



TÊN ĐỀ TÀI:
**HỆ THỐNG HỖ TRỢ ĐIỀU HƯỚNG CHO XE
TỰ HÀNH TRONG MÔI TRƯỜNG MÔ PHỎNG**

Nguyễn Trọng Nhân - 1914446
Lê Hoàng Minh Tú - 1915812
Hồ Hữu Trọng - 1915672

1. Hệ điều hành ROS

Hệ điều hành ROS



ROS (Robot operating system) là hệ điều hành meta chuyên dụng để lập trình và điều khiển robot. ROS có những ưu điểm nổi bật như là:

- ROS có mã nguồn mở, và hoàn toàn miễn phí
- ROS có thể được lập trình bằng nhiều ngôn ngữ khác nhau như C++, Python, ...

2. Gazebo Simulation

Gazebo Simulation



GAZEBO

2. Gazebo Simulation

Gazebo Simulation

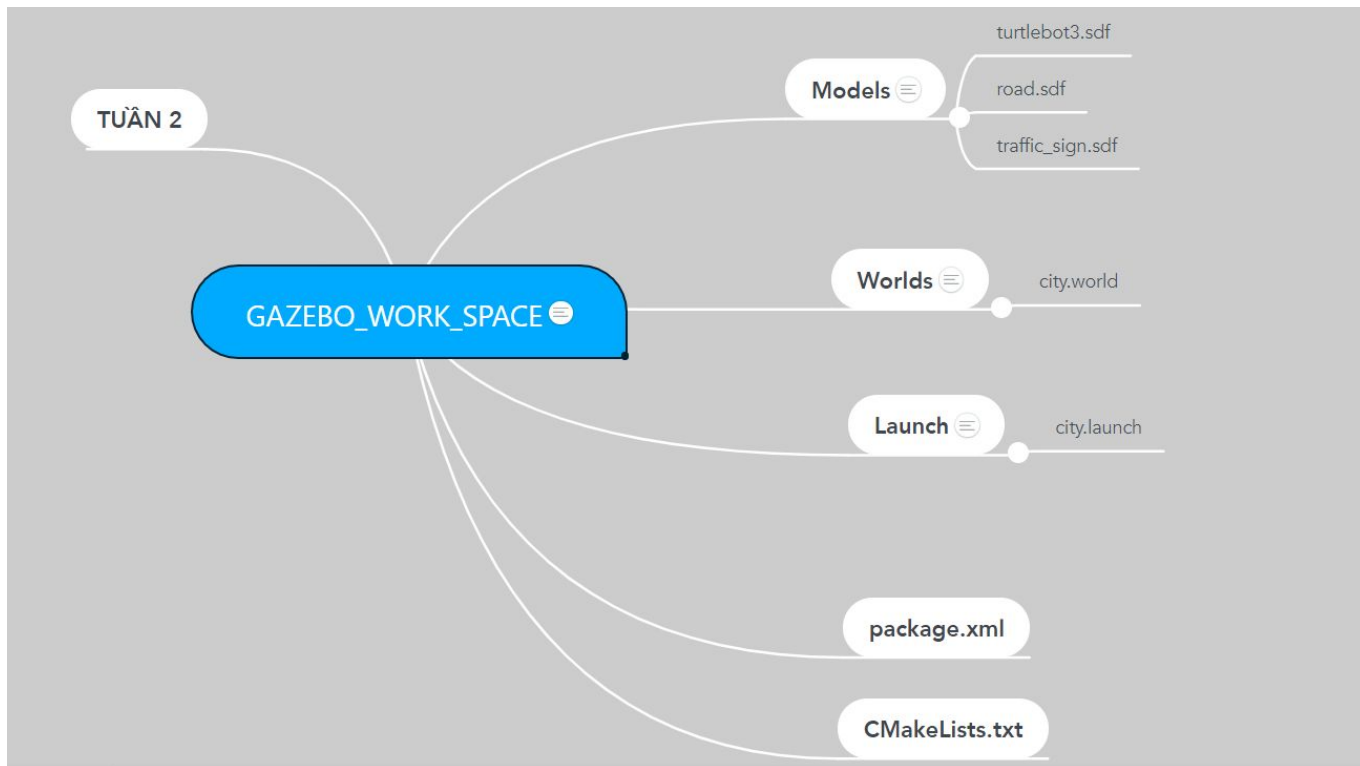


Gazebo là gì?

- Là công cụ mô phỏng 3D
- Được sử dụng cho Robot Design, kiểm thử các chương trình AI trước khi deploy ngoài thực tế, ...
- Được nhiều người ưa chuộng bởi:
 - Có GUI thân thiện, tương tác tốt
 - Là công cụ không tính phí
 - Có cộng đồng sử dụng lớn, và các ví dụ, model có sẵn ở mã nguồn mở

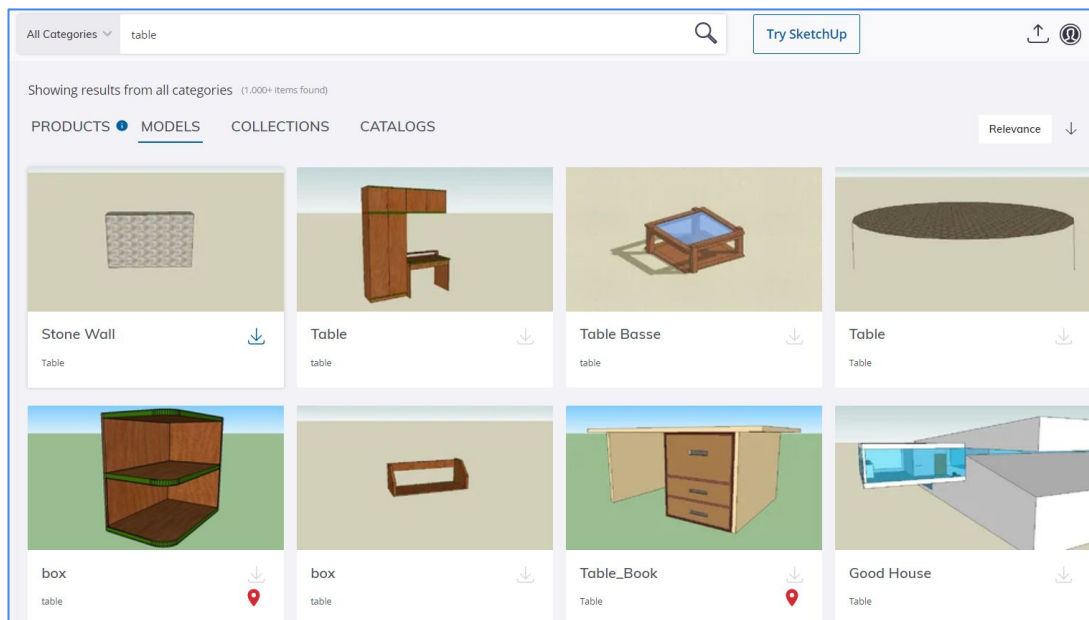
2. Gazebo Simulation

Gazebo Simulation - Cấu trúc không gian làm việc



2. Gazebo Simulation

Gazebo Simulation - Cấu trúc không gian làm việc - Models



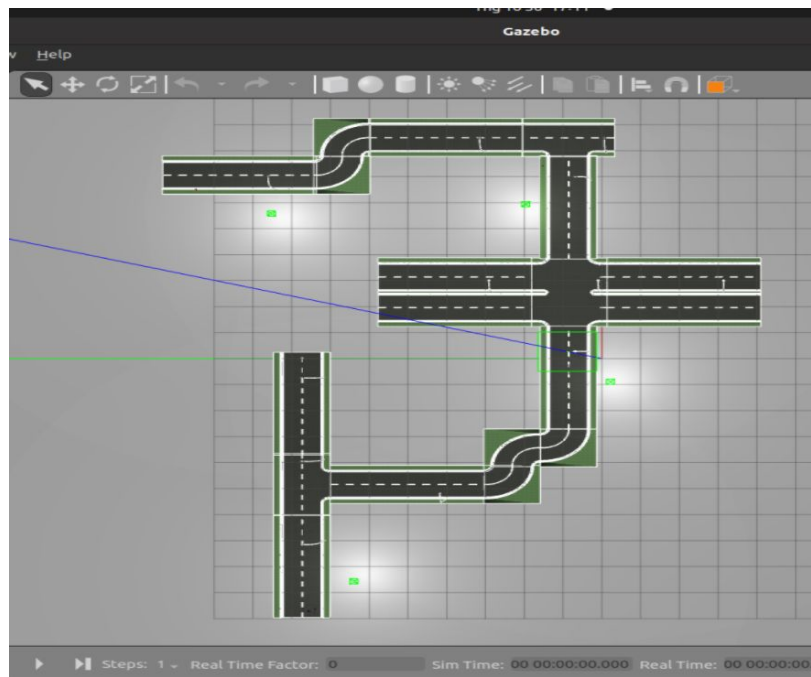
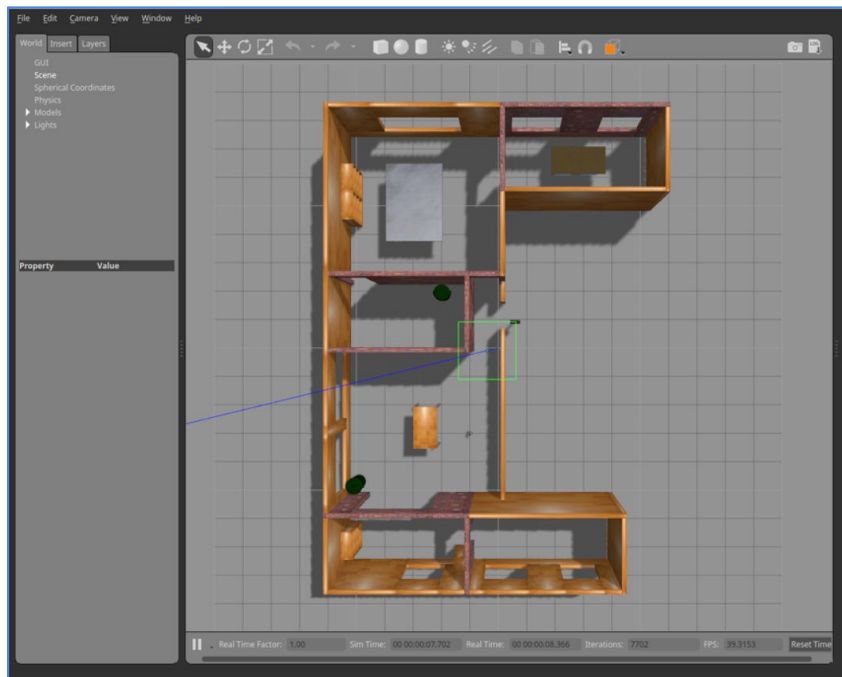
Model gồm các object được hiển thị bằng các file sau:

- `model.sdf`
 - Chỉ hiển thị khung cấu trúc của object (chỉ có 1 màu đen duy nhất)
- `model.config`
 - Chứa các mô tả, và tên model
- `model.dae`
 - Hiển thị phần texture (màu sắc của các bộ phận)

2. Gazebo Simulation

Gazebo Simulation - Cấu trúc không gian làm việc - Worlds

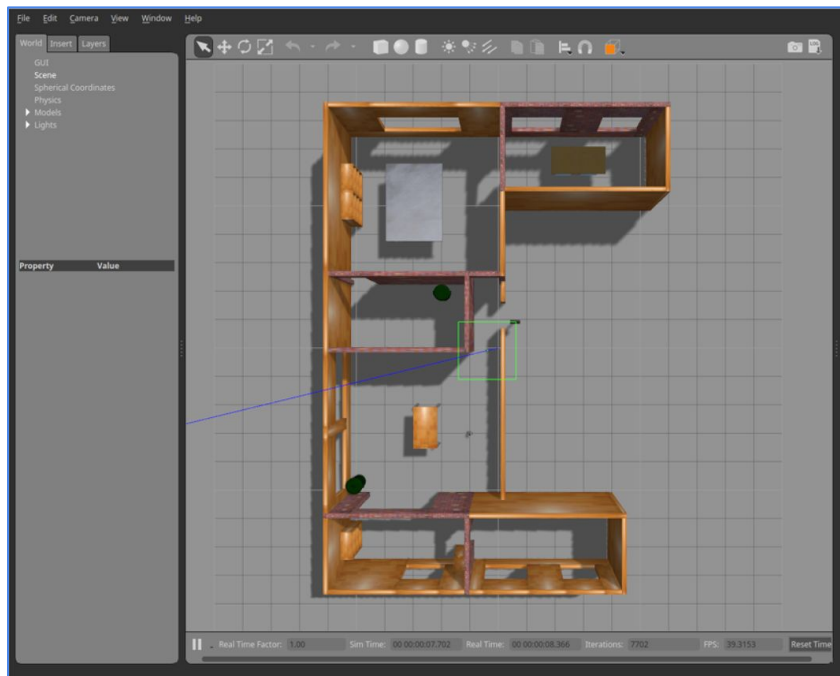
Worlds bao gồm các model đã build trước đó và có đuôi là .world.



