

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
UTE-AI LABORATORY



UIT CAR RACING 2021 - MÙA X
(BẢNG CHUYÊN NGHIỆP)

Giảng viên hướng dẫn: TS. Trần Vũ Hoàng

Thành viên nhóm:

1. Đỗ Trần Nhật Tường
2. Đặng Công Ty
3. Trần Nhật Thắng
4. Lê Vĩnh Thái

➤ **Cấu hình server BTC:**

- CPU: Intel I9 10900x
- GPU: 3060TI 8G
- RAM: 32GB bus 3200

➤ **Quy chế thi đấu và kết quả:**

- Các đội sẽ thi đấu đơn lẻ trên phần mềm mô phỏng do BTC cung cấp. Mỗi đội sẽ có 2 lượt thi đấu, kết quả sẽ lấy lượt thi đấu có số điểm cao nhất, kết quả sẽ được tính dựa trên số checkpoint và thời gian xe hoàn thành checkpoint cao nhất.
- Map thi sẽ không được biết trước (ngoại trừ map thi vòng chung kết sẽ được biết trước **70%**)

VÒNG SƠ KHẢO

➤ **Thách thức:**

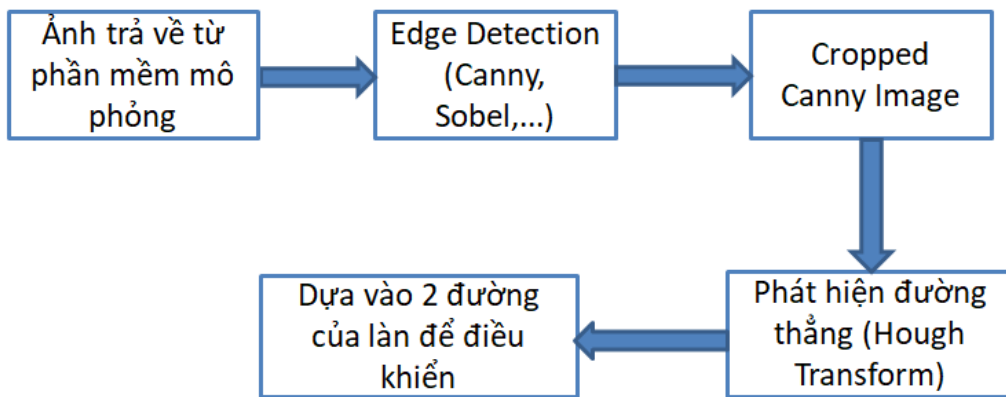
- Đường 1 chiều có 2 làn đường.
- Đường 1 chiều có 1 làn đường.
- Đường quanh co.
- Đổi loại đường, đường có điều kiện ánh sáng thay đổi, có bóng cây.

- **Yêu cầu vượt qua vòng sơ loại:** Để có thể vượt qua vòng này ta cần phải nhận diện được làn đường và từ đó có thể cho xe chạy. Do vòng này chỉ có xử lý nhận diện làn đường => nên đội phải cần chạy **tốc độ cao nhất** có thể, tuy nhiên vẫn ưu tiên chạy **10 checkpoints**

➤ **Các phương pháp:**

1. Xử lý ảnh thuần:

• **Các bước cơ bản:**

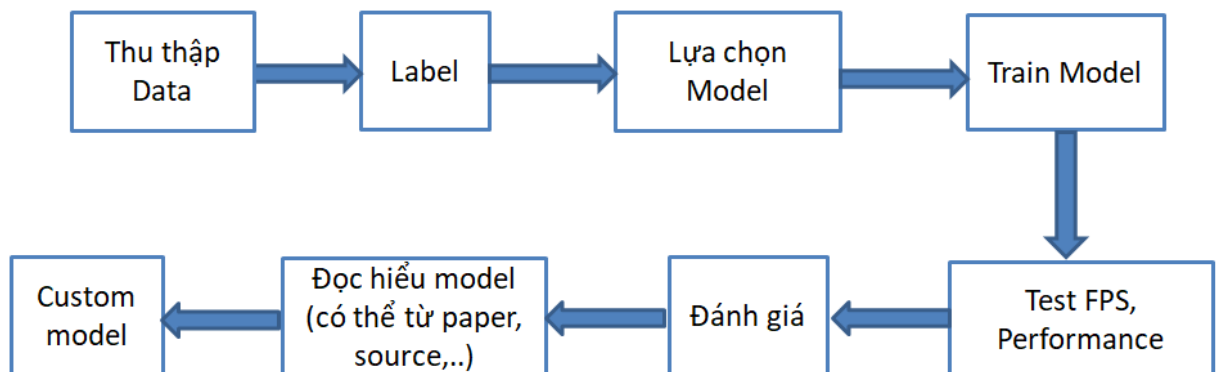


• **Ưu điểm:** Nhẹ, dễ code.

- **Nhược điểm:** Do vòng này có khá nhiều bóng cây nên đối với phương pháp này dễ gây nhiễu.

2. Segmentation:

• **Các bước cơ bản:**



- **Ưu điểm:** Xử lý gần như hoàn toàn các trường hợp nhiễu (bóng cây, thay đổi điều kiện đường, ...).

- **Nhược điểm:** Model nặng, cần hiểu rõ task Segment và các bước thu thập data.

➤ **Các vấn đề gặp phải và cách giải quyết:**

I. Segmentation

Phản nhận diện làn đường (**Segmentation**): Ban đầu nhóm dùng mạng **Bisenet** với mong muốn tăng tốc độ chạy của mạng nhưng không giảm nhiều độ chính xác. Nhưng cuộc thi lần này chỉ chạy trên mô phỏng và máy chủ của ban tổ chức cũng không yếu nên nhóm quyết định sử dụng mạng **Unet** để tăng độ chính xác. Nhóm train mạng **Unet** với

bộ dataset > **20k** ảnh (bộ Carla và dataset tự lấy từ các map mô phỏng) và kích thước ảnh đầu vào là 80x160 (**Note**: nhóm mình crop - bỏ 1 nửa ảnh phía trên để giảm nhiễu và tăng tốc độ của mạng).

Việc **thu thập data** ta sẽ sử dụng phần mềm Labelme để thu thập dữ liệu để training cho model.

Đầu tiên các bạn sẽ phải clone hoặc download file zip của phần mềm Labeme về máy. Các bạn nhấn vào link github [Labelme](https://github.com/labelme/labelme). Ở link github trên đã có sẵn cách để cài đặt vào máy nên các bạn theo hướng dẫn nha!



