

VIETNAM NATIONAL UNIVERSITY - HO CHI MINH

UNIVERSITY OF SCIENCE



## BÁO CÁO ĐỒ ÁN CUỐI KÌ

COMPUTER VISION IN SELF-DRIVING CAR

THỊ GIÁC MÁY TÍNH TRONG LĨNH VỰC XE TỰ LÁI

Môn học: Chuyên Đề Thị Giác Máy Tính

Lớp: 21TGMT

Giảng viên: Võ Hoài Việt

Phạm Minh Hoàng

Sinh viên: Lê Nguyễn Phương Uyên - 21127476

Đoàn Việt Hưng - 21127289

Email: lnpuyen21@clc.fitus.edu.vn

dvhung21@clc.fitus.edu.vn

Ho Chi Minh City - 2024

# MỤC LỤC

<b>1) GIỚI THIỆU</b>	<b>3</b>
1.1 Định nghĩa công nghệ tự lái và xe tự lái: . . . . .	3
1.1.1 <u>Thế nào là tự lái ?</u> . . . . .	3
1.1.2 <u>Thế nào là xe tự lái ?</u> . . . . .	3
1.2 Tầm quan trọng chung của thị giác máy tính (CV) trong xe tự lái[1]: . . . . .	3
1.3 Mục tiêu của báo cáo: . . . . .	4
<b>2) CÁC CẤP ĐỘ CỦA XE TỰ LÁI[2]</b>	<b>4</b>
<b>3) HIỆN TRẠNG CỦA CÔNG NGHỆ TỰ LÁI</b>	<b>8</b>
3.1 Xe Tự Lái Hiện Nay Có Thật Sự Tự Lái Không ? . . . . .	8
3.2 Khả Năng Hiện Tại: Xe Mức 2 và 3 Vẫn Cần Giám Sát . . . . .	9
<b>4) TỔNG QUAN CÁC MÔ HÌNH HỌC SÂU</b>	<b>9</b>
4.1 Mạng Nơ-ron Tích Chập Sâu (CNN): . . . . .	9
4.2 Mạng Nơ-ron Hồi Quy (RNN): . . . . .	10
4.3 Học Tăng Cường Sâu (DRL): . . . . .	11
<b>5) XU HƯỚNG THỊ TRƯỜNG VÀ CÔNG NGHỆ</b>	<b>11</b>
<b>6) CÁC THÀNH PHẦN CHÍNH CỦA XE TỰ LÁI</b>	<b>15</b>
6.1 Cảm biến phần cứng: Camera vs. LiDAR[24]: . . . . .	16
6.2 Nhận diện làn đường: . . . . .	17
6.3 Nhận diện vật thể: . . . . .	22
6.4 Phân đoạn ngữ nghĩa: . . . . .	25
6.5 Lập kế hoạch đường đi cho xe tự hành: . . . . .	28
6.6 Học bắt chước và tăng cường học sâu: . . . . .	29
6.7 Ví dụ chi tiết trong TESLA: . . . . .	30
6.7.1 Thu thập và gán nhãn dữ liệu: . . . . .	30

6.7.2 Quy trình trong mạng học sâu của Tesla - Full Self-Driving (FSD)[33][34][35]: . . . . .	31
<b>7) THỰC NGHIỆM VÀ ĐÁNH GIÁ</b>	<b>46</b>
7.1 Dataset: . . . . .	46
7.2 Các tác vụ: . . . . .	47
7.2.1 Thực nghiệm nhận diện làn đường: . . . . .	47
7.2.2 Thực nghiệm nhận diện, phân loại và truy vết vật thể: . . . . .	48
7.2.3 Thực nghiệm phân đoạn ngữ nghĩa: . . . . .	50
<b>8) AN TOÀN VÀ ĐỘ TIN CẬY</b>	<b>52</b>
8.1 Xử Lý Lỗi Hệ Thống: Cơ Chế và Thách Thức: . . . . .	53
8.2 Phản Ứng Khi Xảy Ra Lỗi: Chế Độ An Toàn và Giám Sát Thời Gian Thực: . . . . .	53
8.3 Các Phương Pháp Khắc Phục: Cập Nhật OTA và Thu Hồi: . . . . .	53
8.4 Tổng kết: . . . . .	54
<b>9) THÁCH THỨC VÀ HẠN CHẾ CỦA THỊ GIÁC MÁY TÍNH TRONG XE TỰ LÁI</b>	<b>54</b>
<b>10 KẾT LUẬN VÀ DỰ ĐOÁN</b>	<b>56</b>
10.1 Tổng hợp: . . . . .	56
10.2 Dự đoán: . . . . .	57
10.3 Kết luận: . . . . .	58