2. Gazebo Simulation

Gazebo Simulation - Cấu trúc không gian làm việc - Launchs

Đối với các file `.launch`



- Ta sẽ include `.world` mà ta đã build
- Đồng thời cho phép quy định vị trí, và `TURTLEBOT_MODEL` ta spawn turtlebot trong map
- Ngoài ra, ta có thể chỉnh một số setting của gazebo (Vd: enable paused, enable gui, ...)

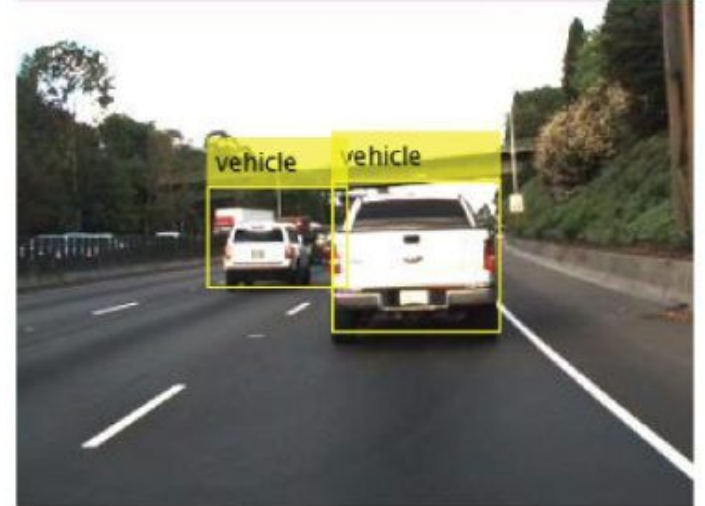
Note: Đối với file `.launch` ta có thể chạy bằng lệnh `roslaunch` của `ros`.

3. Sign Detection & Sign Classification

Sign Detection

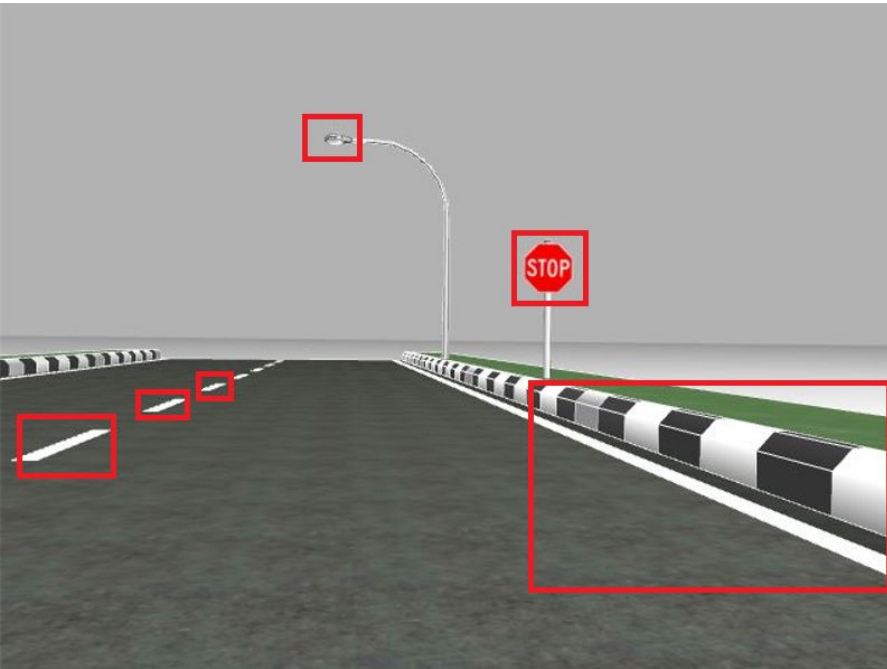


OBJECT DETECTION
ALGORITHM



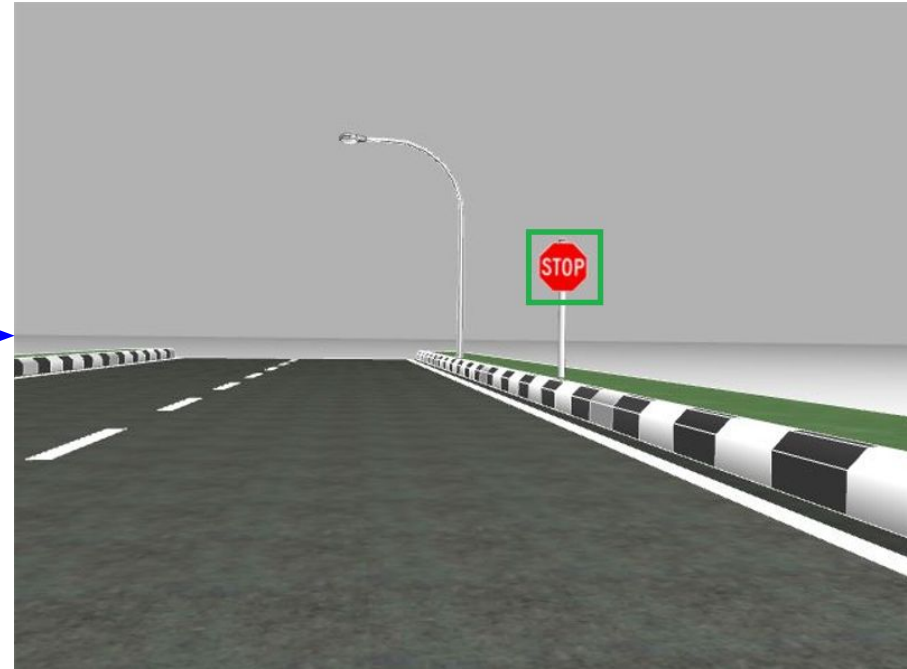
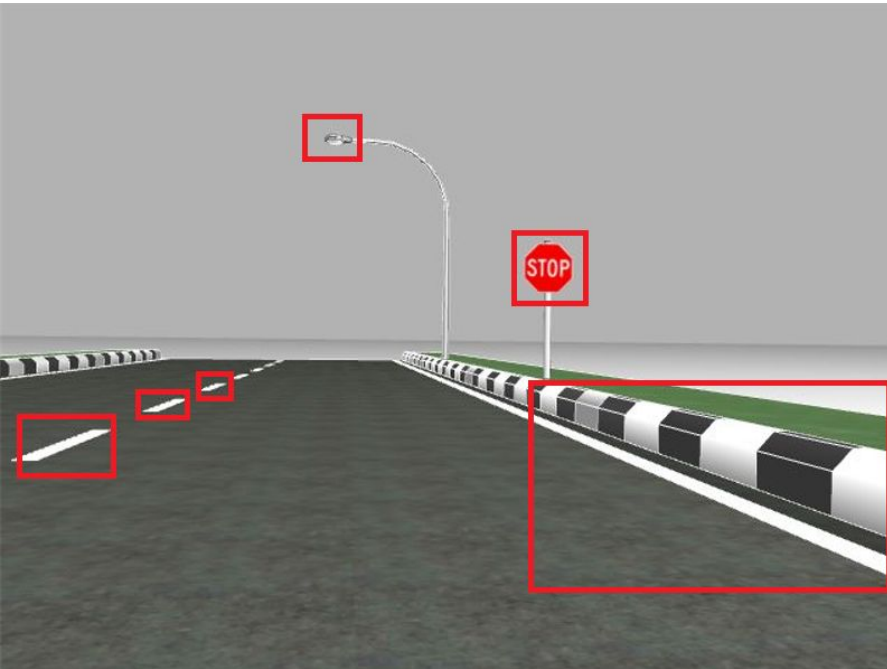
3. Sign Detection & Sign Classification

Sign Detection - Mục tiêu



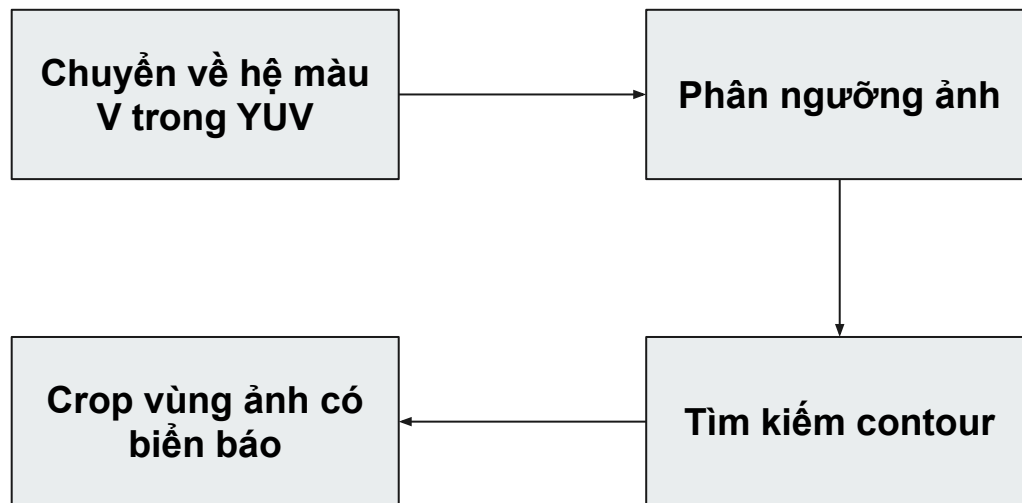
3. Sign Detection & Sign Classification

Sign Detection - Mục tiêu



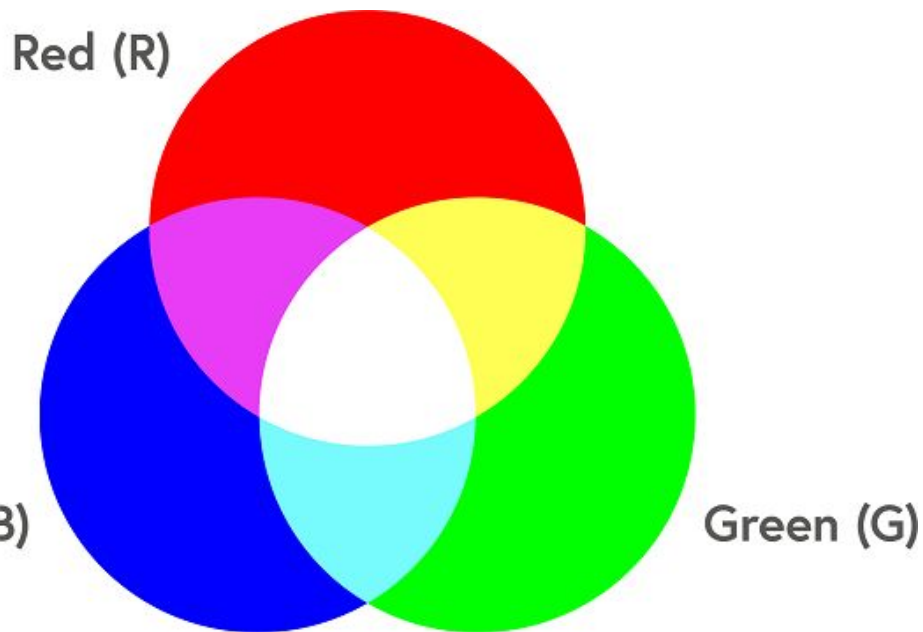
3. Sign Detection & Sign Classification

Sign Detection - Quy trình



3. Sign Detection & Sign Classification

Sign Detection - Chuyển về hệ màu V trong YUV



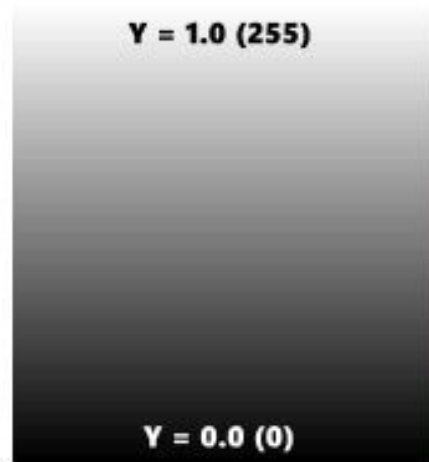
RGB Color

Hệ màu **RGB**, đây là hệ màu ta thường sử dụng:

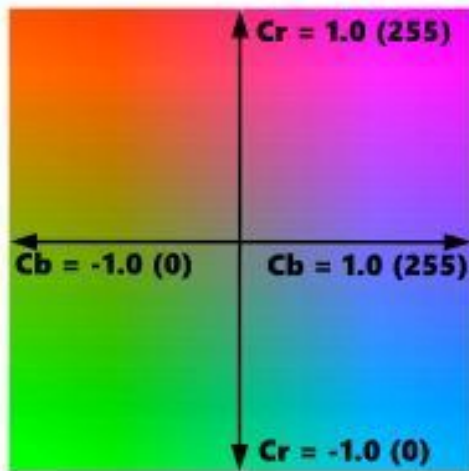
- R: giá trị **RED**
- G: giá trị **GREEN**
- B: giá trị **BLUE**

3. Sign Detection & Sign Classification

Sign Detection - Chuyển về hệ màu V trong YUV



Luminance



Chrominance (Y=0.5)

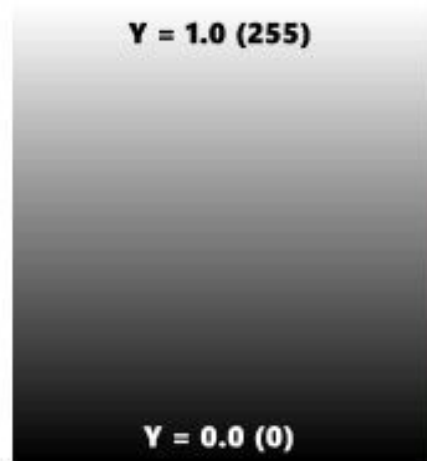
Hệ màu **YUV**:

- Y: Độ sáng (Luminosity)
- U và V: Sự khác nhau về màu sắc (Color Difference)

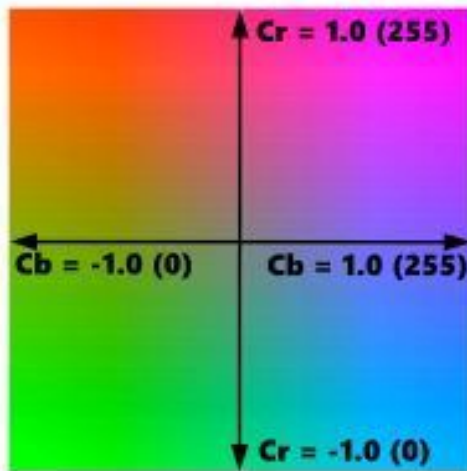
YUV Color

3. Sign Detection & Sign Classification

Sign Detection - Chuyển về hệ màu V trong YUV



Luminance



Chrominance ($Y=0.5$)

YUV Color

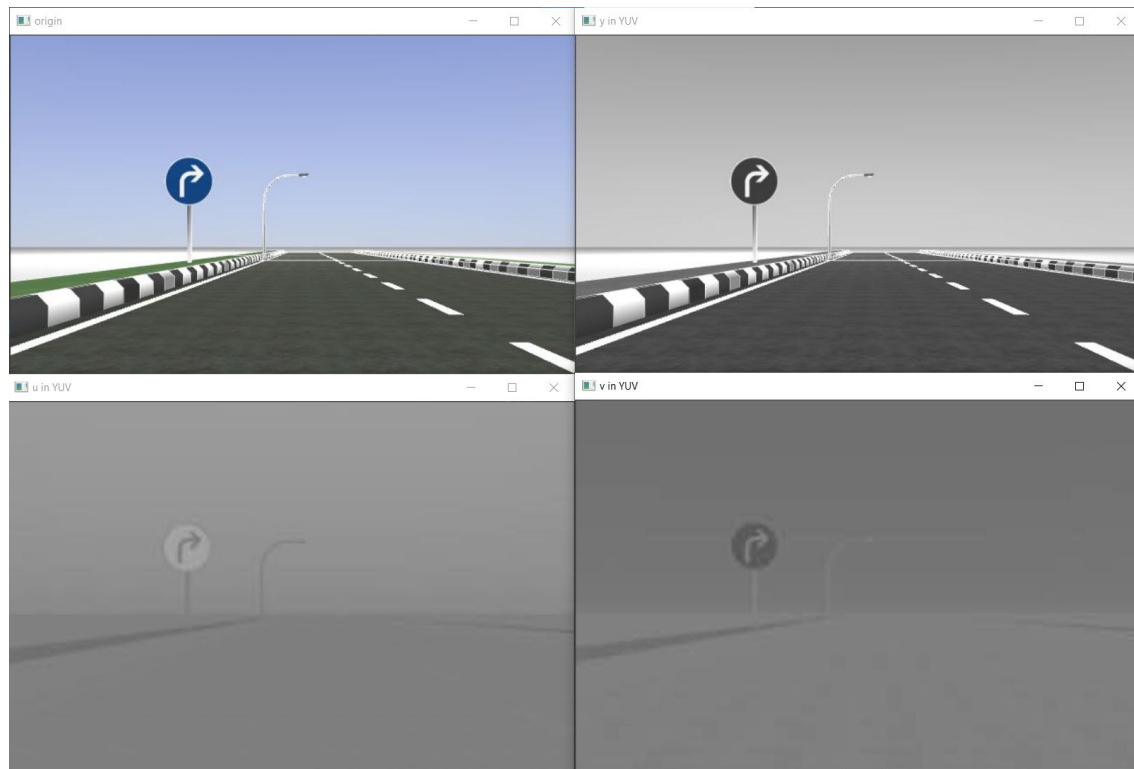
Hệ màu **YUV**:

- Y: Độ sáng (Luminosity)
- U và V: Sự khác nhau về màu sắc (Color Difference)

Lưu ý: Sự mã hóa về màu sắc trong YUV tối ưu hơn RGB trong thị giác máy tính.

3. Sign Detection & Sign Classification

Sign Detection - Chuyển về hệ màu V trong YUV

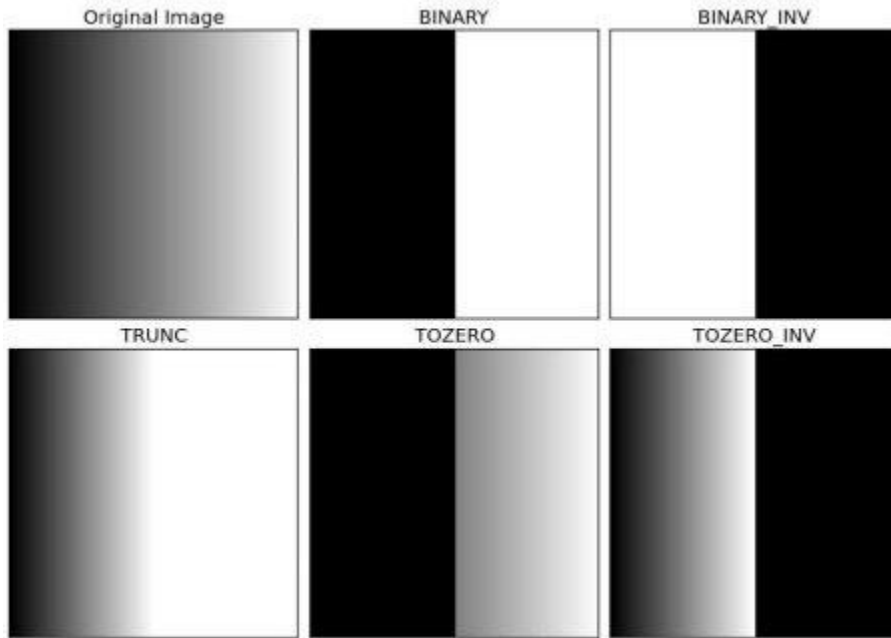


Mục tiêu:

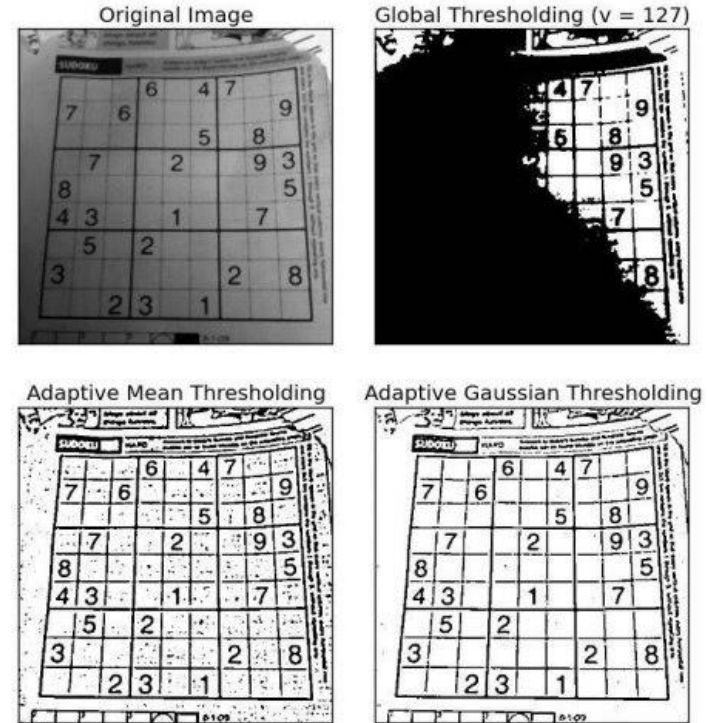
- Loại bỏ các thông tin thừa không quan tâm
- Chuyển từ ma trận ảnh 2 chiều với mỗi điểm ảnh là một mảng nhỏ chứa giá trị R, G, B
→ mỗi điểm ảnh chỉ chứa 1 giá trị V hoặc U.
=> Tăng tốc độ xử lý

3. Sign Detection & Sign Classification

Sign Detection - Phân ngưỡng ảnh (threshold)



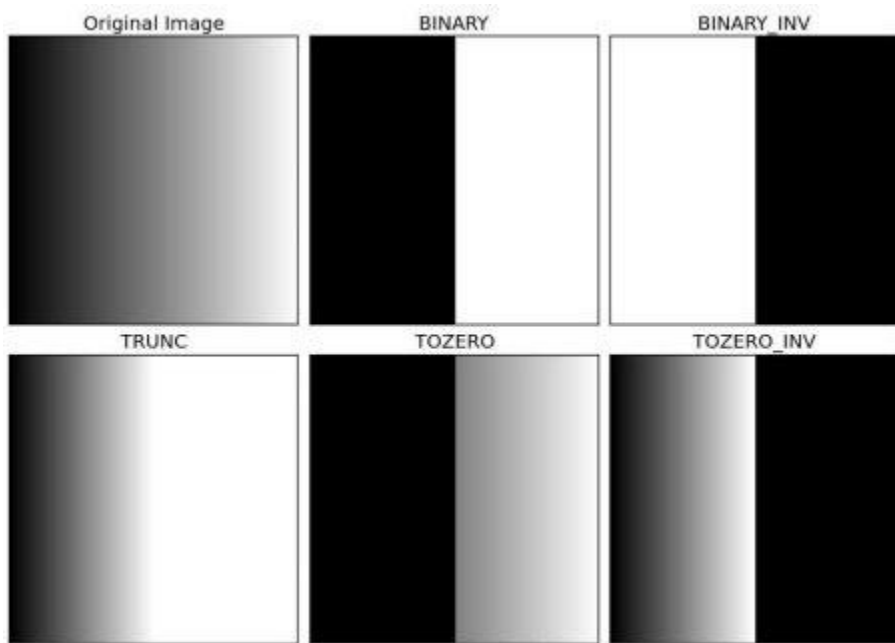
Normal Threshold



Adaptive Threshold

3. Sign Detection & Sign Classification

Sign Detection - Phân ngưỡng ảnh (threshold)