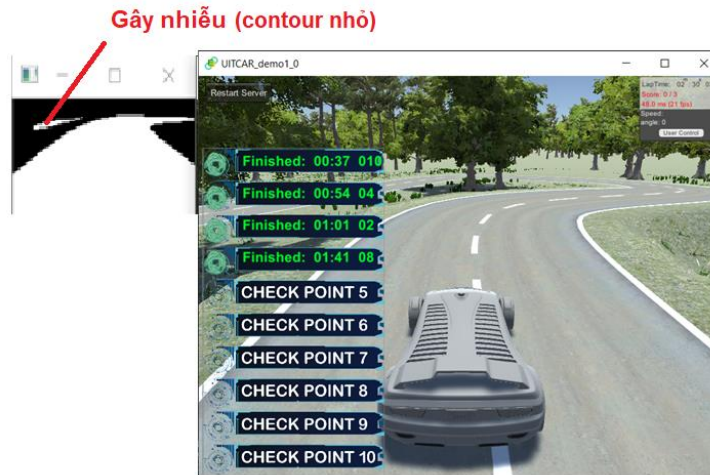
Hình 1.1 Giao diện phần mềm Labelme.

Sau đó các bạn chọn edit và Add Polygon để phân vùng giữa đường và nền. Sau khi lưu các bạn sẽ thấy xuất hiện 1 file .json như ở hình 1.2.

	0	1/7/2022 3:50 PM	JSON File	442 KB
	0	11/14/2021 6:23 PM	PNG File	366 KB

Hình 1.2. File json được tạo ra khi lưu.

Vấn đề gặp phải: Do Unet rất chính xác nên mạng bắt được những đoạn đường rất xa gây nhiễu cho việc điều khiển (hình 1.3) => Nhóm xử lý bằng cách “**remove small contour**” thì nó sẽ xóa đi các chỗ nhiễu (contour nhỏ) và chỉ giữ lại làn đường.

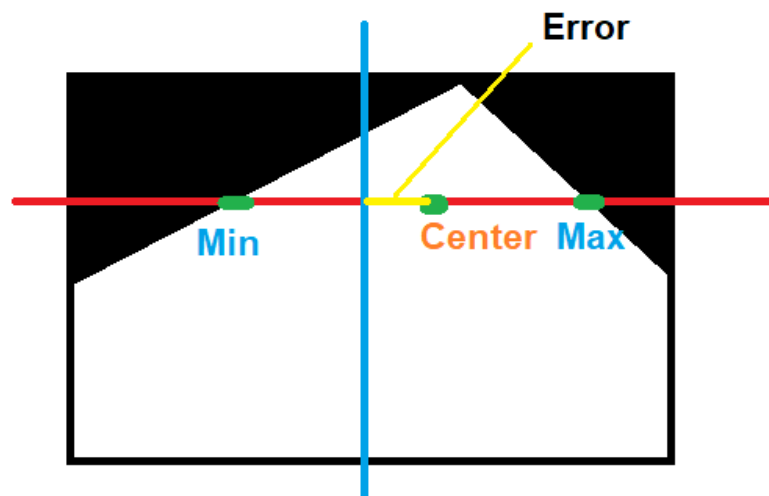


Hình 1.3. Bắt được những đoạn đường rất xa gây nhiễu cho việc điều khiển
Kinh nghiệm rút ra:

1. Training và Testing cả 2 bước đều phải đưa data như nhau (crop, scale, ... như nhau)
2. Chú ý kênh màu của OpenCV (BGR)
3. Lấy data phải đầy đủ tất cả góc nhìn.
4. Nếu chỉ thi trên mô phỏng thì đừng ngại dùng các mạng mạnh để tăng độ chính xác

II. Điều khiển:

Nhóm dùng thuật toán xác định Min–Max của làn đường để tìm “Center”(hình 1.4) từ đó ta có thể tìm được Error (khoảng cách giữa center và đường chính giữa hình (đường vàng)) để sử dụng “PID Controller”



Hình 1.4. Điều khiển xử lý trên ảnh Segmentation

1. **Vấn đề 1:** Do cấu hình máy khác nhau nên việc chọn thông số PID sẽ khác

Cách giải quyết:

- a. Sử dụng Fuzzy Control: Do vòng này có nhiều cua gấp (đường con rắn (hình 1.5a), góc cua > 90 độ (hình 1.5b), các góc cua khá gần nhau (hình 1.5c),...) + chạy tốc độ cao => sử dụng Fuzzy sẽ của không kịp



a. Đường con rắn



b. Góc cua lớn hơn 90°



c. Đường có 2 góc cua gần nhau

Hình 1.5 Các loại đường khó

- b. Vẫn sử dụng PID vì mình sẽ có thể linh hoạt thay đổi thông số PID theo góc cua gấp và khi xe chạy thẳng: Do trước vòng thi đội được test 3 lần trước khi thi chính thức => đã chọn được thông số phù hợp.

2. **Vấn đề 2:** Khi chạy đường có dốc cao (hình 1.4) với tốc độ cao => xe văng ra khỏi làn



Hình 1.4: Đường dốc cao





Cách giải quyết: Khi xe lên dốc cao thì ta không thể xác định được Min – Max, đồng nghĩa cũng không thể xác định được điểm Center (sẽ print ra “*min () arg is an empty sequence*”) => Lúc này ta sẽ cho xe phanh.









VÒNG BÁN KẾT



➤ Thử thách:

- Đường 1 chiều có 2 làn đường.
- Đường 1 chiều có 1 làn đường.
- Đường 2 chiều.
- Đường quanh co.
- Đổi loại đường, đường có điều kiện ánh sáng thay đổi, có bóng cây.
- Biển báo giao thông: đi theo hiệu lệnh biển báo.
- Ngã ba, ngã tư: đi theo chỉ dẫn của biển báo.