# 1) GIỚI THIỆU

- Công nghệ xe tự lái đang trở thành một trong những bước tiến quan trọng và đầy tiềm năng của ngành công nghiệp ô tô, mở ra viễn cảnh thay đổi cách con người di chuyển và tổ chức giao thông trong tương lai. Với sự kết hợp của các công nghệ tiên tiến, xe tự lái không chỉ mang lại sự tiện lợi mà còn hướng tới việc nâng cao an toàn và hiệu quả trên các cung đường. Để hiểu rõ hơn về lĩnh vực này, chúng ta cần bắt đầu từ những khái niệm cơ bản.

## 1.1 Định nghĩa công nghệ tự lái và xe tự lái:

### 1.1.1 Thế nào là tự lái ?

- Tự lái, hay tự động lái, là khả năng của một phương tiện vận hành mà không cần sự can thiệp của con người, sử dụng các cảm biến như camera, radar, và phần mềm để định hướng, nhận thức môi trường, tránh chướng ngại vật, tuân thủ luật giao thông và đưa ra quyết định.

### 1.1.2 Thế nào là xe tự lái ?

- Một chiếc xe tự lái, còn được gọi là xe tự trị (Autonomous Car - AC), xe không người lái, robotaxi, xe robot hoặc robo-car, là một chiếc xe có khả năng hoạt động độc lập mà gần như không/không hề có sự tác động của con người. Xe tự lái chịu trách nhiệm cho tất cả các hoạt động chẳng hạn như nhận thức môi trường, giám sát các hệ thống quan trọng và điều khiển phương tiện, bao gồm điều hướng từ nơi bắt đầu cho đến đích.

## 1.2 Tầm quan trọng chung của thị giác máy tính (CV) trong xe tự lái[1]:

- Có thể nói rằng, CV hoạt động như đôi mắt của xe. Nó sử dụng máy ảnh và cảm biến để chụp ảnh và video về mọi thứ xung quanh xe. Điều này bao gồm các tuyến đường, đèn giao thông, con người và những chiếc xe khác. Chiếc xe sau đó sử dụng các kỹ thuật đặc biệt để hiểu những hình ảnh và video này.

- Còn Học máy (Machine Learning - ML) giống như não của xe. Nó nhìn vào dữ liệu từ các máy ảnh và cảm biến. Sau đó sử dụng các thuật toán đặc biệt để tìm các mẫu, đưa ra dự đoán và học hỏi từ thông tin mới. Điều này giúp chiếc xe đưa ra quyết định thông minh, xử lý các tình huống mới và trở nên tốt hơn theo thời gian.

Bảng 1: Tóm tắt đóng góp của thị giác máy tính

Chức năng	Mô tả
Phát hiện và phân loại vật thể	Nhận diện xe cộ, người đi bộ, biển báo giao thông, giúp đưa ra quyết định.

Chức năng	Mô tả
Phát hiện và giữ làn đường	Giúp xe tự động giữ đúng làn, giảm nguy cơ va chạm.
Ước lượng khoảng cách	Sử dụng tầm nhìn stereo để xác định khoảng cách, hỗ trợ tránh va chạm.
Hiểu môi trường	Phân tích các tình huống phức tạp, như khu vực thi công hoặc thời tiết xấu.
An toàn và dự phòng	Cung cấp lớp an toàn bổ sung, kết hợp với lidar và radar.