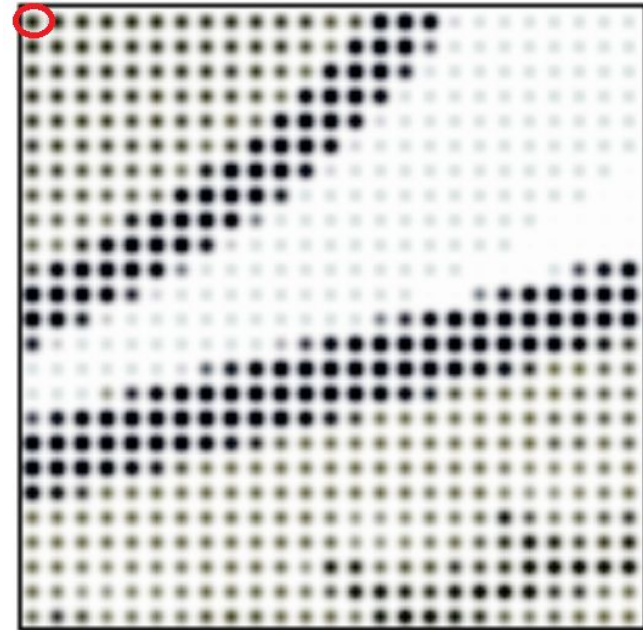
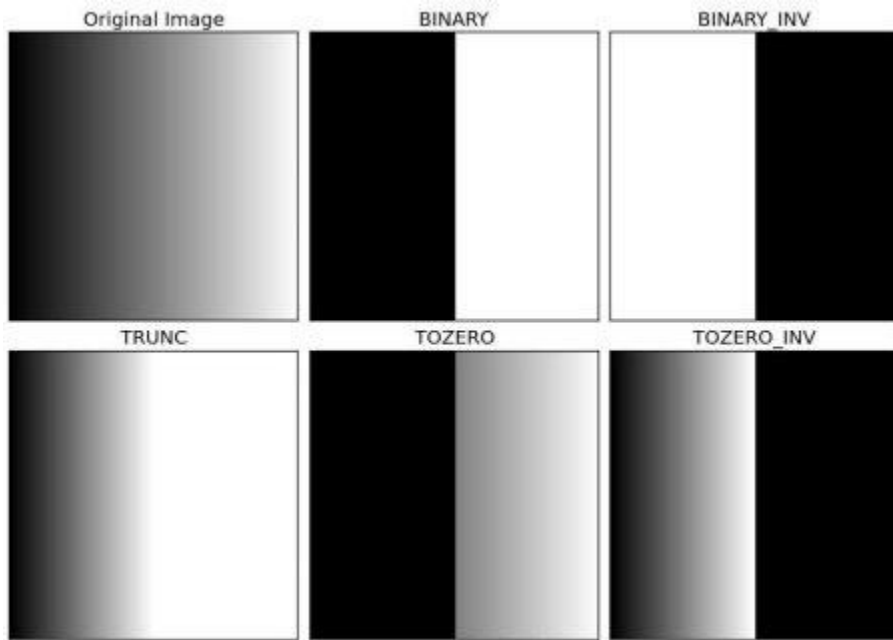


Normal Threshold

- Được áp dụng cho ảnh đã được gray-scale
- Ta sẽ duyệt qua từng pixel một và so sánh nó với ngưỡng T cho trước
- Tùy vào phương thức ta chọn BINARY hay TRUNC hay ...
- Mà hình ảnh trả về sẽ khác nhau

3. Sign Detection & Sign Classification

Sign Detection - Phân ngưỡng ảnh (threshold)



Normal Threshold

3. Sign Detection & Sign Classification

Sign Detection - Phân ngưỡng ảnh (threshold)

Điểm yếu của Normal Threshold:

- Bị ảnh hưởng bởi bóng râm

Adaptive Threshold có thể giúp tránh bị bóng râm ảnh hưởng:

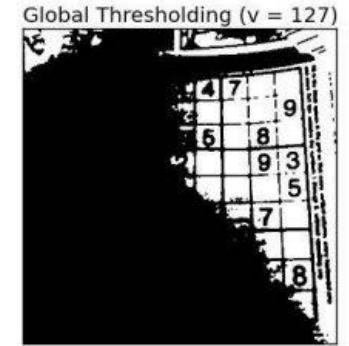
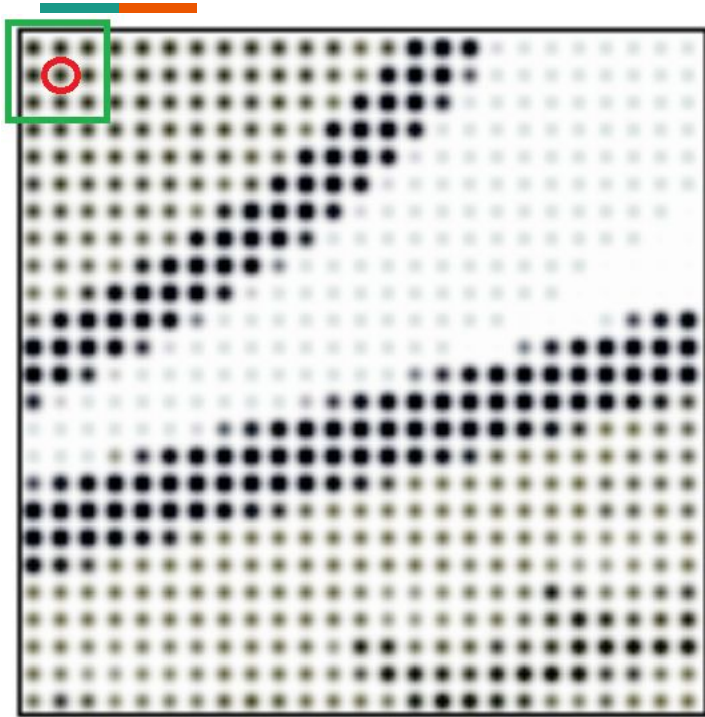
- Được áp dụng cho ảnh đã qua gray-scale
- Với mỗi pixel đang xét tới, ta sẽ tính mean của n pixel xung quanh



Adaptive Threshold

3. Sign Detection & Sign Classification

Sign Detection - Phân ngưỡng ảnh (threshold)

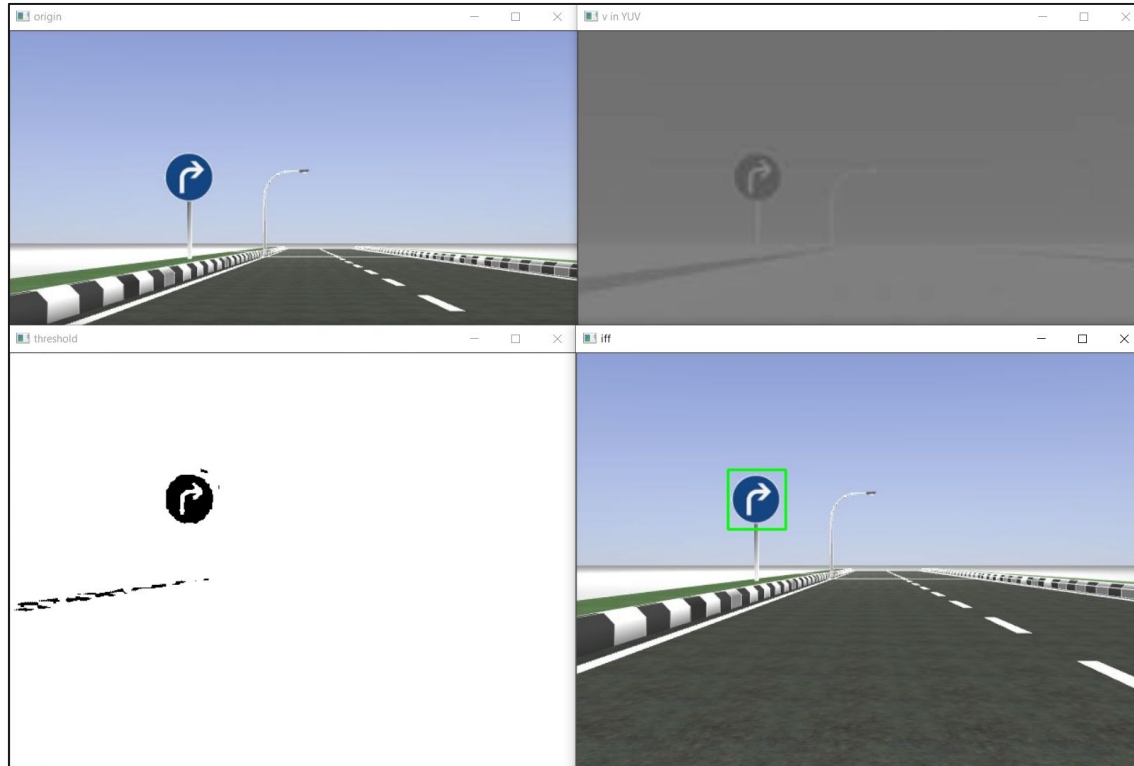


Adaptive Threshold

3. Sign Detection & Sign Classification

Sign Detection - Phân ngưỡng ảnh (threshold)

Phương thức
BINARY

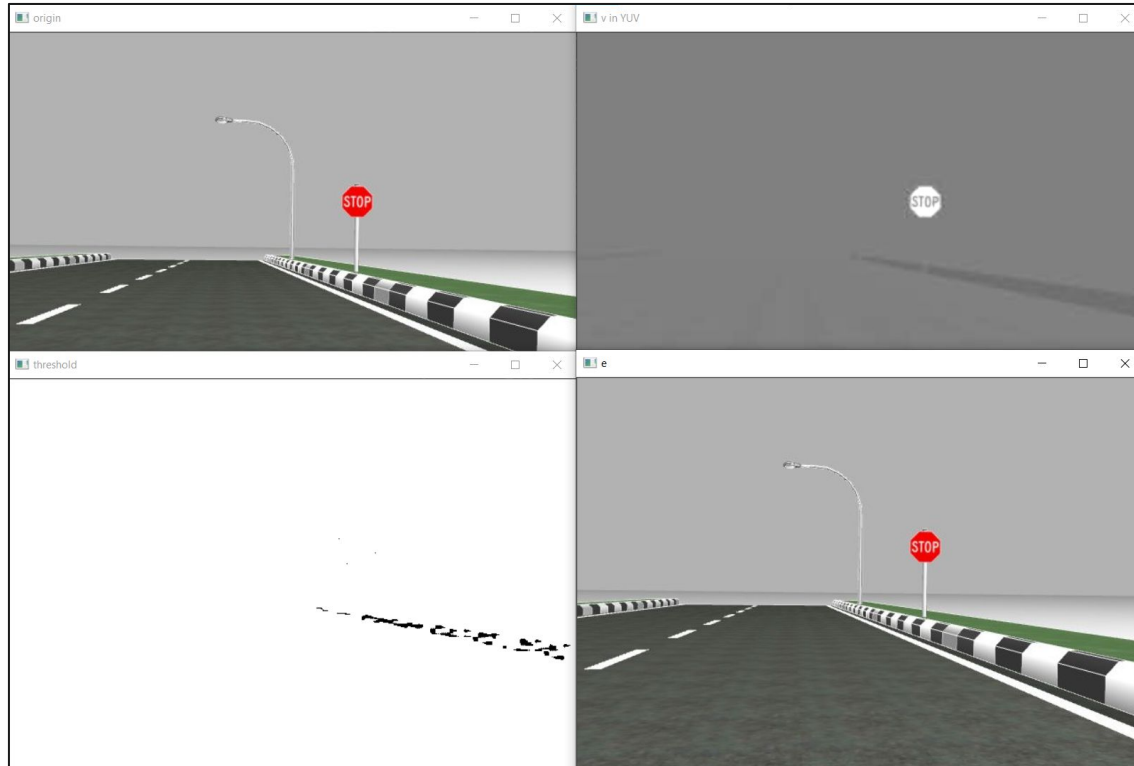


$T = 110$

3. Sign Detection & Sign Classification

Sign Detection - Phân ngưỡng ảnh (threshold)