➤ Danh sách biển báo:

Nhóm biển báo	Kí hiệu biển báo	Tên biển báo
Biển hiệu lệnh		Hướng đi thẳng
		Các xe chỉ được phép rẽ phải
		Các xe chỉ được phép rẽ trái
		Hướng đi sang làn phải

Biển cấm		Cấm đi ngược chiều
		Cấm đi thẳng
		Cấm rẽ trái
		Cấm rẽ phải
Tín hiệu nhận biết đường hai chiều		Cấm đi ngược chiều
		Hướng đi sang làn phải
Tín hiệu nhiều		
		

		
Tín hiệu nguy hiểm		Khi gặp vật cản

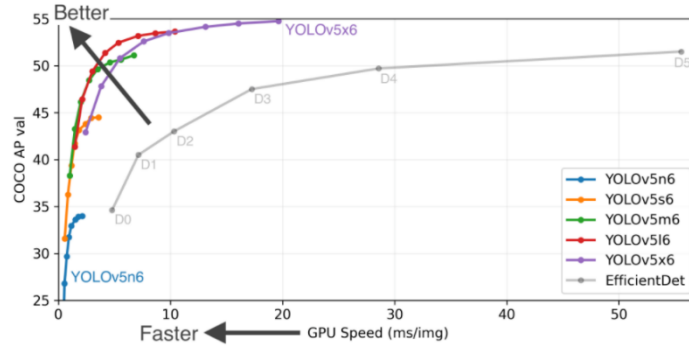
- **Yêu cầu vượt qua vòng bán kết:** Do vòng này khá đặt nặng về task Object Detection và phải nhận diện đúng chính xác biển báo. Nên các đội tập trung tăng Performance để có thể chạy được nhiều checkpoint và phần điều khiển sẽ dễ dàng hơn (ở vòng này nên chạy chậm speed ~25-30)
- **Các vấn đề gặp phải và cách giải quyết:**

I. Object Detection:

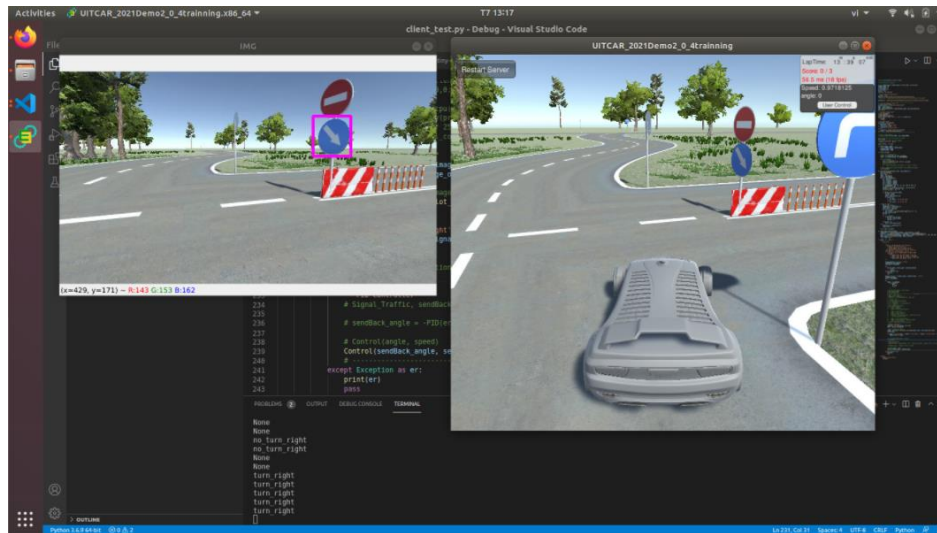
Trong vòng bán kết ở phần Object Detection, nhóm đã sử dụng mạng yolov5-n để phát hiện và nhận diện biển báo. Nhóm sẽ chia biển báo thành 5 class như sau:

- Biển báo đi thẳng
- Biển báo quẹo trái
- Biển báo quẹo phải
- Biển báo cấm đi thẳng
- Biển báo cấm đi trái
- Biển báo cấm đi phải

Ngoài ra nhóm còn so sánh mạng **Yolov4** để phát hiện biển báo và nhận diện xe nhưng không bắt xa bằng mạng **Yolov5-n** nên nhóm đã ứng dụng mạng **Yolov5-n** để phát hiện biển báo xe và tốc độ xử lý của nó cũng khá là nhanh.



Tuy nhiên thì độ chính xác của mạng **Yolov5** này tương đối bị nhầm lẫn như phát hiện biển báo trái và phải và phát hiện nhiều các yếu tố có hình dạng giống biển báo như biển báo chia làn đường (hình 2.1). Lý do là vì mạng **Yolov5** này nó vừa làm 2 nhiệm vụ đó là vừa phải **phát hiện** biển báo **và** vừa phải **phân loại** được biển báo mà vì vậy mà độ chính xác cũng tương đối không cao lắm và hay nhầm lẫn với các biển báo khác nên đã gây trở ngại cho nhóm trong việc xây dựng bộ điều khiển. (xem cách giải quyết ở vòng chung kết)



Hình 2.1. Phát hiện biển báo nhiễu

Về phần data nhóm chia thành 6 class riêng và tự label hơn 3.615 ảnh và chia các bộ data như sau:

- 727 ảnh cấm đi thẳng
- 649 ảnh cấm quẹo phải
- 590 ảnh đi thẳng
- 573 ảnh quẹo trái

- 538 ảnh cấm quẹo trái
- 538 ảnh cấm quẹo phải

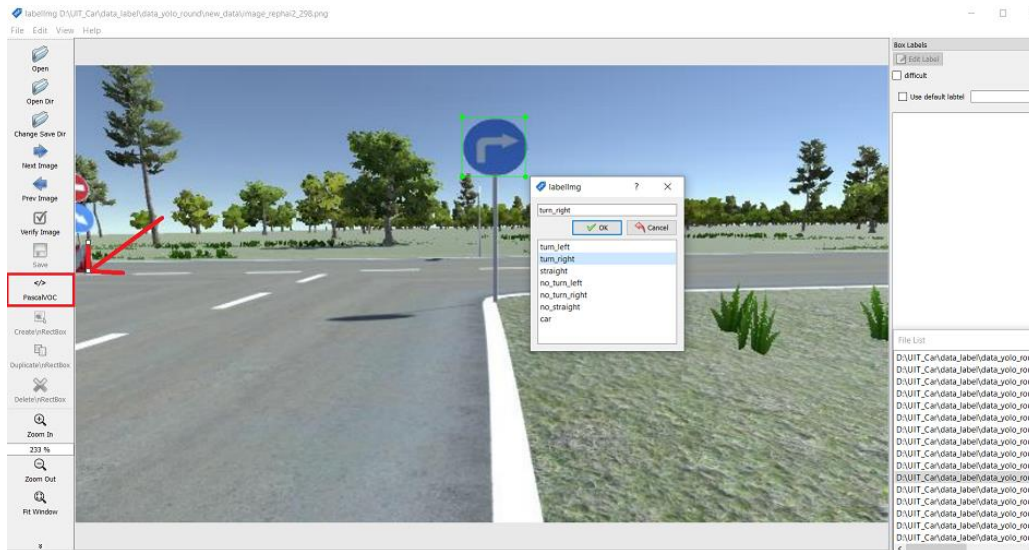
no_straight	723	<div><div></div></div>
no_turn_right	649	<div><div></div></div>
straight	590	<div><div></div></div>
turn_left	573	<div><div></div></div>
no_turn_left	538	<div><div></div></div>
turn_right	538	<div><div></div></div>

