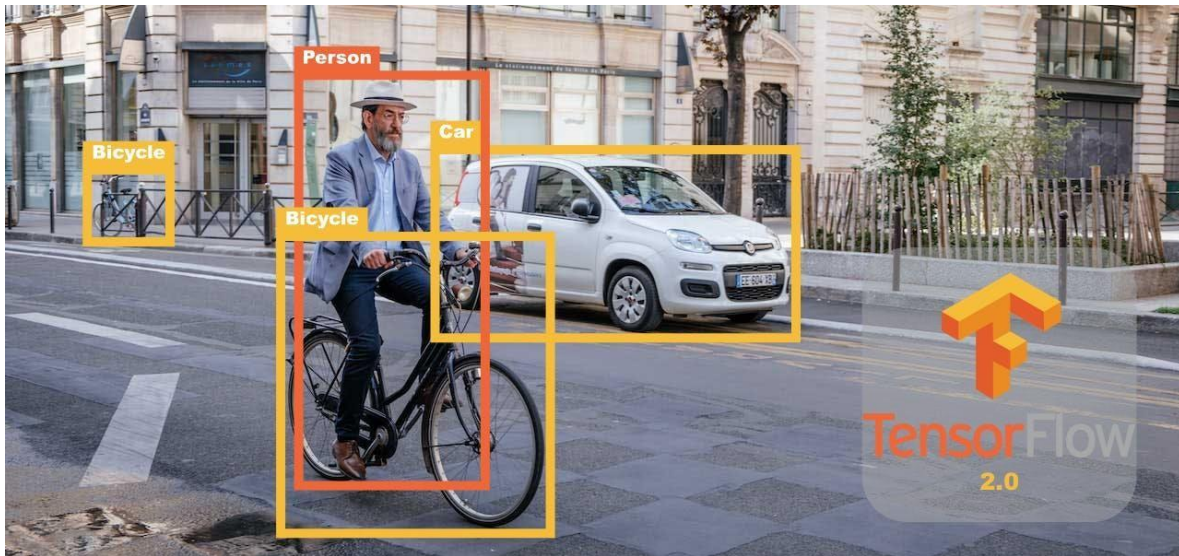
+ Nhận dạng hình ảnh là khả năng nhận dạng và phân loại hình ảnh, cũng như nắm bắt nội dung và ngữ cảnh của hình ảnh cũng như nhận biết và xác định người và sự vật trong đó. Nó cũng có thể được sử dụng để xác định các mẫu và xác định các biểu mẫu cho mục đích mô hình hóa.



Cách thức nhận diện khuôn mặt trong TensorFlow

+ Chuỗi thời gian: Phân tích dữ liệu chuỗi thời gian để trích xuất số liệu thống kê có ý nghĩa, cũng như dự báo các khoảng thời gian không cụ thể và để tạo các phiên bản thay thế của chuỗi thời gian. Trường hợp sử dụng phổ biến nhất cho Chuỗi thời gian là Khuyến nghị.

+ Phát hiện video: để xác định, phân tích và phân loại video. Việc triển khai chính là trong Phát hiện chuyển động, Phát hiện luồng thời gian thực trong các trường Trò chơi, Bảo mật, Sân bay và UX / UI.



Nhận diện hình ảnh bằng TensorFlow

Lợi ích của Tensorflow:

Tensorflow rất được yêu thích vì nó cung cấp một số lợi ích so với các công cụ học máy khác trên thị trường. Trong số những lợi thế này là:

- **Đồ thị:** Tensorflow cung cấp khả năng hiển thị đồ thị tính toán được cải thiện, đây là một trong những ưu điểm chính của nó. Chúng thực sự làm cho việc theo dõi các hoạt động và quan sát luồng dữ liệu trong một bức tranh trở nên đơn giản hơn.
- **Quản lý thư viện:** Tensorflow có thư viện lớn hơn đáng kể và cơ chế quản lý thư viện tốt hơn do Google hỗ trợ. Ngoài ra, nó cung cấp các bản cập nhật nhanh chóng, các bản phát hành mới nhất quán và các bổ sung tính năng mới.
- **Tính linh hoạt:** Tensorflow rất linh hoạt theo cách nó cho phép hiển thị dữ liệu. Nó cung cấp đồng thời nhiều mô hình hoặc phiên bản của cùng một mô hình để sử dụng dữ liệu theo bất kỳ cách nào phù hợp nhất. Bạn cũng có thể sử dụng một số phần riêng lẻ hoặc có thể sử dụng tất cả các phần với nhau. TensorFlow cũng tạo điều kiện cho việc di chuyển không tự động sang các mô hình hay phiên bản mới.
- **Khả năng mở rộng:** TensorFlow chạy GPU, CPU, máy tính để bàn, máy chủ và nền tảng điện toán di động, do đó giúp chạy dễ dàng hơn trên mọi loại thiết bị, cho dù đó là máy tính để bàn, máy tính xách tay hay thậm chí trên điện thoại thông minh.
- **Tự động phân biệt.** TensorFlow tự động phân biệt đối xử. Ưu điểm của việc này là TensorFlow sẽ tự động tuân theo các thuật toán học máy dựa trên độ dốc khi chúng trộn hàm mục tiêu và dữ liệu với kiến trúc tính toán của mô hình dự đoán.