Phương thức
BINARY

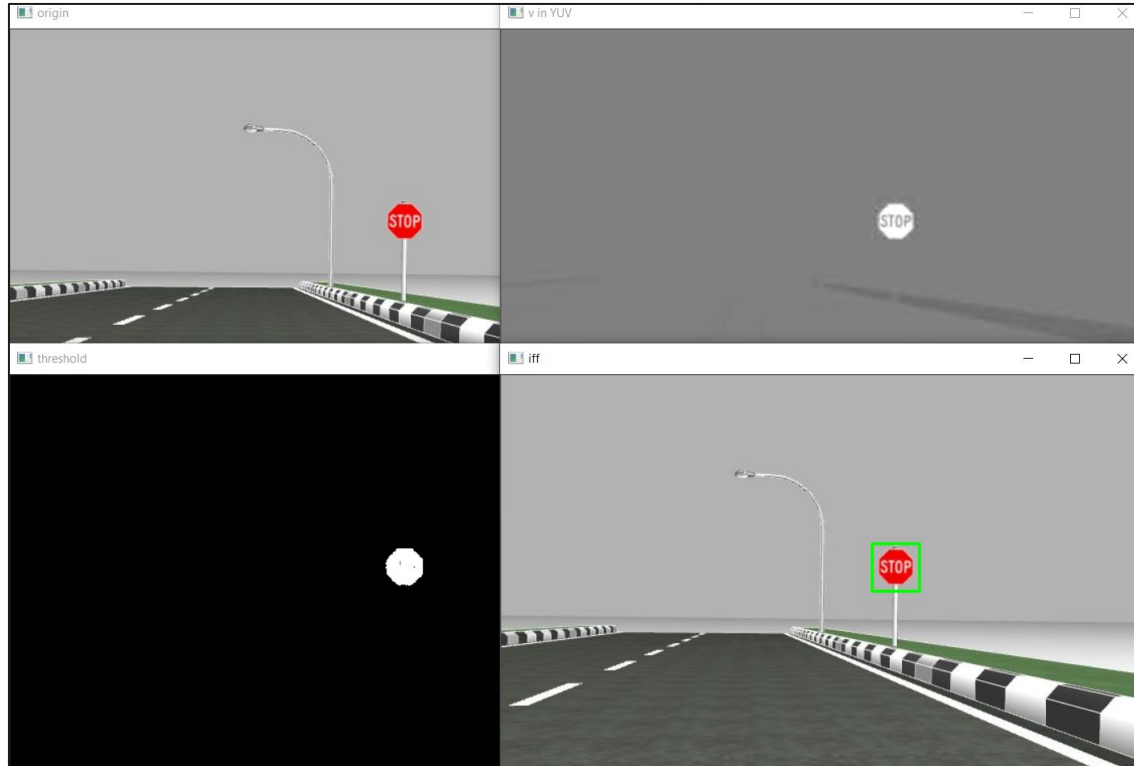


$T = 110$

3. Sign Detection & Sign Classification

Sign Detection - Phân ngưỡng ảnh (threshold)

Phương thức
BINARY



$T = 150$

3. Sign Detection & Sign Classification

Sign Detection - Tìm kiếm Contour



Contour là gì?

3. Sign Detection & Sign Classification

Sign Detection - Tìm kiếm Contour



Contour là gì?

- Contour là tập hợp các điểm liên tục mà tạo thành một đường biên bao quanh vật thể
- Ta cần thực hiện bước threshold trước, bởi **findContour()** cần độ tương phản cao để phân biệt đường biên.

3. Sign Detection & Sign Classification

Sign Detection - Tìm kiếm Contour

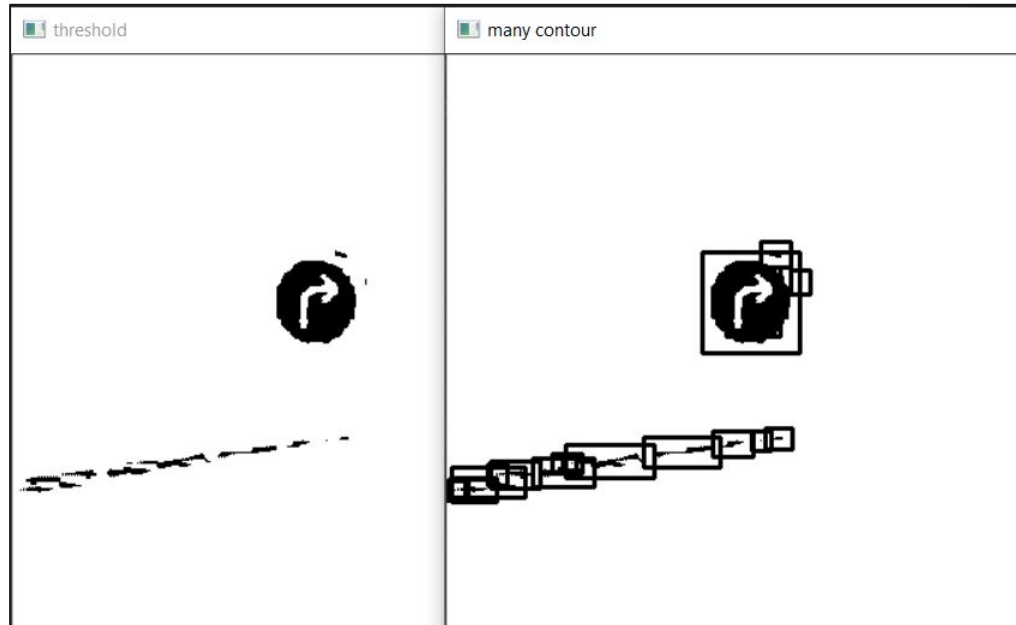


Trong thực tế, có bao nhiêu contour được phát hiện được?

3. Sign Detection & Sign Classification

Sign Detection - Tìm kiếm Contour

Trong thực tế, có bao nhiêu contour được phát hiện được?

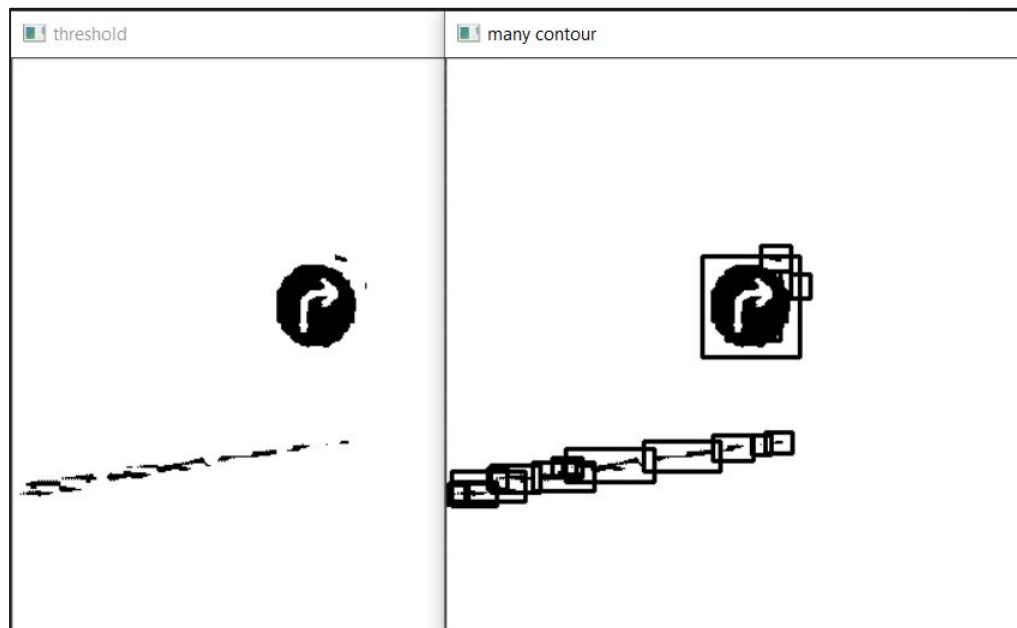


3. Sign Detection & Sign Classification

Sign Detection - Tìm kiếm Contour

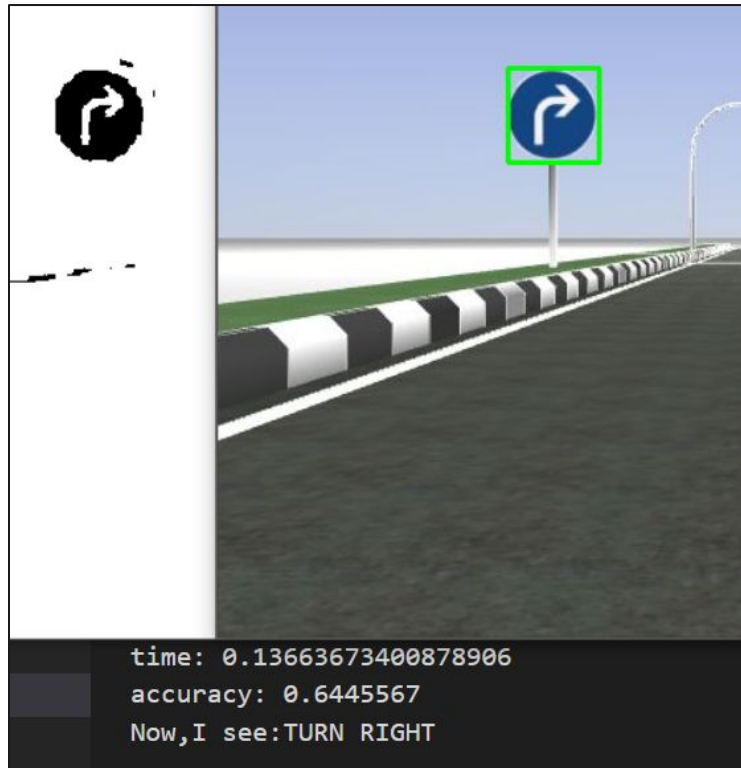
Để loại bỏ các contour dư thừa:

- Tính diện tích vùng contour đó. Và chọn hai đầu cho giá trị diện tích đó.
(Vd: $800 < S < 15.000$)
- Loại bỏ các contour có hình chữ nhật.
(Vd: $w/h < 1,25 \ \&\& \ h/w < 1,25$)



3. Sign Detection & Sign Classification

Sign Detection - Crop vùng chứa biển báo



3. Sign Detection & Sign Classification

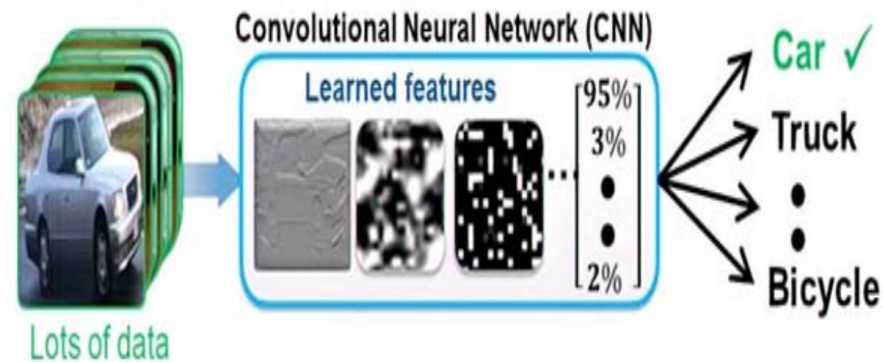
Sign Classification

Ở bước này ta cần làm 2 công việc:



TRAIN MODEL

1. TRAIN MODEL



2. ÁP DỤNG VÀO HỆ THỐNG

3. Sign Detection & Sign Classification

Sign Classification - Train model



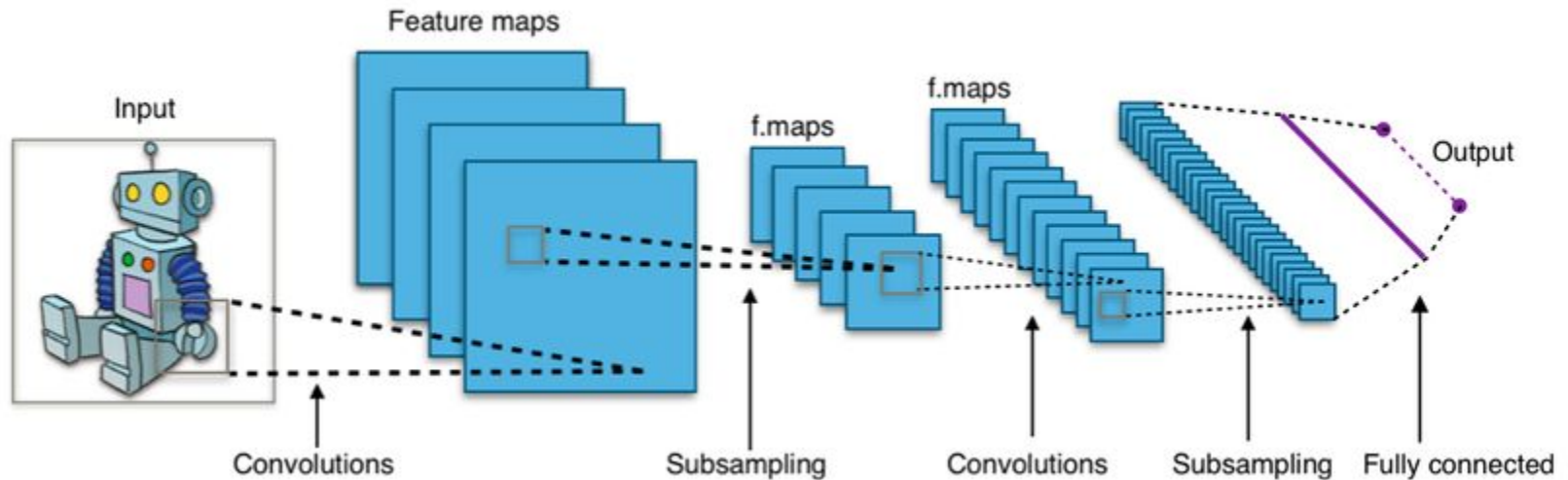
CNN là gì?