Tương tự như việc Segmentation. Ta cần thu thập dữ liệu để có thể training cho bài toán nhận diện biển báo. Ở đây, nhóm đã sử dụng phần mềm LabelImg. Đây là link của phần mềm [LabelImg](#). Các bạn có thể download và sử dụng ngay nếu sử dụng link trên.



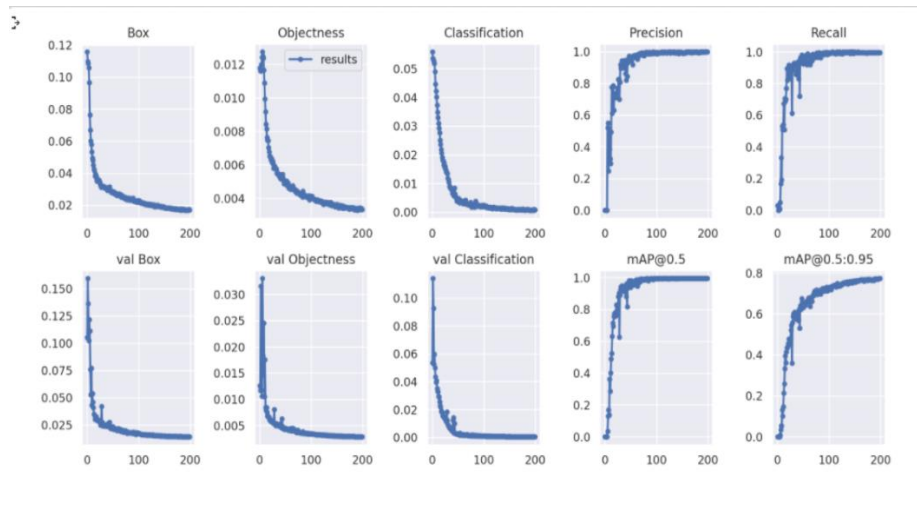
Hình 2.2. Giao diện phần mềm LabelImg.

Ngoài ra sẽ có 2 chế độ lưu là file voc hoặc file txt. Các bạn sẽ đổi ở hình 2.3.



Hình 2.3. Tạo rectangle và đổi định dạng file lưu.

Kết quả nhóm đạt được sau khi ứng dụng mạng Yolov5 trong nhận diện và phát hiện biển báo (hình 2.4).





Hình 2.4. Kết quả YOLOv5

II. Điều khiển:

Sẽ gồm có các mode:

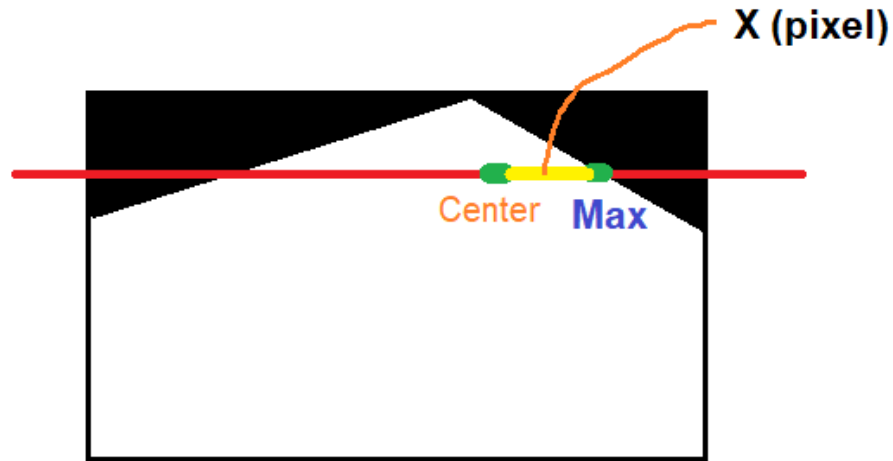
- ✓ Chạy thẳng
- ✓ Rẽ trái
- ✓ Rẽ phải

1. Vấn đề 1: Đi thẳng (Gồm có 3 biển báo: Straight, No_TurnLeft, No_TurnRight)

Đến vòng này đội sẽ tìm điểm “Center” bằng cách chỉ bấm lane phải trừ đi x(pixel) sau đó tính **Error** như vòng sơ loại => Lúc này xe chỉ lệ thuộc 1 lane đường bên phải

Ưu điểm: ít xử lý, dễ kiểm soát.

Nhược điểm: không chạy được với tốc độ cao do chỉ phụ thuộc vào 1 lane bên phải nên sẽ bị giới hạn góc cua đối với các đường cong.



Hình 2.5. Thuật toán điều khiển cho vòng bán kết

Tuy nhiên khi xe đi qua ngã tư (hình 2.6a) hoặc ngã 3 bên phải (hình 2.6b) thì xe sẽ không thể đi thẳng được vì lúc này điểm “Center” sẽ bị lệch sang bên phải => Xe sẽ rẽ phải nên không còn đúng với làn đường đi thẳng nữa.



a. Ngã tư



b. Ngã 3 bên phải

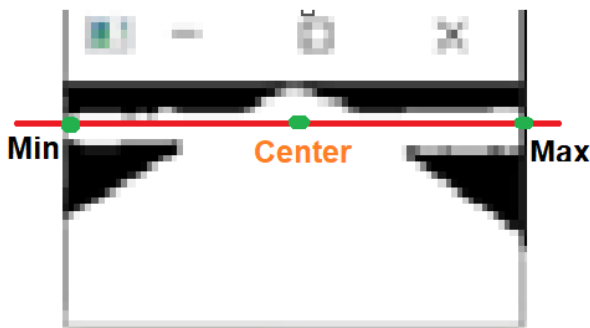
Hình 2.6. Trường hợp đi thẳng

Ý tưởng: lane nào không đúng với làn đường đi thẳng thì ta sẽ chuyển sang lane còn lại. Nếu trường hợp 2 lane không đúng thì ta sẽ bắt cả 2 lane