- Hiệu suất: Khi so sánh với các ứng dụng trước đây, TensorFlow đã nâng cao đáng kể hiệu suất và chức năng bằng cách bao gồm hỗ trợ phức tạp cho các rãnh và tính toán không đồng bộ. Ngoài ra, có một số ngôn ngữ có sẵn để tính toán đồ thị.
- Có thể phân nhánh phụ của đồ thị với Tensorflow cho mục đích gỡ lỗi. Kết quả là, một phương pháp sửa lỗi tuyệt vời cho phép đưa và truy xuất dữ liệu rời rạc vào một cạnh được tạo ra.

Hạn chế của Tensorflow:

- Overkill: Hạn chế chính của Tensorflow là nó quá mạnh. Mặc dù nó là tuyệt vời cho học sâu, đặc biệt là với thư viện mạnh mẽ của nó, nó thường được coi là quá mức cho các nhiệm vụ đơn giản.
- Không hỗ trợ Windows: TensorFlow không cung cấp hỗ trợ Windows hoặc ít nhất là hỗ trợ Windows rất hạn chế. Nó hỗ trợ chính cho Linux, điều mà không phải ai cũng có thể thoải mái. Tuy nhiên, người ta có thể cài đặt nó trong môi trường conda hoặc sử dụng thư viện gói python, pip, để sử dụng nó trong Windows.
- Tốc độ tính toán: khi so sánh với một số đối thủ khác trên thị trường. TensorFlow thua một chút về tốc độ và cách sử dụng, mặc dù sự khác biệt không nhiều. Hỗ trợ GPU hạn chế - hiện tại TensorFlow chỉ hỗ trợ các GPU của NVIDIA và hỗ trợ ngôn ngữ đầy đủ duy nhất là của Python, điều mà không phải ai cũng có thể cảm thấy thoải mái. Nó bỏ qua các ngôn ngữ khác trong học tập sâu.
- Bất chấp những hạn chế này, Tensorflow chắc chắn là một trong những chương trình tốt nhất cho các lập trình viên làm việc trên máy học. Các bản cập nhật trong tương lai có thể giải quyết những hạn chế này. Ngoài ra, hãy nhớ rằng học máy liên tục phát triển trên tất cả các ngành, bao gồm cả các lĩnh vực kỹ thuật phi truyền thống.

Phụ lục 3: Sơ lược về thiết bị

- Jetson Nano Developer kit:

Jetson Nano là 1 board máy tính nhúng do hãng NVIDIA sản xuất với cấu hình khá mạnh bởi vì nó được trang bị GPU 128-core NVIDIA Maxwell. Khả năng tính toán của Jetson mạnh cùng với RAM 4G giúp nó có thể thực hiện được nhiều tác vụ hơn vùng lúc. Tuy nhiên, do việc tối giản lại hệ điều hành nên việc cài đặt một số package cũng khó khăn hơn.



Jetson Nano Developer kit

Các cổng của NVIDIA Jetson Nano Developer Kit: 4 cổng USB 3.0, cổng HDMI, M.2, DisplayPort và một cổng Ethernet.

Các chuẩn kết nối: SDIO, I2C, SPI và UART.

Giao thức MIPI-CSI để kết nối với máy ảnh. Người dùng cũng có thể kết nối mạng cho Jetson Nano bằng Wifi nếu không muốn sử dụng dây cáp Ethernet truyền thống.

Thông số kỹ thuật của jetson nano:

- + GPU: 128-core Maxwell™ GPU
- + CPU: quad-core ARM® Cortex®-A57 CPU
- + RAM: 4GB 64-bit LPDDR4
- + Bộ nhớ: khe cắm thẻ microSD (devkit) hoặc 16GB eMMC flash (Từ nhà sản xuất)

*Video:

- + Encode: 4K @ 30 (H.264/H.265)
- + Decode: 4K @ 60 (H.264/H.265)

*Các giao diện:

- + Ethernet: 10/100/1000BASE-T self-negotiation

+ Camera: 12-ch (3x4 OR 4x2) MIPI CSI-2 DPHY 1.1 (1.5Gbps)

+ Display: HDMI 2.0, DP (DisplayPort)