- Viết tắt cho Convolutional Neural NetWork (mạng nơ-ron tích chập)
- Đây là tập hợp của nhiều lớp Convolution chồng lên nhau.
- Mỗi neuron ở lớp kế tiếp được sinh ra từ kết quả của filter áp đặt lên một vùng ảnh của neuron trước đó

3. Sign Detection & Sign Classification

Sign Classification - Train model

CNN là gì?



3. Sign Detection & Sign Classification

Sign Classification - Train model

Sliding window trong CNN

- Khối màu cam 3x3 sẽ trượt qua ma trận ảnh như hình
- Và áp dụng cơ chế, hay công thức gì đó để trả về giá trị bên ma trận mới

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

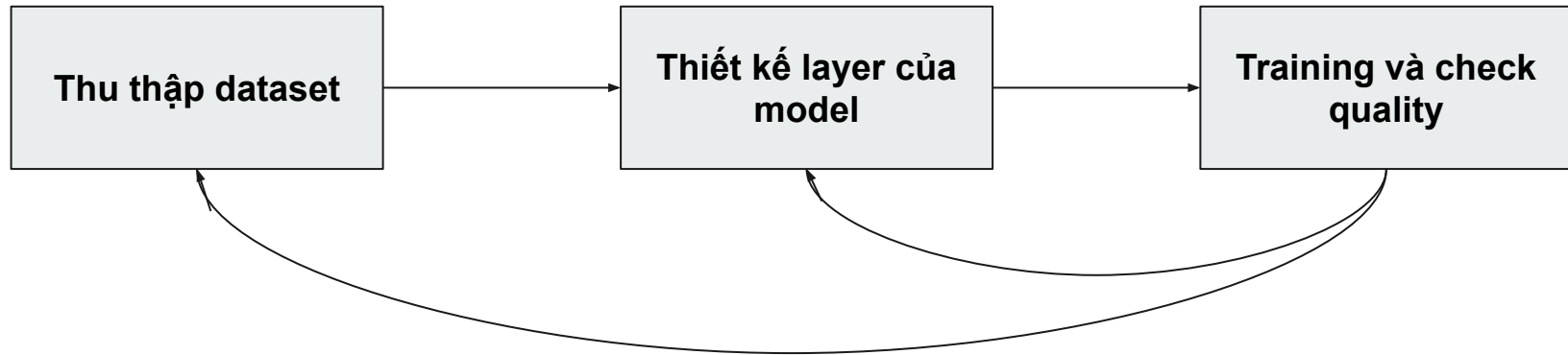
Convolved
Feature

3. Sign Detection & Sign Classification

Sign Classification - Train model

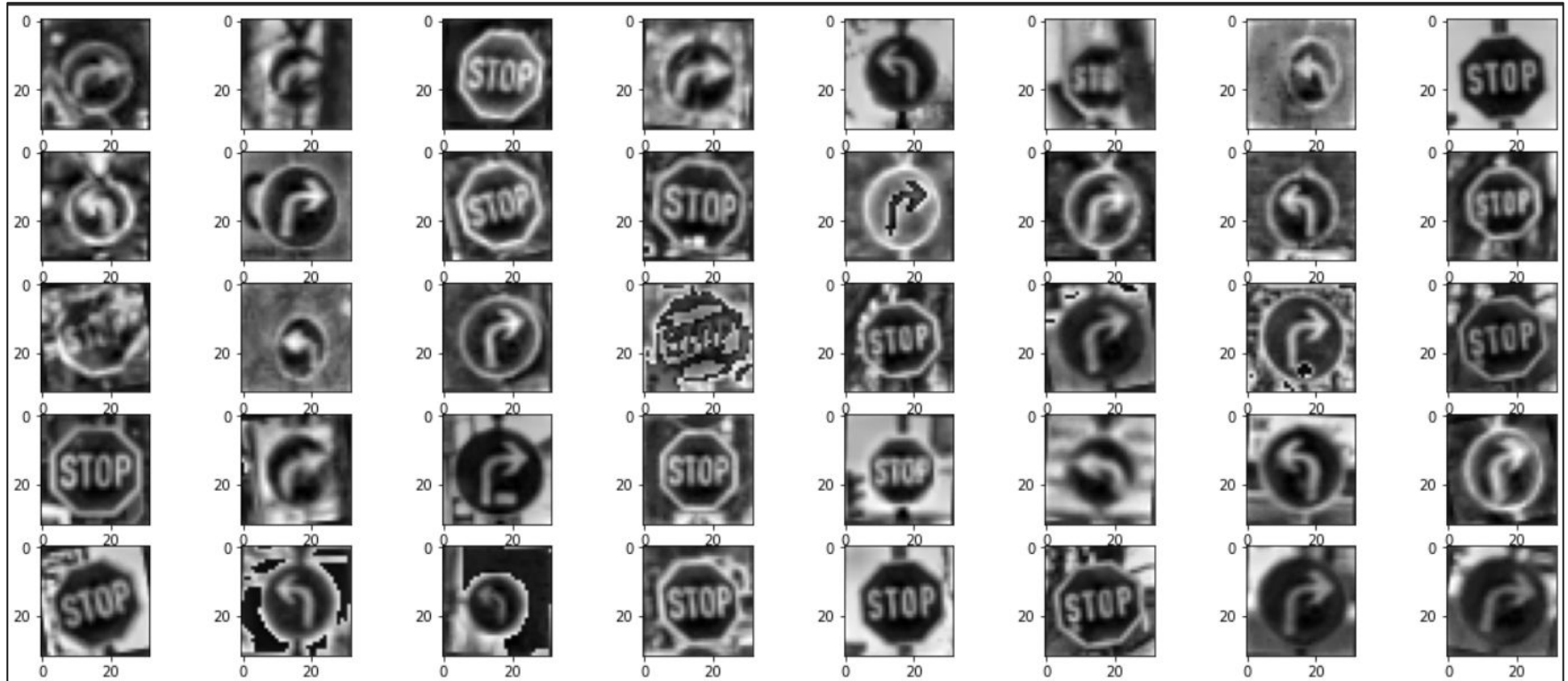


Quy trình của Train model



3. Sign Detection & Sign Classification

Sign Classification - Train model - Thu thập dataset



3. Sign Detection & Sign Classification

Sign Classification - Train model - Thiết kế layer cho model

▼ Summary Model

✓
0
giây

▶ `model[0].summary()`

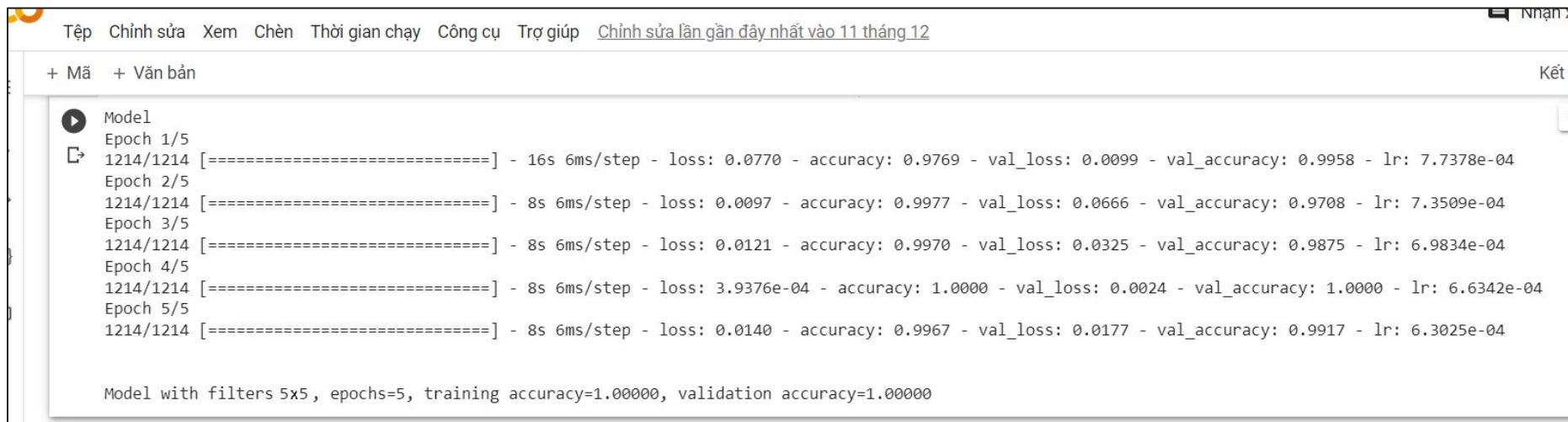
📄 Model: "sequential_2"

Layer (type)	Output Shape	Param #
=====		
conv2d_2 (Conv2D)	(None, 32, 32, 32)	832
max_pooling2d_2 (MaxPooling 2D)	(None, 16, 16, 32)	0
flatten_2 (Flatten)	(None, 8192)	0
dense_4 (Dense)	(None, 370)	3031410
dense_5 (Dense)	(None, 3)	1113

=====
Total params: 3,033,355
Trainable params: 3,033,355
Non-trainable params: 0
=====

3. Sign Detection & Sign Classification

Sign Classification - Train model - Kết quả training model và kiểm tra chất lượng



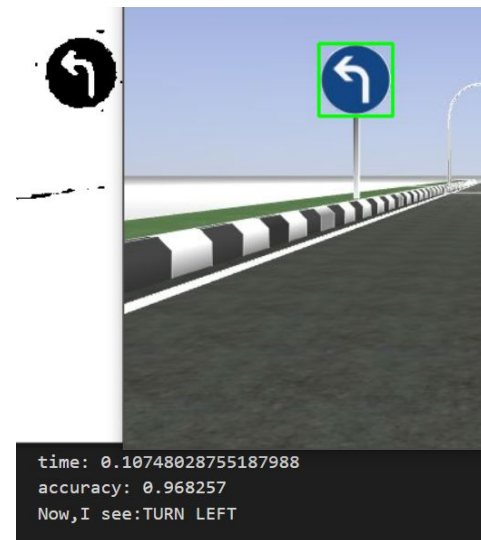
```
Tệp  |  Chỉnh sửa  |  Xem  |  Chèn  |  Thời gian chạy  |  Công cụ  |  Trợ giúp  |  Chỉnh sửa lần gần đây nhất vào 11 tháng 12
+ Mã  + Văn bản  Kết
Model
Epoch 1/5
1214/1214 [=====] - 16s 6ms/step - loss: 0.0770 - accuracy: 0.9769 - val_loss: 0.0099 - val_accuracy: 0.9958 - lr: 7.7378e-04
Epoch 2/5
1214/1214 [=====] - 8s 6ms/step - loss: 0.0097 - accuracy: 0.9977 - val_loss: 0.0666 - val_accuracy: 0.9708 - lr: 7.3509e-04
Epoch 3/5
1214/1214 [=====] - 8s 6ms/step - loss: 0.0121 - accuracy: 0.9970 - val_loss: 0.0325 - val_accuracy: 0.9875 - lr: 6.9834e-04
Epoch 4/5
1214/1214 [=====] - 8s 6ms/step - loss: 3.9376e-04 - accuracy: 1.0000 - val_loss: 0.0024 - val_accuracy: 1.0000 - lr: 6.6342e-04
Epoch 5/5
1214/1214 [=====] - 8s 6ms/step - loss: 0.0140 - accuracy: 0.9967 - val_loss: 0.0177 - val_accuracy: 0.9917 - lr: 6.3025e-04

Model with filters 5x5, epochs=5, training accuracy=1.00000, validation accuracy=1.00000
```