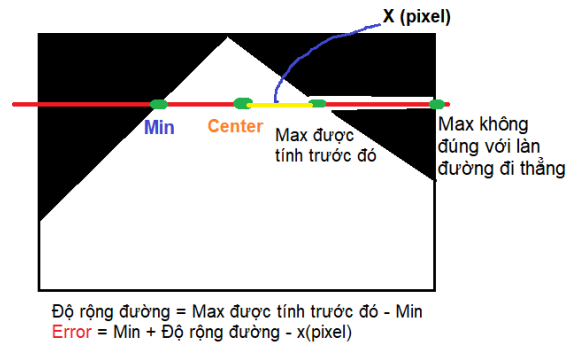
⇒ Ngã tư: xe sẽ chuyển qua điều khiển bám 2 lane (hình 2.7a)

⇒ Ngã ba bên phải: xe sẽ chuyển qua bám lane trái + Độ rộng đường - $x(\text{pixel})$ (hình 2.7b)

Trong đó: Độ rộng đường = Max – Min: Được tính khi xe chạy đường thẳng hoàn toàn (không có giao lộ, đường cong, đường con rắn,...)



a. Bắt 2 lane



b. Chuyển sang bắt lane trái

Hình 2.7. Cách xử lý với trường hợp ngã 4 và ngã 3 bên phải

2. Vấn đề 2: Rẽ phải (Gồm có 3 biển báo: Turn_Right, No_Straight, No_TurnLeft)

Nên chạy qua biển báo rồi quẹo phải vì nếu quẹo sớm sẽ bị tông cột hoặc xe sẽ ra ngoài

Hiện đội đang sử dụng cách vẽ đường ngang (giả sử đường đỏ ở chính giữa hình) (hình 2.8a), đường này phải điều chỉnh cho phù hợp. Nếu Max của đường ngang này bằng **image.shape[1]** (chiều ngang ảnh) thì lúc xe sẽ cho xe rẽ phải hay lúc này ta xác định được vị trí xe cần của phải (hình 2.8b)



a. Đường ngang xác định vị trí rẽ phải



b. Vị trí xe rẽ phải

Hình 2.8. Cách xác định vị trí rẽ phải

Chú ý: Xử lý kỹ các trường hợp đường cong (hình 2.9). Vì điểm **Max** của đường xác định vị trí của phải (đường đỏ) sẽ bằng **image.shape[1]** trước khi xe đến vị trí cần của => Dẫn đến xe của sớm => Phải điều chỉnh **đường ngang này** lại cho phù hợp.

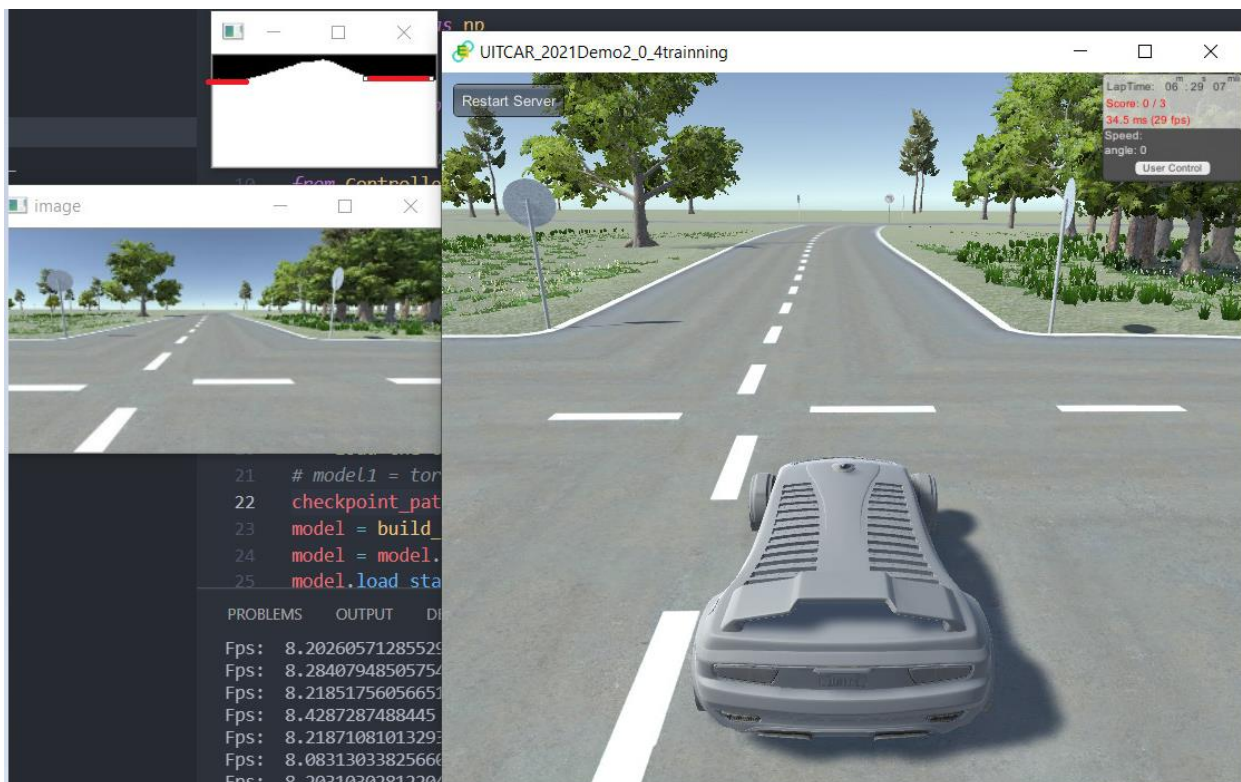


Hình 2.9. Trường hợp rẽ phải đường cong

3. Vấn đề 3: Rẽ trái (Gồm có 3 biển báo: Turn_Left, No_Straight, No_TurnRight)

Ý tưởng: Dùng Hough_lines tính khoảng cách sau đó đủ khoảng cách thì dùng **“Timer”** để xe queo trái (queo mù).