### 1.3 Mục tiêu của báo cáo:

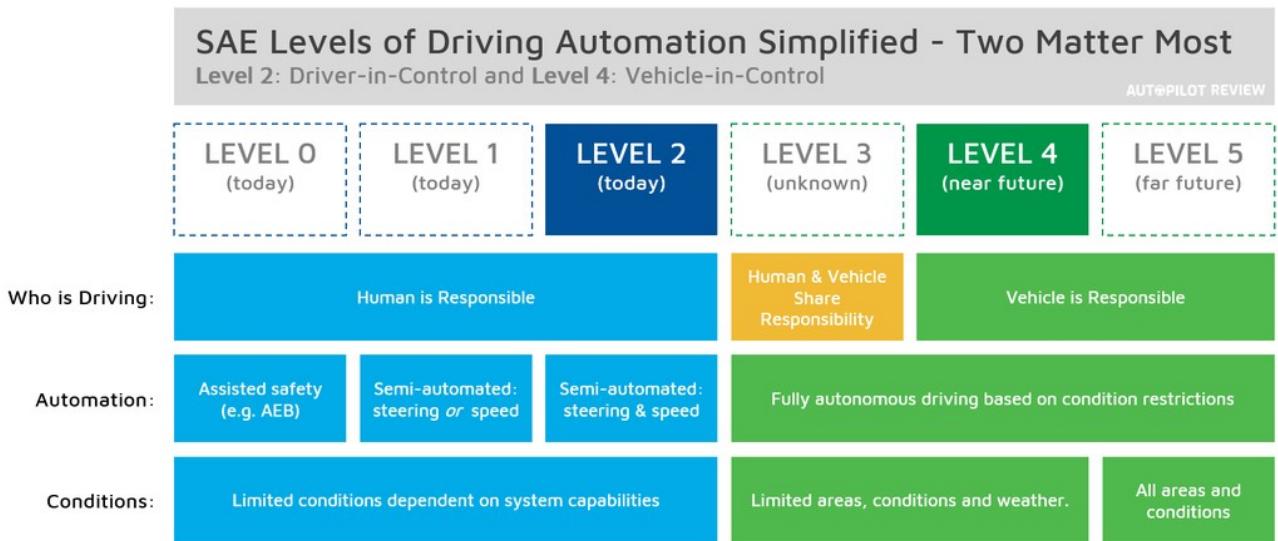
- Báo cáo này sẽ cung cấp cái nhìn toàn diện về computer vision trong xe tự lái, bao gồm các định nghĩa quan trọng, các cấp độ tự động hóa, các mô hình Deep Learning quan trọng/phổ biến trong xe tự lái, xu hướng thị trường và những khó khăn, thách thức mà chúng ta đã và đang gặp phải (điều kiện thời tiết, những tình huống bất ngờ khi lưu thông trên đường hay lỗi phần mềm,...) và phương pháp/những đề suất trong tương lai để giải quyết các khó khăn này.

## 2) CÁC CẤP ĐỘ CỦA XE TỰ LÁI[2]

- Tính đến năm 2019, hầu hết các xe ô tô vẫn chưa có hệ thống lái xe tự động, mặc dù nhiều xe có hệ thống hỗ trợ người lái tiên tiến, chẳng hạn như phanh khẩn cấp tự động và kiểm soát hành trình.

- Tuy nhiên, tính tự chủ hoàn toàn không còn quá xa vời nữa, và một số công ty, chẳng hạn như Tesla và Waymo, đã cho ra mắt những chiếc xe có tính năng tự hành đáng kể. Các rào cản về công nghệ, văn hóa và pháp lý đã khiến quá trình chuyển đổi sang xe không người lái trở nên khó dự đoán, vì vậy các dự đoán về tự động hóa vẫn chưa chắc chắn. Tuy nhiên, sáu cấp độ lái xe của SAE đóng vai trò là lược đồ để suy nghĩ về hướng đi của công nghệ này và chúng ta có thể sử dụng nó để phân loại khi cần. SAE coi mỗi cấp độ là chuẩn mực cho phạm vi nhiệm vụ lái xe mà hệ thống lái xe tự động có thể đạt được, từ nhập làn đường đến lái xe tốc độ cao.

- Mỗi cấp độ có số từ 0 đến 5, với số cao hơn biểu thị mức độ tự hành tăng lên trong xe. Những con số này có thể xuất hiện trên tài liệu tiếp thị của công ty, trong một bài báo phổ biến hoặc trong thông cáo báo chí về sản phẩm mới của một công ty ô tô.