+ USB: 4x USB 3.0, USB 2.0 (Micro USB)

+ Others: GPIO, I2C, I2S, SPI, UART

*Nguồn cấp:

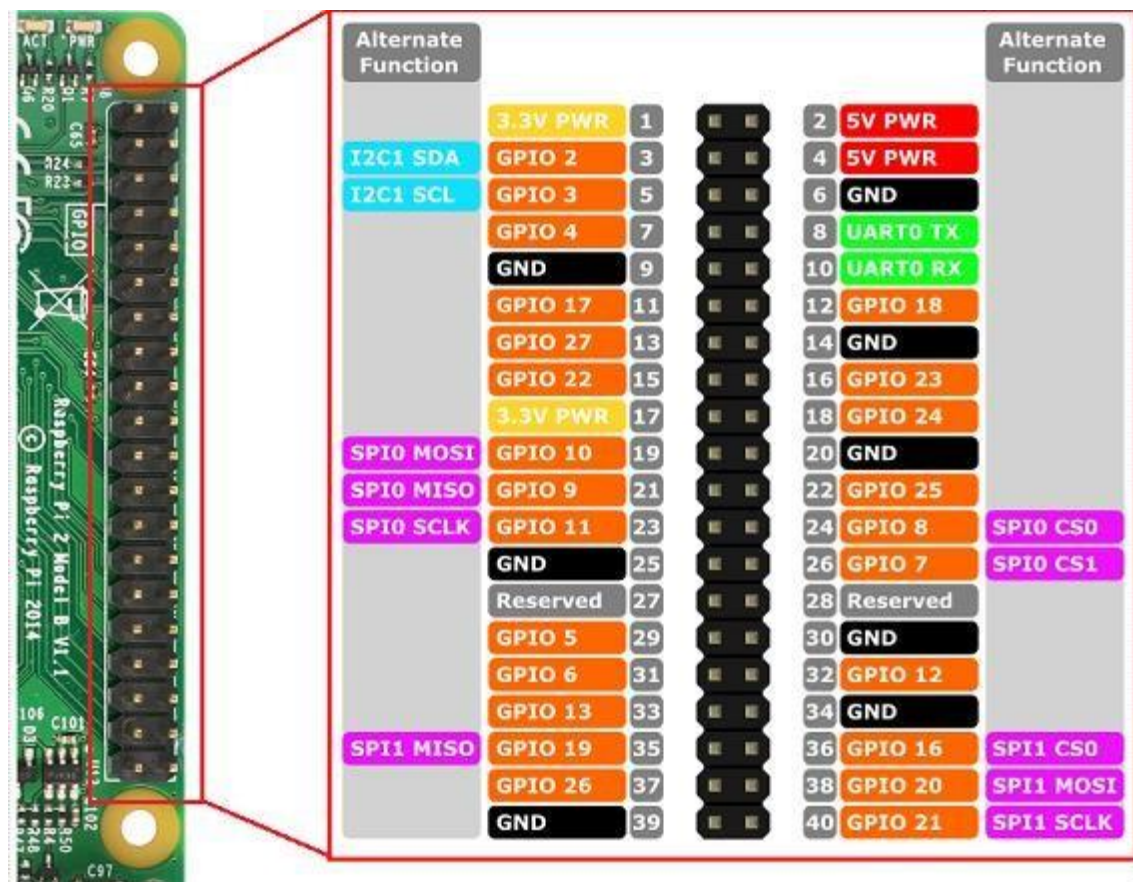
+ Micro USB (5V 2.5A)

+ DC jack (5V 4A)

*Kích thước:

+ Core module: 69.6 mm × 45 mm

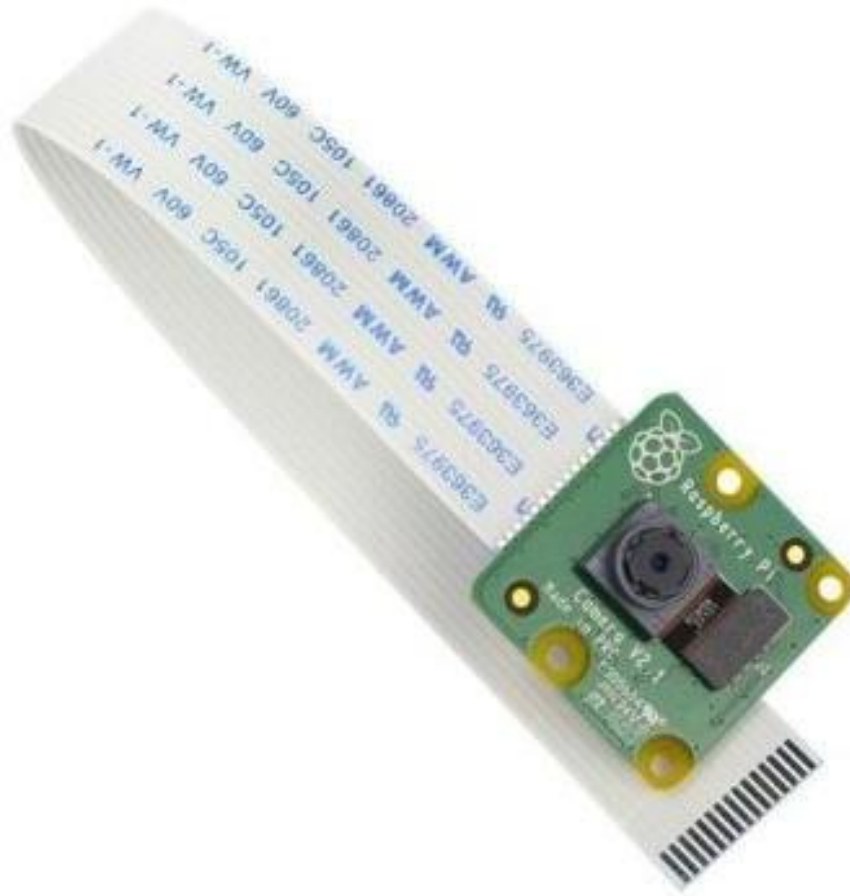
+ Whole kit: 100mm × 80mm × 29mm



Sơ đồ các chân kết nối của Jetson Nano Developer kit

- Lựa chọn camera:

Camera sử dụng trong đề án là Raspberry Pi Camera Module V2 8MP. Raspberry Pi Camera Module V2 có một cảm biến 8-megapixel của Sony IMX219 (so với cảm biến 5-megapixel OmniVision OV5647 trên Camera Module phiên bản cũ).. Mức giá hợp lý phù hợp với đề án.



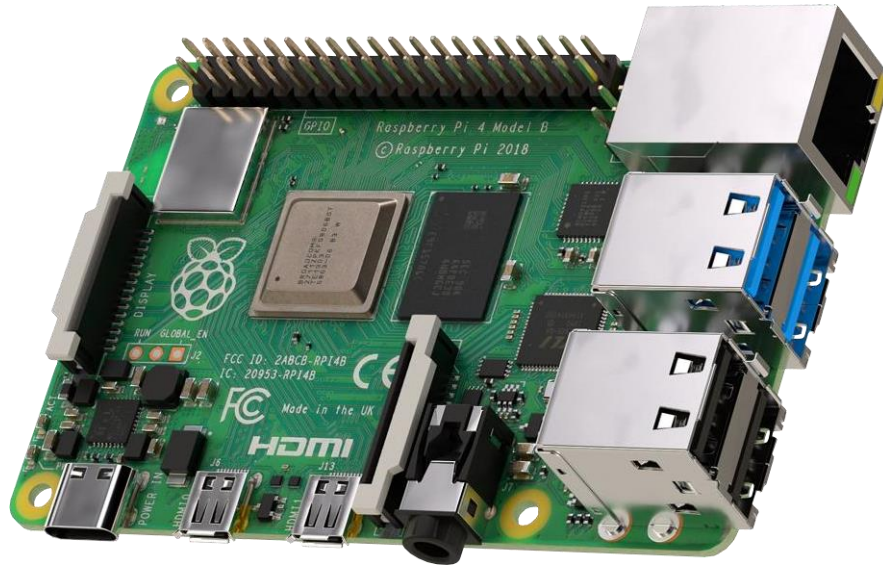
Raspberry Pi Camera Module V2 8MP

Thông số kỹ thuật:

- +Ổng kính tiêu cự cố định
- +Cảm biến độ phân giải 8-megapixel cho khả năng chụp ảnh kích thước 3280 x 2464
- +Hỗ trợ video 1080p30, 720p60 và 640x480p90
- +Kích thước 25mm x 23mm x 9mm
- +Trọng lượng chỉ hơn 3g
- +Kết nối với Raspberry Pi thông qua cáp ribbon đi kèm dài 15 cm
- +Camera Module được hỗ trợ với phiên bản mới nhất của Raspbian

- Raspberry pi 4 model B:

Để có thể giảm giá thành chúng em quyết định lựa chọn bộ vi điều khiển Raspberrypi 4 thay vì sử dụng thêm 1 chiếc Jetson Nano gắn với camera mà vẫn đảm bảo đầy đủ tính năng.



Phụ lục 4: Nguồn lấy Haar Cascade

[GitHub - andrewsobral/vehicle_detection_haarcascades: Vehicle Detection by HaarCascades with OpenCV](#)