3. Sign Detection & Sign Classification

Sign Classification - Train model - Kết quả training model và kiểm tra chất lượng

Trong điều kiện ánh sáng đầy đủ



3. Sign Detection & Sign Classification

Sign Classification - Train model - Kết quả training model và kiểm tra chất lượng

Trong điều kiện ánh sáng vừa



Trong điều kiện ánh sáng tối



3. Sign Detection & Sign Classification

Sign Classification - Train model - Áp dụng vào hệ thống



Để model dự đoán ảnh, ta sẽ sử dụng hàm `model.predict()` và truyền vào hàm bức hình đã crop ở bước sign detection

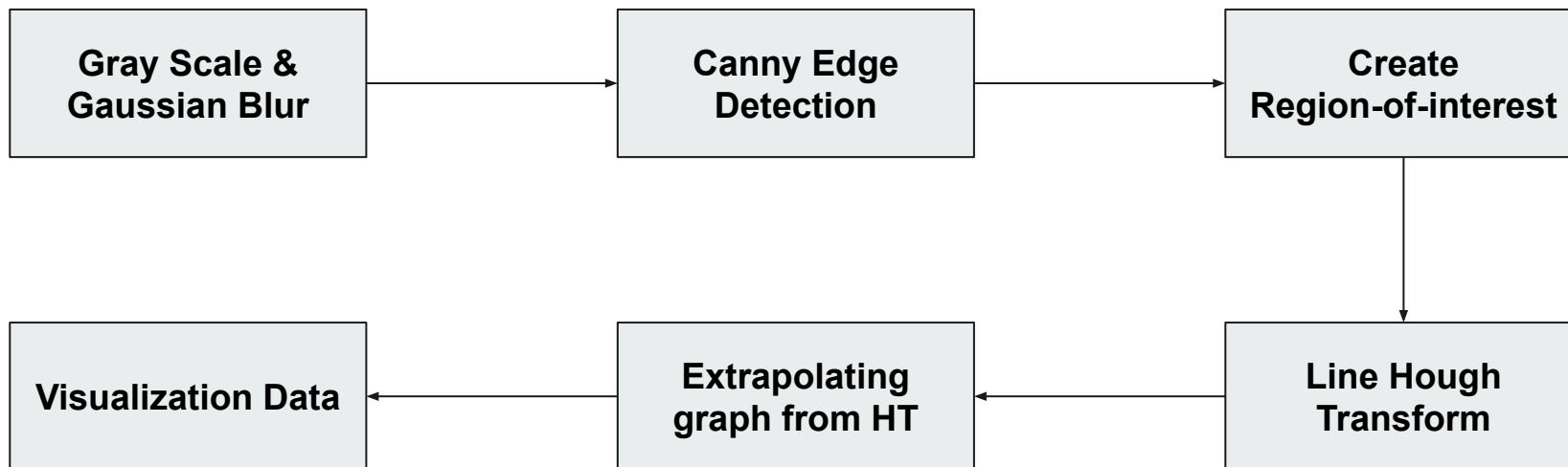
```
prediction = model.predict(img)

#in ra biển báo nào, số trả về từ 0 đến 2
print(np.argmax(prediction))

#in ra độ chính xác khi nhận diện biển báo đó
print(np.amax(prediction))
```

4. Lane Detection

Lane Detection



4. Lane Detection

Image Preprocessing

GRAYSCALE

- $(R, G, B) \rightarrow \text{GRAY}$
- Chúng ta sẽ loại bỏ nhiều thông tin không cần thiết để xử lý.
- Là một cách để chuyển từ ma trận 3D sang 2D => Tiết kiệm hiệu năng tính toán



4. Lane Detection

Image Preprocessing

Gaussian Blur

- Kỹ thuật làm mờ (làm mịn) một hình ảnh bằng cách sử dụng chức năng Gaussian để giảm mức độ nhiễu.
- Được coi là một bộ lọc giúp giảm nhiễu hình ảnh và các chi tiết có thể bỏ qua trong hình ảnh



4. Lane Detection

Image Preprocessing

Gaussian Blur

- Tích chập ma trận hình ảnh với Gaussian Kernel.
- Thư viện khuyên dùng, chúng ta sẽ sử dụng Gaussian Kernel 5×5



4. Lane Detection

Canny Edge Detection

Step of Canny Edge Detection

1. Noise Reduction
2. Finding Intensity Gradient of the Image
3. Non-maximum Suppression
4. Hysteresis Thresholding



4. Lane Detection

Canny Edge Detection

STEP 1: Noise Reduction

- Tích chập với ma trận Gaussian Kernel



4. Lane Detection

Canny Edge Detection

STEP 2: Finding Intensity Gradient of the Image

- Lọc bằng Sobel Kernel theo cả hướng ngang và dọc để thu được đạo hàm bậc nhất theo hướng ngang (G_x) và hướng dọc (G_y)
- Từ đó, chúng ta có thể tìm thấy độ dốc và cường độ sau

- $edge_gradient(G) = \sqrt{G_x^2 + G_y^2}$

- $angle(\theta) = \arctan(G_y/G_x)$

4. Lane Detection

Canny Edge Detection



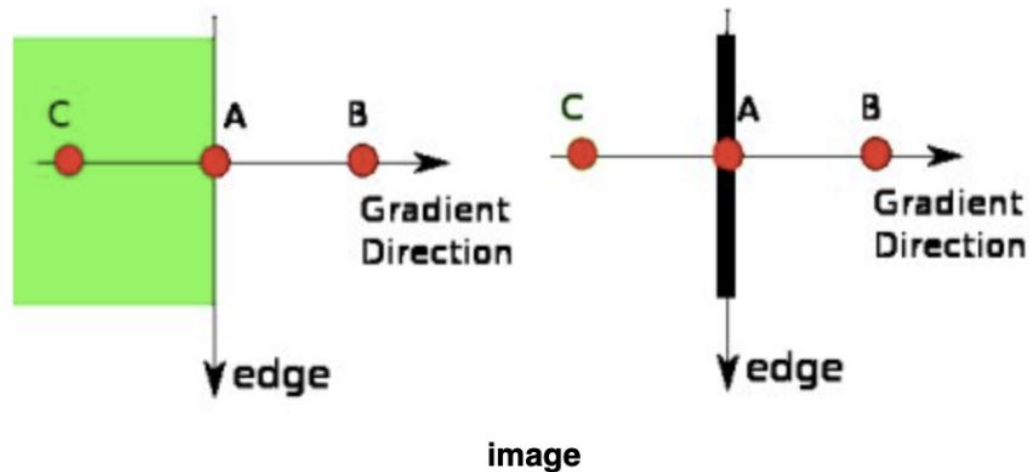
STEP 3: Non-maximum Suppression

- Sau khi nhận được độ lớn và hướng của gradient, quá trình quét toàn bộ hình ảnh được thực hiện để loại bỏ bất kỳ pixel không mong muốn nào có thể không tạo thành cạnh
- Tại mỗi pixel, pixel được kiểm tra xem nó có phải là cực đại cục bộ trong vùng lân cận của nó theo hướng gradient hay không

4. Lane Detection

Canny Edge Detection

STEP 3: Non-maximum Suppression

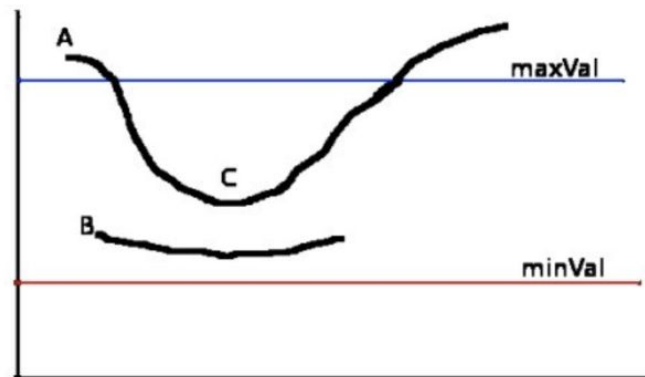


4. Lane Detection

Canny Edge Detection

STEP 4: Hysteresis Thresholding

- Giai đoạn này quyết định đâu là cạnh thực sự là cạnh và đâu không thực sự là cạnh
- Cần hai giá trị ngưỡng, `minVal` và `maxVal`
- $|G| > \text{maxVal}$
- $|G| < \text{minVal}$
- $\text{minVal} \leq |G| \leq \text{maxVal}$



4. Lane Detection

Create Region-of-interest



Region-of-interest

- Vùng chú ý (Region-of-interest) đối với camera của xe tự hành chỉ là hai làn đường hiện thời trong vùng quan sát của nó.
- Lọc ra các pixel không liên quan bằng cách tạo một vùng đa giác và loại bỏ tất cả các pixel khác không có trong đa giác.

4. Lane Detection

Create Region-of-interest



4. Lane Detection

Hough Transform

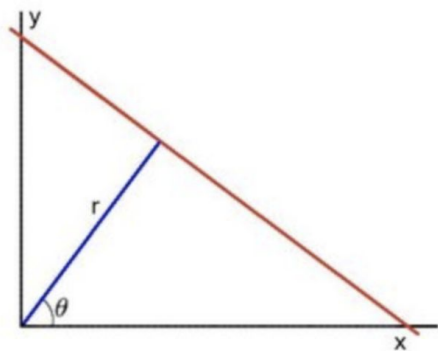


Hough Transform

- **Mục đích:** Loại bỏ cạnh cong, giữ lại cạnh thẳng.
- Toạ độ Cartesian: $y = mx + b$, (m, b)
- Toạ độ cực (Polar): $r = x\cos(\theta) + y\sin(\theta)$, (r, θ)
- Tương quan qua: $y = \left(-\frac{\cos \theta}{\sin \theta}x + \frac{r}{\sin \theta}\right)$

4. Lane Detection

Hough Transform

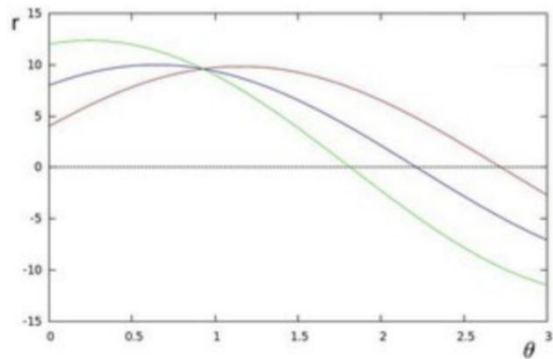
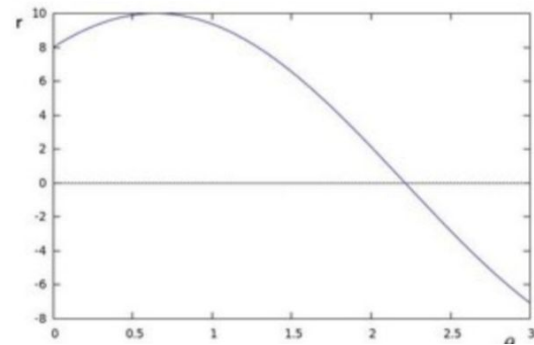