Hình 1: Các cấp độ SAE của xe tự lái

**a) Level 0: Không tự động (No Automation):**

- Phương tiện không có hệ thống tự động lái, mặc dù chúng có thể có một số tính năng tự động. Mặc dù người lái sẽ luôn kiểm soát hoàn toàn chiếc xe, chiếc xe có thể có một số khả năng như đưa ra cảnh báo an toàn (ví dụ: hệ thống cảnh báo lệch khỏi làn đường) hoặc ngay lập tức can thiệp vào hệ thống xe (ví dụ: phanh khẩn cấp tự động).

**b) Level 1: Hỗ trợ lái xe (Driver Assistance) - "hands-on":**

- Xe cấp độ 1 có các cơ chế hỗ trợ đánh lái hoặc tăng tốc, mặc dù không phải cả hai cùng lúc. Người lái phải chuyển từ chức năng này sang chức năng khác. Đôi khi được gọi là cấp độ "hands-on", vì người lái xe được yêu cầu phải giữ tay trên vô lăng, cấp độ tự động hóa này chia quyền kiểm soát xe giữa người lái và phần mềm.

- Adaptive Cruise Control (ACC) là một tính năng phổ biến của xe tự lái cấp độ 1, là một hệ thống hỗ trợ lái xe giúp duy trì tốc độ đã đặt và khoảng cách an toàn với xe phía trước. Hệ thống này sử dụng radar hoặc lidar để theo dõi khoảng cách và tốc độ của xe phía trước. Nếu xe phía trước chậm lại, ACC sẽ tự động giảm tốc độ của xe bạn để duy trì khoảng cách an toàn. Khi xe phía trước tăng tốc hoặc chuyển làn, ACC sẽ tăng tốc trở lại tốc độ đã đặt. Tuy nhiên, điều quan trọng cần lưu ý là ACC ở cấp độ 1 chỉ là một tính năng hỗ trợ. Người lái xe vẫn phải chịu trách nhiệm hoàn toàn trong việc điều khiển xe và phải luôn sẵn sàng can thiệp khi cần thiết. Hệ thống này không thể tự động xử lý tất cả các tình huống giao thông và có thể không hoạt động tốt trong điều kiện thời tiết xấu hoặc giao thông đông đúc.

- Ngoài ra, xe tự hành cấp độ 1 cũng có thể bao gồm hỗ trợ giữ/chuyển làn đường, cảnh báo xe về việc xe khác di chuyển vào và ra khỏi làn đường; phanh khẩn cấp tự động, cảnh báo người lái xe

về vụ va chạm và cho phép phanh hết công suất; và hỗ trợ đỗ xe. Nhiều xe Cấp độ 1 có chức năng phát hiện mép đường, cho người lái biết nếu họ ở quá gần lề đường.

c) **Tự động hóa 1 phần (Partial Automation) - "pay attention":**

- Đây là cấp độ phổ biến nhất trong các xe ô tô dành cho người tiêu dùng có chế độ lái tự động ngày nay và mô tả các loại xe có thể kiểm soát các tính năng lái và tốc độ. Tuy nhiên, người lái xe vẫn chịu trách nhiệm về xe và phải luôn chú ý, ngay cả khi Hệ thống giám sát người lái (DMS) cho phép người lái xe "rảnh tay" với vô lăng.

- Xe tự động có thể điều khiển cả tốc độ và đánh lái cùng một lúc, nhưng chỉ trong một số điều kiện nhất định, chẳng hạn như dưới giới hạn tốc độ được đặt. Một chiếc xe tự động ở cấp 2 có thể có cả công nghệ ACC và công nghệ xác định tâm làn đường và cả 2 hoạt động song song. Một ví dụ khác là hỗ trợ đỗ xe (hoặc tự đỗ xe). Người lái xe cần phải cảnh giác hoàn toàn, theo dõi môi trường xung quanh (về cơ bản là giám sát hệ thống lái xe tự động) và sẵn sàng tiếp quản việc lái xe nếu cần.

d) **Tự động hóa có điều kiện (Conditional Automation) - "eyes off":**

- Cấp độ 3 của hệ thống phân loại SAE là cấp độ gây tranh cãi và tranh luận nhiều nhất vì cho phép người lái xe tập trung vào những việc khác ngoài việc lái xe khi đang ở trong xe, nhưng vẫn phải chịu trách nhiệm cuối cùng.

- Với công nghệ ở cấp độ 3, xe không cần ta phải giữ tay trên vô lăng hay mắt phải tập trung nhìn đường. Ta có thể đọc sách, xem phim hoặc chơi điện thoại.