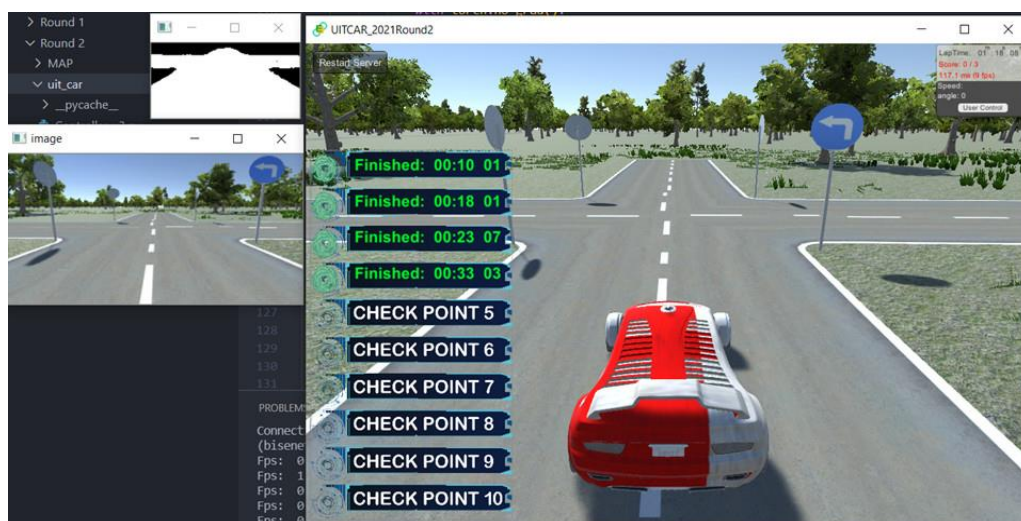
Ở vòng bán kết nhóm chỉ bắt lane phải trừ đi x(% theo độ rộng đường) pixel để điều khiển và xác định “khoảng cách” để xe cua trái dựa vào tính **trung bình** đường ngang tìm được bằng Hough_lines (hình 2.10)



Hình 2.10. Tìm khoảng cách dựa vào đường màu đỏ

Do nhóm dùng Hough_lines để tìm “vị trí của trái” liên tục nên bị nhiễu khá nhiều. Ngoài ra do lúc test map demo vòng 2 chỉ gặp trường hợp đường lớn (đường 2 làn) nên lúc thi gặp trường hợp rẽ trái sang đường 1 làn (hình 2.11) => xe bị của sớm

⇒ Sẽ được tối ưu lại vị trí tìm khoảng cách theo ý tưởng của thầy Hoàng ở vòng chung kết



Hình 2.11. Rẽ trái sang đường 1 làn.

VÒNG CHUNG KẾT

➤ Thử thách:

Tương tự như vòng bán kết nhưng có sẽ thêm:

- Vật cản (car) (hình 3.1a)
- Điều kiện đường: có thể đường tuyết, đường mưa + sấm chớp,... (hình 3.1b)



a. Vật cản (car)



b. Thay đổi điều kiện đường

Hình 3.1. Thách thức vòng chung kết

- **Yêu cầu có giải vòng chung kết (Giải thì có cũng được mà không có cũng không được nhé. 🤔🤔🤔 - Thầy Hoàng):** Ở vòng này chủ yếu cải tiến lại từ vòng bán kết nhưng phải chú ý xử lý thêm phần vật cản.
- **Các vấn đề gặp phải và cách giải quyết:**

I. Object Detection:

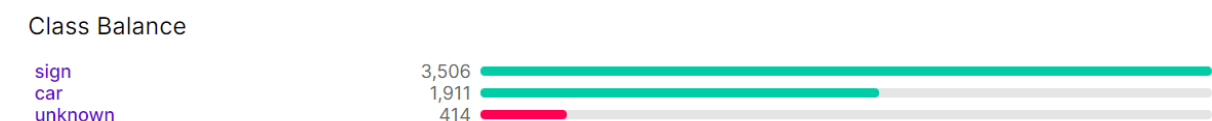
Ở phần chung kết nhóm ở phần Object Detection ngoài chia ra 6 class ở “vòng bán kết” thì nhóm còn label thêm 1 class nữa là car do đó nhóm có 7 class đó là:

- 797 ảnh xe
- 727 ảnh cấm đi thẳng
- 649 ảnh cấm quẹo phải
- 590 ảnh đi thẳng
- 573 ảnh quẹo trái
- 538 ảnh cấm quẹo trái

- 538 ảnh cấm quẹo phải



Về phần phát hiện biển báo, nhóm cũng đã sử dụng mạng Yolov5 như ở phần bán kết để phát hiện biển báo xe nhưng sang phần phân loại biển báo nhóm đã quyết định sử dụng mạng CNN để phân loại biển báo. Nhóm sẽ chuyển 6 class biển báo giao thông như là cấm đi thẳng, quẹo trái, quẹo phải, quẹo trái về thành 1 sign và 1 class unlabel (biển báo chia làn đường, biển báo xe ngựa....) và 1 class car.



Sau đó nhóm dùng mạng CNN để phân loại biển báo từ phần phát hiện biển báo của mạng Yolov5.

Việc xử lý bài toán với thời gian thực luôn là một thách thức lớn đối cuộc đua số nói riêng và các dự án realtime nói chung.

Vậy một trong những cách chúng ta có thể làm để giải quyết vấn đề này là thiết kế hoặc tối ưu model ở mức tối giản sao cho có thể cân bằng được giữa Performance và tốc độ xử lý (FPS).

Bài toán: Vào bài toán cụ thể, trong cuộc thi năm ngoái có 6 loại biển báo: trái, cấm trái, phải, cấm phải, đi thẳng và stop.

- Đầu vào là các ảnh ROI của biển báo sau khi qua bộ Object Detection.
- Đầu ra là dự đoán được loại biển báo.