$$(x_0, y_0) = (8, 6)$$

$$(x_0, y_0) = (8, 6)$$

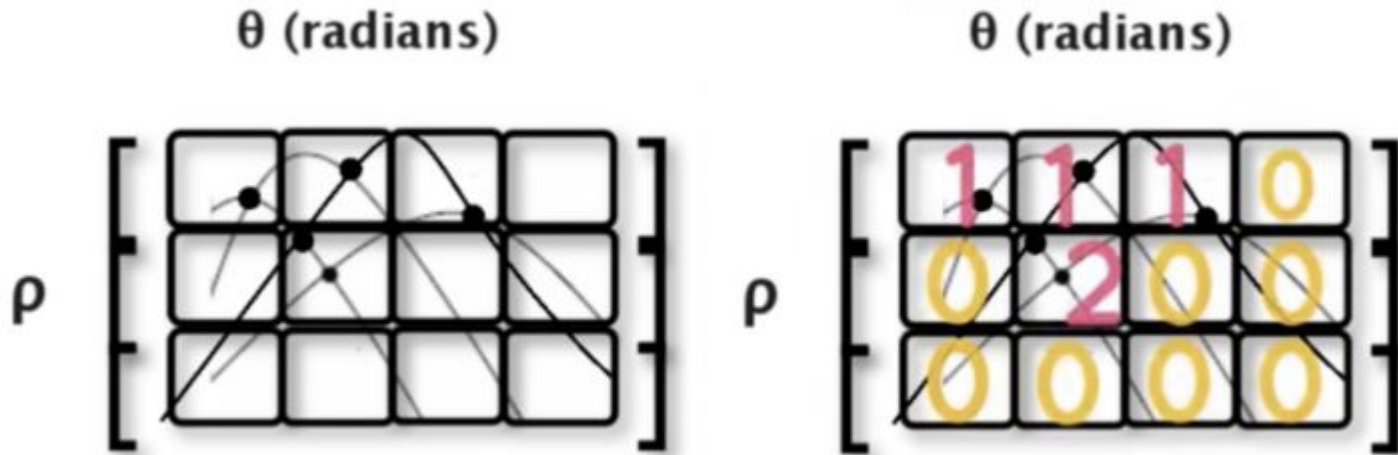
$$(x_0, y_0) = (4, 9)$$

$$(x_0, y_0) = (12, 3)$$



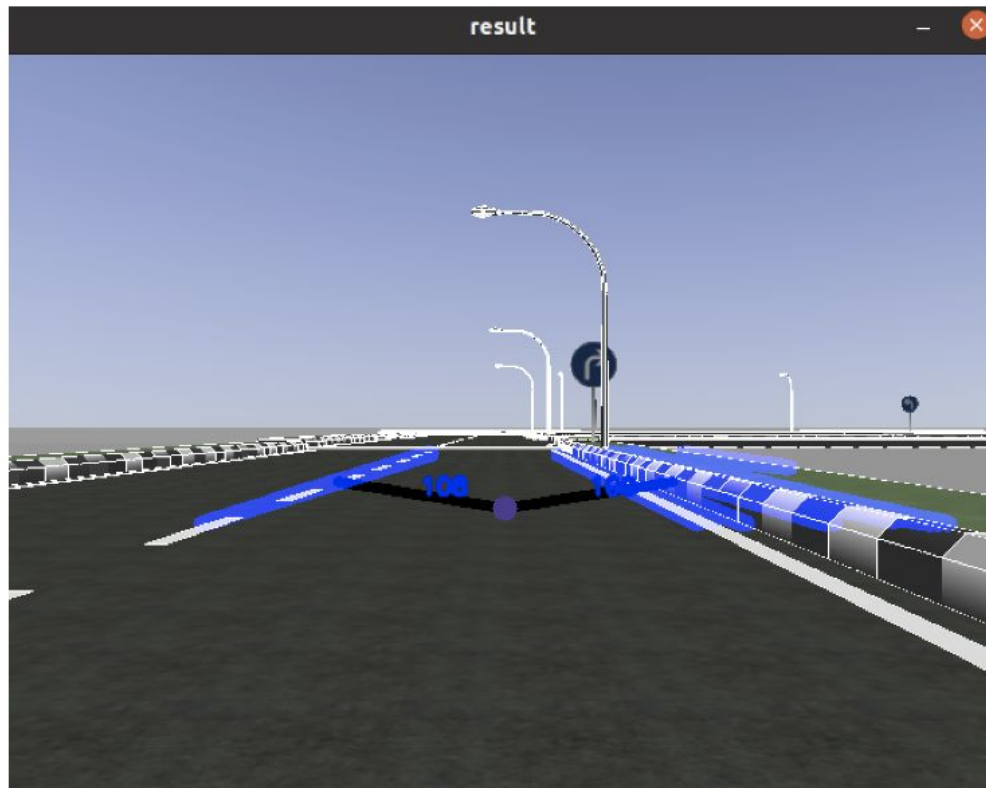
4. Lane Detection

Hough Transform



4. Lane Detection

Hough Transform



4. Lane Detection

Visualization



Visualization

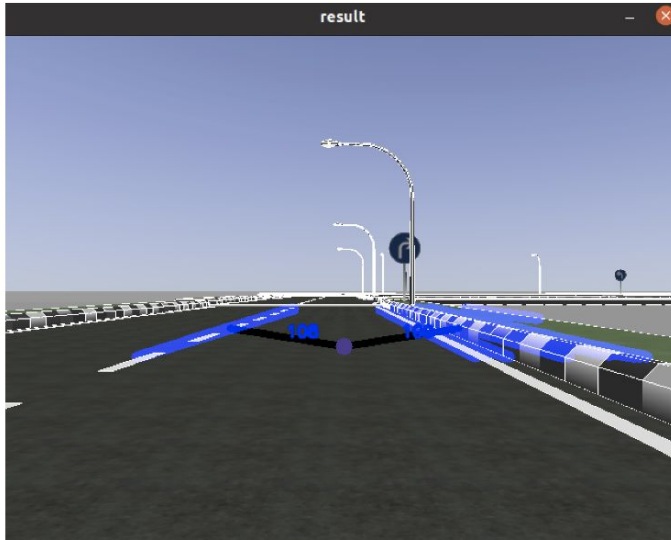
- **Xử lí các đường chắn ngang:** Dựa vào tham số m của $y = mx + b$
- **Xử lí các đường rời rạc:** Đối với những đường thẳng bị tách rời rạc nhưng có xu hướng gần và nối liền nhau, ta sẽ tính trung bình để tạo ra được một đường thẳng mới duy nhất
- **Xử lí khoảng cách trái/ phải:** đo khoảng cách giữa một điểm trung tâm của camera với làn trái và phải

4. Lane Detection

Visualization

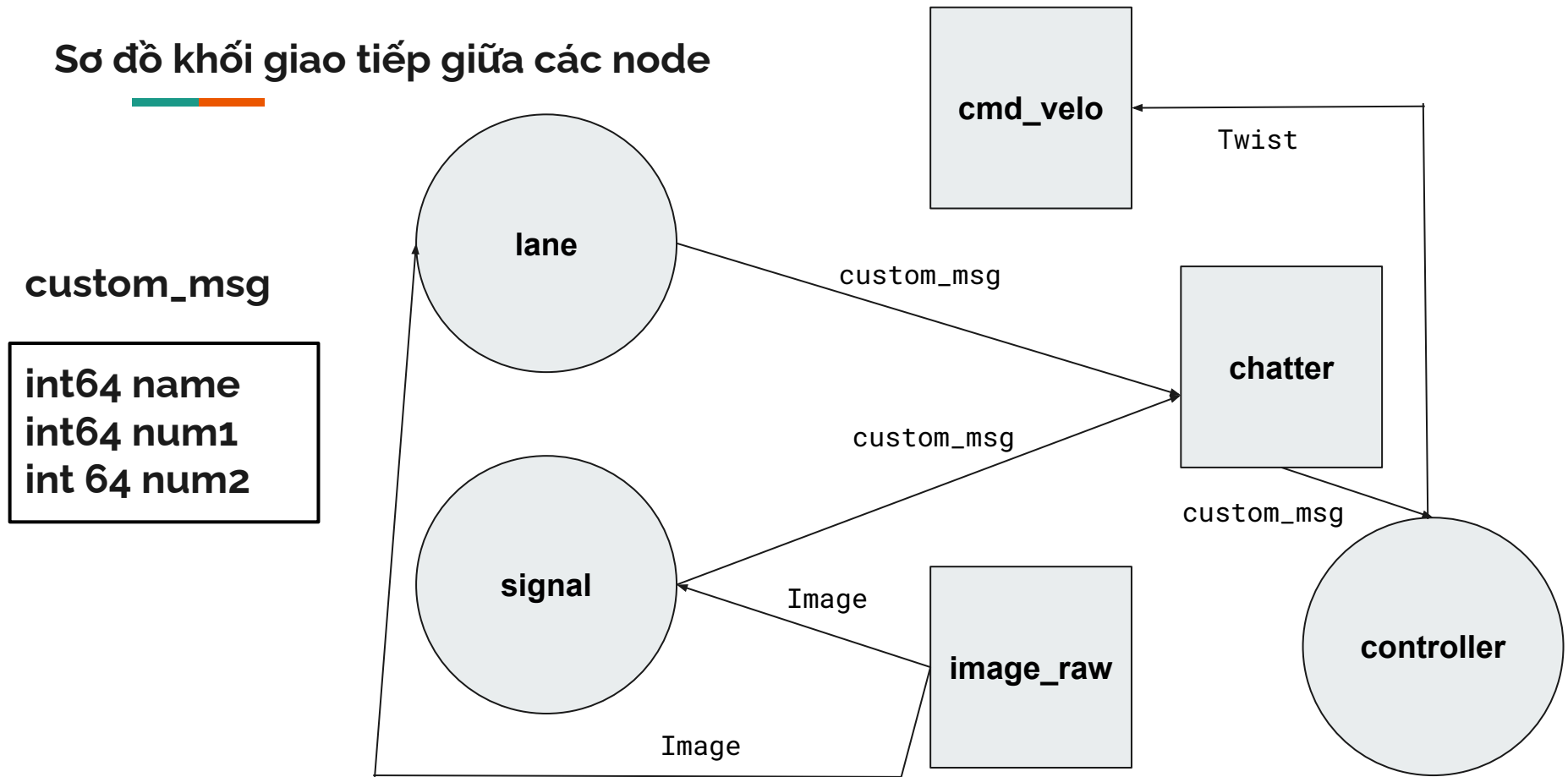


Visualization



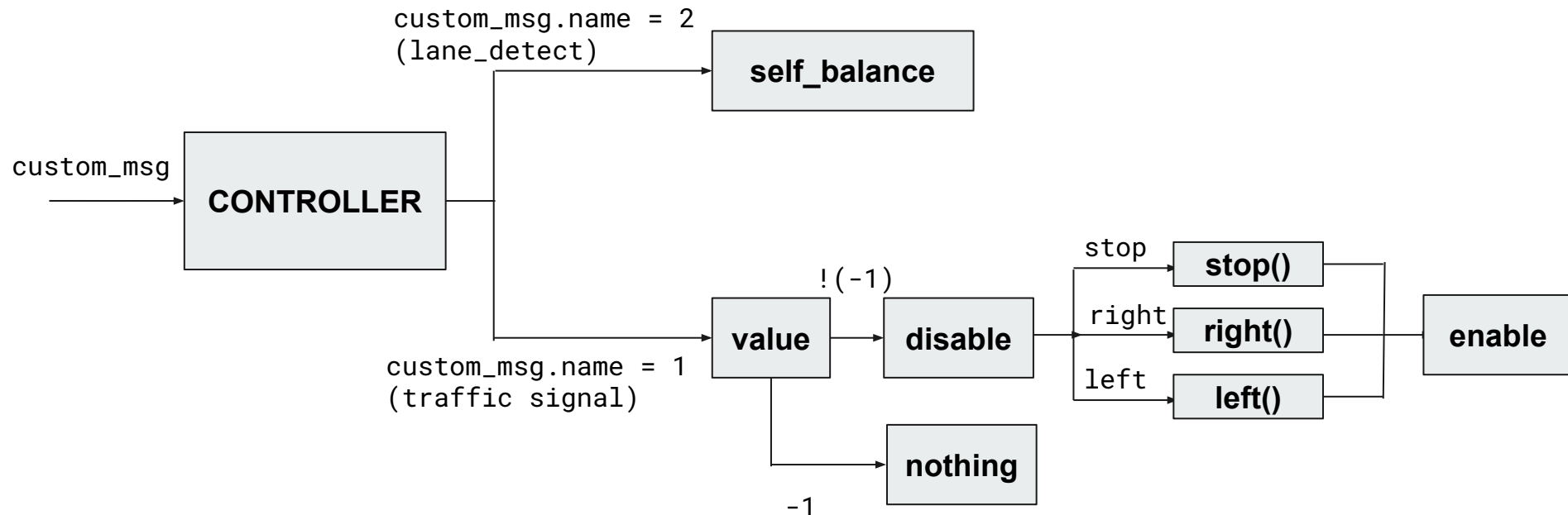
5. Tổng hợp hệ thống

Sơ đồ khối giao tiếp giữa các node



5. Tổng hợp hệ thống

Sơ đồ khối của khối điều khiển (Controller)



6. Video demo



THE END

Thank you for your listening !!!



*Thank
You*