- Xe có thể điều khiển cả tốc độ và hướng lái cùng một lúc và có thể theo dõi môi trường xung quanh, vì vậy nó có thể tự lái trong một số điều kiện nhất định. Hơn nữa, xe có thể tự phanh khẩn cấp và chúng đặc biệt hiệu quả trên đường cao tốc, nơi mà người đi bộ và ngã tư không phải là vấn đề đáng chú ý. Xe cấp độ 3 thường có cảnh báo bằng âm thanh để cảnh báo người lái xe kiểm soát xe. Nếu người lái xe không phản hồi, xe có thể tiếp tục lái xe cho đến khi người lái xe tiếp tục lái. Yêu cầu duy nhất là ta phải ở trong xe. Người lái vẫn cần phải chú ý, nhưng thường thì xe sẽ cảnh báo người lái rằng họ cần kiểm soát lại tay lái vì chiếc xe không còn trong điều kiện mà nó có thể tự xử lý.

- Mức độ này gây nhiều tranh cãi vì người lái xe phải có mặt và chịu trách nhiệm về phương tiện mặc dù phương tiện chủ yếu nằm trong tầm kiểm soát và người lái xe có thể không nhận thức được tình hình mọi lúc. Nếu có vấn đề, điều này sẽ khiến người lái xe rất khó để mà đột nhiên lấy lại nhận thức về tình huống và bối cảnh của tình huống khi thường phải đưa ra quyết định trong tích tắc.

- Do đó, hầu hết các nhà sản xuất có thể sẽ bỏ qua Cấp độ 3 này và làm cho việc tự lái trở nên rõ ràng hơn, với người lái xe chịu trách nhiệm và kiểm soát (Cấp độ 0, 1, 2) hoặc không (Cấp độ 4 và

5). Những nhà sản xuất khác, như Mobileye, đang cố gắng cải thiện Cấp độ 2, gọi công nghệ của họ là “Cấp độ 2+”, có khả năng gây thêm nhầm lẫn.

e) **Level 4: Tự động hóa cao (High Automation) - "mind off":**

- Cấp độ 4 của hệ thống phân loại SAE bao gồm các phương tiện không cần bất kỳ tương tác nào của con người, ngay cả khi nói đến vấn đề an toàn. Chúng có thể tự thực hiện mọi chức năng lái xe và có thể xử lý hầu hết các điều kiện lái xe bình thường, cho phép người lái thư giãn hoặc ngủ.

- Nếu chiếc xe ở trong tình huống mà hệ thống tự động không thể lái (như thời tiết khắc nghiệt), nó sẽ thực hiện các biện pháp an toàn như buộc người lái phải trực tiếp điều khiển trở lại hoặc tự động đỡ xe ở đâu đó nếu người lái xe từ chối tiếp quản. Ở cấp độ 4, người lái có thể giành lại quyền kiểm soát chiếc xe bất cứ khi nào họ muốn. Ngoài ra, những chiếc xe cấp độ 4 có thể xử lý các công trường xây dựng đường bộ và các tình huống khó khăn khác miễn là chúng phù hợp với các thông số vận hành của phần mềm xe.

f) **Level 5: Hoàn toàn tự động (Full Automation) - "chauffeured":**

- Tự động hóa hoàn toàn là mục tiêu cuối cùng của hầu hết các công ty xe tự lái. Ở cấp độ này, sự can thiệp của con người là không cần thiết, ngay cả trong những môi trường đầy thách thức nhất, chẳng hạn như đường đất. Tóm lại, không có tài xế trong những chiếc xe cấp độ 5, chỉ có hành khách. Tuy nhiên, tài xế vẫn có thể tiếp quản lại bất cứ khi nào họ muốn.

- Ở những dạng tiên tiến hơn, những chiếc xe như vậy có thể giống như phòng khách di động, với chỗ ngồi thoải mái tận dụng tối đa nội thất và chúng có thể bao gồm các tiện nghi như máy tính cá nhân và tủ lạnh. Cuối cùng, những chiếc xe như vậy thậm chí có thể có giường gấp cho phép hành khách ngủ trong những chuyến đi đường dài hoặc ngủ trưa trên đường về nhà sau giờ làm việc. Vì xe cấp độ 5 không cần giấy phép nên bất kỳ ai cũng có thể lái thử.