II. Điều khiển

Sẽ gồm có các mode:

- ✓ Chạy thẳng
 - Chạy đường 1 lần (**thêm**)
 - Chạy đường 2 lần (**thêm**)
- ✓ Rẽ trái
- ✓ Rẽ phải

✓ Né vật cản (car) (**thêm**)

Tối ưu lại:

Ta nên chia mode chạy đường 1 làn và đường 2 làn bằng cách tính độ rộng đường (độ rộng đường phải được tính lúc xe chạy trên đường thẳng tránh trường hợp tính trong giao lộ ngã 3, ngã 4, cua gấp sẽ dẫn đến sai số)

Vì:

Đường 1 làn: thường sẽ đi mấy đoạn đường cua gấp, đường con rấn như “**vòng sơ loại**” => Cần phải tối ưu tốc độ (**nên bám 2 lane để mở rộng góc cua**)

Đường 2 làn: chỉ nên bám 1 bên lane phải trừ đi x(pixel) vì sẽ đỡ sai số khi chạy 1 bên làn phải

1. Vấn đề 1: Đi thẳng (Gồm có 3 biển báo: Straight, No_TurnLeft, No_TurnRight)

Sẽ có thêm trường hợp đi thẳng từ đường 2 làn sang đường 1 làn (*hình 3.2a*) và đường bị background ra làn đường (*hình 3.2b*)



a. Đường 2 làn sang đường 1 làn

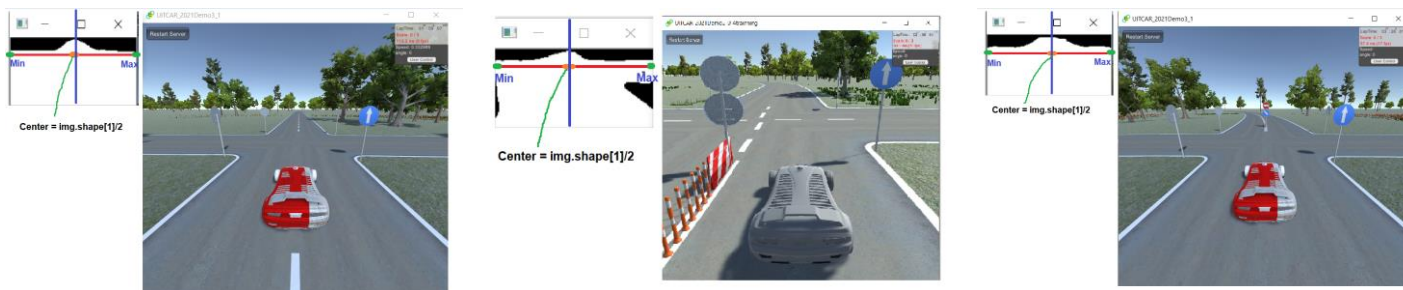


b. Đường bị background ra làn đường

Hình 3.2. Trường hợp đi thẳng trong vòng chung kết

1.1. Từ đường 1 làn sang đường 1 làn (*hình 3.3a*) **hoặc** từ đường 2 làn sang đường 2 làn (*hình 3.3b*) **hoặc** từ đường 1 làn sang đường 2 làn (*hình 3.3c*)

Vì các trường hợp trên giống nhau giữa 2 loại đường và từ đường nhỏ sang đường lớn nên việc đi thẳng khá dễ (lúc này Center = image.shape[1]/2)=> **Cho nên vẫn cho xe bám 2 lane** (xe trong giao lộ sẽ chạy mù)



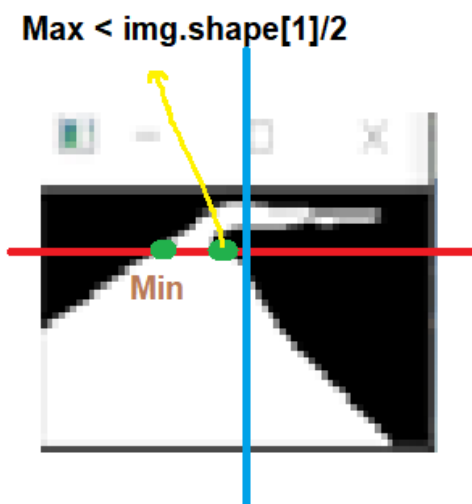
a. Đường 1 làn sang đường 1 làn b. Đường 2 làn sang đường 2 làn c. đường 1 làn sang đường 2 làn
Hình 3.3 Các trường hợp đi thẳng qua ngã tư

1.2. Từ đường 2 làn sang đường 1 làn (hình 3.2b)

Vì xe đi từ đường lớn sang đường nhỏ nên chuyển từ bám 1 làn sang bám 2 làn vì trong trường hợp này ảnh Segment rất dễ bị nhiễu và với việc chạy với tốc độ cao nên xe sẽ bị lạng (ra ngoài khoảng 5-10%) => Cần phải cho xe chạy chậm lại khi qua tới ngã tư (thêm phanh)

1.3. Đường bị background ra làn đường

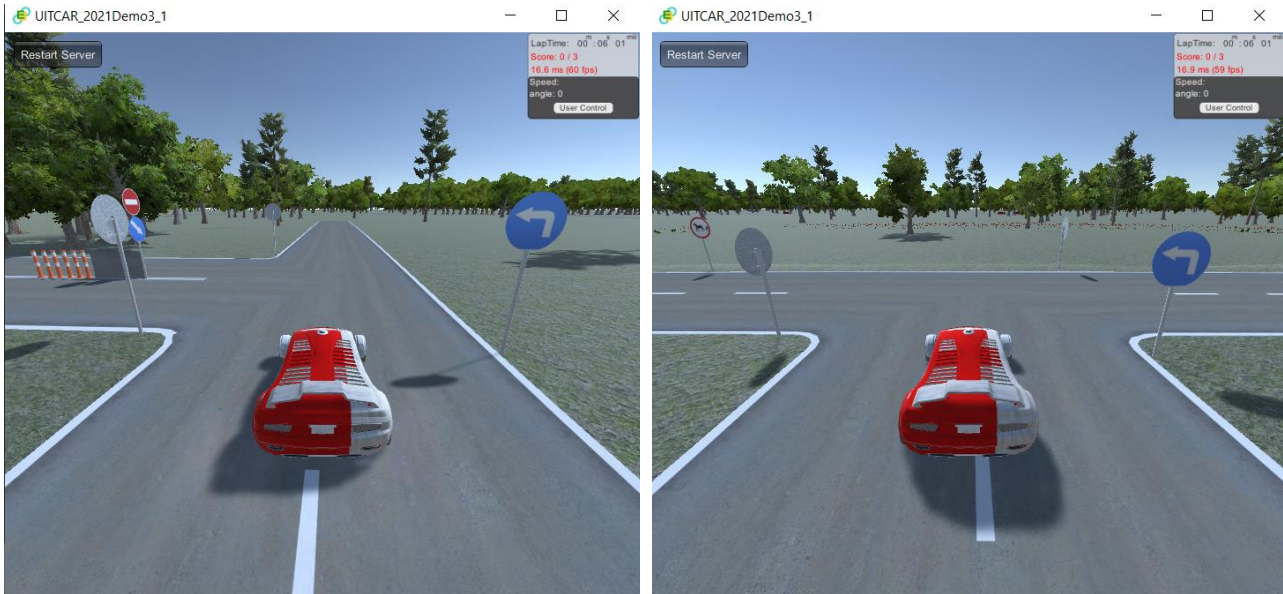
Ý tưởng: Khi xe đi vào đường này thì ta thấy điểm Max sẽ nhỏ hơn một nửa khung hình ($\text{image.shape}[1]/2$) => Nên lúc này ta sẽ cho xe bám theo lane trái + Độ rộng đường/2 (hình 3.4)