- Những người khuyết tật khiến cho việc lái xe trở nên khó khăn hoặc không thể, chẳng hạn như mù hoặc liệt, được hưởng lợi nhiều nhất từ việc tự động hóa hoàn toàn. Mặc dù xe chở khách thương mại ở cấp độ này vẫn chưa lăn bánh trên đường, nhưng xe tải giao hàng tạp hóa cấp độ 5 của Nuro đã lăn bánh. Chiếc xe tải này giao đồ ăn đến tận nhà bạn và lái trở lại bãi đỗ xe sau khi hoàn thành. Ngoài ra, Waymo hiện đang phát triển một đội xe gồm 600 xe hybrid Chrysler Pacifica cấp độ 5 và BMW đang hợp tác với Mobileye để đưa chiếc xe tự lái hoàn toàn đầu tiên vào sản xuất hàng loạt.

- Những chiếc xe Cấp độ 5 đầu tiên có thể chỉ giới hạn ở các khu vực đô thị có nhịp độ chậm trong khi các công ty đang hoàn thiện những khâu cuối cùng của công nghệ và xây dựng thư viện dữ liệu về hiệu suất thực tế.

Bảng 2: Tổng quan các cấp độ của xe tự lái

	<b>Level 0</b>	<b>Level 1</b>	<b>Level 2</b>	<b>Level 3</b>	<b>Level 4</b>	<b>Level 5</b>
<b>Cấp độ tự động</b>	Không	Hỗ trợ tài xế	Một phần	Có điều kiện	Cao	Hoàn toàn
<b>Yêu cầu người lái ?</b>	Có	Có	Có	Có	Không	Không
<b>Điều khiển tốc độ ?</b>	Không	Có, nhưng không cùng lúc với bánh lái	Có, trong một số điều kiện nhất định	Có, trong một số điều kiện nhất định	Có, trong một số điều kiện nhất định	Có
<b>Điều khiển bánh lái ?</b>	Không	Có, trong một số điều kiện nhất định	Có, trong một số điều kiện nhất định	Có, trong một số điều kiện nhất định	Có, trong một số điều kiện nhất định	Có
<b>Nhận thức được môi trường xung quanh ?</b>	Không	Không	Không	Không	Không	Có
<b>Hành động khi gặp điều kiện không thể lái xe (ví dụ: thời tiết xấu)</b>	Người lái tự kiểm soát	Người lái tự kiểm soát	Người lái xe tự quyết định, nếu điều kiện không phù hợp thì kiểm soát lại	Hệ thống sẽ cảnh báo người lái lấy lại quyền kiểm soát	Thêm các thủ tục an toàn	Tự lái được trong mọi điều kiện

### 3) HIỆN TRẠNG CỦA CÔNG NGHỆ TỰ LÁI

#### 3.1 Xe Tự Lái Hiện Nay Có Thật Sự Tự Lái Không ?

- Câu hỏi liệu xe tự lái hiện nay có thực sự tự lái hoàn toàn không có thể được trả lời bằng cách xem xét các mức độ tự động hóa và nghiên cứu cho thấy xe tự lái hiện nay không thực sự tự lái hoàn toàn. Tính đến tháng 2 năm 2025, xe mức 3 và trên đó vẫn hiếm, với chỉ một số xe như Honda ở Nhật Bản và Mercedes-Benz ở Đức, California, Nevada đạt mức 3[3]. Mức 3 cho phép xe tự xử lý hầu hết tác vụ lái trong điều kiện nhất định, nhưng tài xế vẫn phải sẵn sàng can thiệp khi hệ thống yêu cầu. Điều này cho

thấy xe tự lái hiện nay không đạt mức 5, tức là tự lái hoàn toàn, mà vẫn cần sự giám sát của con người.

- Một chi tiết bất ngờ là, mặc dù nhiều công ty như Tesla quảng bá hệ thống "Full Self-Driving", thực tế, theo TechCrunch[4], hệ thống này chỉ ở mức 2 hoặc 3, và Ford đã từ bỏ phát triển mức 4, tập trung vào hệ thống hỗ trợ lái nâng cao. Điều này cho thấy sự khác biệt lớn giữa quảng cáo và thực tế, làm dấy lên tranh cãi về mức độ tự lái thực sự của các hệ thống hiện tại.

### 3.2 Khả Năng Hiện Tại: Xe Mức 2 và 3 Vẫn Cần Giám Sát

- Mức 2 (Tự động hóa một phần): Xe ở mức 2 có thể đồng thời kiểm soát tốc độ và lái, như với tính năng Cruise Control thích ứng và giữ làn đường, nhưng người lái phải luôn chú ý và sẵn sàng can thiệp. Ví dụ, Tesla Autopilot (ACV Auctions), là một hệ thống mức 2, yêu cầu người lái luôn sẵn sàng, với các cảm biến như camera và radar hỗ trợ nhưng không thay thế hoàn toàn người lái.

- Mức 3 (Tự động hóa có điều kiện): Xe ở mức 3, như hệ thống Drive Pilot của Mercedes-Benz, cho phép tài xế thả tay khỏi vô-lăng trong một số tình huống, chẳng hạn như trên đường cao tốc với thời tiết tốt, nhưng vẫn cần tài xế sẵn sàng nhận lại quyền kiểm soát khi hệ thống yêu cầu[5]. Chỉ một số thị trường, như Nhật Bản và một số bang ở Mỹ, cho phép sử dụng mức 3, với các hạn chế về điều kiện đường xá.

- Tóm lại, xe tự lái hiện nay, chủ yếu ở mức 2 và 3, không thực sự tự lái hoàn toàn và vẫn cần sự giám sát của con người. Mặc dù có những tiến bộ đáng kể, như cải tiến cảm biến và AI, công nghệ vẫn chưa đủ để xử lý mọi tình huống giao thông phức tạp một cách độc lập. Các nghiên cứu và thử nghiệm cho thấy cần thêm thời gian để phát triển và xác nhận an toàn trước khi xe mức 4 hoặc 5 trở thành hiện thực cho người tiêu dùng.

## 4) TỔNG QUAN CÁC MÔ HÌNH HỌC SÂU

- Phần này trình bày các nguyên lý cơ bản của công nghệ học sâu áp dụng trong xe tự hành, đồng thời đánh giá khả năng của từng phương pháp. Các kỹ thuật chính bao gồm Mạng Nơ-ron Tích Chập (CNN), Mạng Nơ-ron Hồi Quy (RNN) và Học Tăng Cường Sâu (DRL), vốn được sử dụng phổ biến trong lĩnh vực này.

- Trong suốt nội dung, các ký hiệu được sử dụng để biểu diễn dữ liệu chuỗi thời gian. Một biến có thể được xác định tại thời điểm rời rạc  $t$ , ký hiệu là  $< t >$ , hoặc trong một khoảng thời gian  $< t, t+k >$ , trong đó  $k$  là số bước thời gian. Ví dụ, trạng thái  $z$  tại thời điểm  $t$  có thể viết dưới dạng  $z^{<t>}$ , còn chuỗi giá trị của nó trong khoảng thời gian  $k$  là  $z^{<t,t+k>}$ . Các vectơ và ma trận được ký hiệu bằng chữ in đậm.

### 4.1 Mạng Nơ-ron Tích Chập Sâu (CNN):

- CNN là một trong những phương pháp hiệu quả nhất để xử lý dữ liệu không gian như hình ảnh. Khác với các phương pháp truyền thống, CNN có khả năng học trực tiếp đặc trưng từ dữ liệu mà không

cần thiết kế thủ công.

- Mạng CNN bao gồm tập hợp trọng số  $\theta = [W, b]$ , trong đó  $W$  là ma trận trọng số kết nối giữa các nơ-ron, còn  $b$  là hệ số bù. Các trọng số  $W$  được tổ chức dưới dạng bộ lọc và học được qua quá trình huấn luyện. Quá trình truyền thông tin trong CNN tuân theo công thức:

$$M^{k+1,l} = \phi(M^k * w^{k,l} + b^{k,l}) \quad (1)$$

trong đó:

- $M^k$  là đầu vào của lớp  $k$
- $w^{k,l}$  là bộ lọc tích chập
- $b^{k,l}$  là hệ số bù
- $*$  là phép tích chập
- $\phi(\cdot)$  là hàm kích hoạt, thường là ReLU

- Trong học có giám sát, CNN có thể được huấn luyện bằng phương pháp tối ưu hóa, chẳng hạn như ước lượng hợp lý cực đại (MLE). Nếu bài toán là hồi quy, ta có thể sử dụng hàm mất mát bình phương tối thiểu:

$$\hat{\theta} = \arg \min_{\theta} \sum_{i=1}^m (R(x_i; \theta) - y_i)^2 \quad (2)$$

- Còn đối với bài toán phân loại, hàm mất mát phổ biến là entropy chéo hoặc log-likelihood.