Hình 3.4. Cách xử lý đường background ra làn đường

2. Vấn đề 2: Rẽ trái

Do rút kinh nghiệm từ vòng bán kết, nhóm đã thêm trường hợp vị trí để bắt đầu tính khoảng cách (chọn khi xe vừa vào giao lộ (hình 3.5)) để giảm nhiễu => giúp xe có thể rẽ trái mọi loại đường



Hình 3.5. Vị trí tính khoảng cách của

3. Vấn đề 3: Xử lý vật cản

Dựa vào Bounding Box trả về từ Yolov5, nhóm có thể biến được vị trí vật cản (car) nằm bên trái hay bên phải và từ đó ta sẽ xử lý theo từng trường hợp

Sẽ phải thêm Timer bám lane vì sau khi tầm nhìn không còn nhận được vật cản (car) thì xe sẽ chạy mode chạy thẳng

3.1. Đường 1 làn

- Nếu vật cản (car) nằm bên phải => cho xe bám lane trái + $y(\text{pixel})$ ($y \sim 5-10 \text{ pixel}$) (hình 3.6a)
- Nếu vật cản (car) nằm bên trái => cho xe bám lane phải - $y(\text{pixel})$ ($y \sim 5-10 \text{ pixel}$) (hình 3.6b)



a. Vật cản (car) bên phải



b. Vật cản (car) bên trái

Hình 3.6. Đường một làn

3.2 Đường 2 làn

- Nếu vật cản (car) nằm bên phải thì ta sẽ cho xe bám lane trái + $y(\text{pixel})$ ($y \sim 25-20$ pixel do đường lớn) (hình 3.7a)
- Nếu vật cản (car) nằm bên trái \Rightarrow không cần xử lý do lúc này xe đã chạy bên phải (hình 3.7b)



a. Vật cản (car) bên phải



b. Vật cản (car) bên trái

Hình 3.7. Đường hai làn

Lưu ý và giải pháp khi thu thập data

- **Lưu ý**

- Dữ liệu data quá ít. Nếu dữ liệu quá ít thì độ chính xác của model sẽ giảm, hoặc nếu độ chính xác cao thì khả năng model chỉ nhận diện tốt trên ảnh đã train chứ không nhận diện tốt trên ảnh khác.
- Có nhiều dữ liệu bị trùng. Việc có nhiều ảnh giống nhau sẽ làm cho model nhận diện tốt hơn khi gặp ảnh đó và nhận diện kém hơn khi gặp các ảnh khác.
- Mất cân bằng dữ liệu. Việc này có thể khiến model nhận diện biển báo rẽ trái tốt hơn nhận diện biển báo rẽ phải và có thể bị nhầm lẫn giữa 2 biển báo do dữ liệu biển báo kia nhiều hơn.

- **Giải pháp**

- Nếu dữ liệu quá ít ta có thể sử dụng Augmentation dữ liệu để dữ liệu được phong phú hơn. Nếu chưa quen việc augmentation bạn có thể sử dụng trang web [Roboflow](https://roboflow.com/) để làm phong phú dữ liệu. Đây là trang web khá tốt nếu bạn muốn train thử model trên Google Colab.
- Nếu dữ liệu bị trùng thì bạn nên xóa bớt các dữ liệu trùng nhau vì việc Augmentaton xong bạn sẽ tăng bộ dữ liệu của bạn lên rất đáng kể nên không cần quá quan tâm về việc thiếu dữ liệu.
- Về việc mất cân bằng dữ liệu giữa các class thì tốt nhất bạn nên tách riêng thư mục từng class và kiểm tra xem các class có cân bằng với nhau không. Có thể chấp nhận được nếu dữ liệu chênh lệch các class < 1.5 .

Link tham khảo:

1. **Vòng sơ loại:**

Phương pháp xử lý ảnh thuần:

<https://www.youtube.com/watch?v=eLTLtUVuuy4&t=3721s>

Phương pháp Segmentation (Unet):

<https://www.youtube.com/watch?v=yrGeyfQUBso&list=PLdDI53OVr0ENncpqMkXyCg4k4Lldvhw3H&index=2>

Thuật toán điều khiển:

<https://www.youtube.com/watch?v=uDkqiuMYZm8&list=PLdDI53OVr0ENncpqMkXyCg4k4Lldvhw3H>

Bài thi (1:34:00):

<https://www.youtube.com/watch?v=imc1yLvz91g&list=PLdDI53OVr0ENncpqMkXyCg4k4Lldvhw3H&index=3>

2. Vòng bán kết:

Yolov5: <https://blog.roboflow.com/how-to-train-yolov5-on-a-custom-dataset/>

Bài thi (1:04:00):

<https://www.youtube.com/watch?v=Th0XqO1MvJU&list=PLdDI53OVr0ENncpqMkXyCg4k4Lldvhw3H&index=4>

3. Vòng chung kết:

CNN – Anh Phú:

<https://www.facebook.com/groups/1015789475526341/posts/1252569808514972/>

Ý tưởng xử lý ngã 3, ngã 4 – thầy Hoàng:

<https://www.youtube.com/watch?v=KdbbiiFTfn4&list=PLxat3wSWURMVvCyuJoxP2Gjd-Pg0cqRc7&index=47>

Bài thi (46:45):

<https://www.youtube.com/watch?v=yspJeNuXQY8&list=PLdDI53OVr0ENncpqMkXyCg4k4Lldvhw3H&index=5>

*****UTE_AI Lab*****

Nhóm cảm ơn các bạn đã đọc bài viết của nhóm. Nếu các bạn nào có ý tưởng hay hơn và mới lạ hơn thì có thể đóng góp ý kiến để nhóm mình phát triển hơn. Cuối cùng cảm ơn các bạn đã bỏ chút ít thời gian để đọc bài viết của mình ạ!