## 4.2 Mạng Nơ-ron Hồi Quy (RNN):

- RNN là một mô hình học sâu được thiết kế để xử lý dữ liệu chuỗi thời gian. Điểm khác biệt của RNN so với các mạng nơ-ron thông thường là nó có cơ chế hồi quy, giúp lưu trữ và xử lý thông tin từ các bước thời gian trước đó.

- LSTM là một phiên bản nâng cấp của RNN nhằm giải quyết vấn đề mất dần gradient bằng cách sử dụng ba cổng điều khiển: cổng đầu vào, cổng quên và cổng đầu ra. Cách cập nhật trạng thái bộ nhớ trong LSTM được tính như sau:

$$c^{<t>} = \Gamma_u^{<t>} * \tanh(W_c s^{<t>} + U_c h^{<t-1>} + b_c) + \Gamma_f^{<t>} * c^{<t-1>} \quad (3)$$

$$h^{<t>} = \Gamma_o^{<t>} * \tanh(c^{<t>}) \quad (4)$$

- LSTM đã được ứng dụng rộng rãi trong các mô hình xử lý chuỗi thời gian, bao gồm cả phân tích dữ liệu cảm biến trong xe tự hành.

### **4.3 Học Tăng Cường Sâu (DRL):**

- Học tăng cường sâu (DRL) là một phương pháp kết hợp giữa học sâu và học tăng cường, cho phép hệ thống học cách điều hướng môi trường bằng cách tối ưu hóa hành động để nhận được phần thưởng cao nhất.

- Trong bối cảnh xe tự hành, bài toán điều khiển có thể được mô hình hóa như một Quy trình Quyết định Markov Ân (POMDP), trong đó xe tự hành là tác nhân thực hiện hành động dựa trên trạng thái môi trường và nhận phần thưởng để điều chỉnh chiến lược lái. Mục tiêu của DRL là tìm ra chính sách tối ưu  $\pi^*$  giúp tối đa hóa phần thưởng tích lũy.

- Một phương pháp phổ biến trong DRL là Deep Q-Network (DQN), sử dụng mạng nơ-ron sâu để xấp xỉ hàm giá trị  $Q^*(s, a)$ , giúp xe tự hành học được chiến lược di chuyển an toàn và hiệu quả.

## **5) XU HƯỚNG THỊ TRƯỜNG VÀ CÔNG NGHỆ**

- Năm 2025, thị trường xe tự lái đang phát triển mạnh mẽ, với các thương hiệu lớn áp dụng công nghệ tiên tiến để cải thiện khả năng tự lái. Phần 7 sẽ là cái nhìn tổng quan về các thương hiệu được đề cập và công nghệ của họ:

### **a) Tesla:**

- Tesla đã nổi lên như một trong những công ty tiên phong trong lĩnh vực công nghệ xe tự lái, với một hành trình phát triển đầy tham vọng và những bước tiến đáng chú ý. Khác với nhiều đối thủ trong ngành, Tesla theo đuổi một cách tiếp cận độc đáo, tập trung chủ yếu vào hệ thống thị giác (vision-centric) để đạt được khả năng tự lái. Báo cáo này sẽ phân tích chi tiết các công nghệ xe tự lái mà Tesla đang phát triển và sử dụng, đặc biệt là hệ thống Full Self-Driving (FSD), đồng thời so sánh chiến lược của Tesla với các công ty khác như Waymo, công ty nổi tiếng với phương pháp tiếp cận đa cảm biến. Sự khác biệt trong triết lý phát triển này, giữa việc chỉ dựa vào camera của Tesla và việc sử dụng kết hợp LiDAR, radar và camera của Waymo, đặt ra một nền tảng thú vị để phân tích và đánh giá con đường mà mỗi công ty đang theo đuổi để hiện thực hóa tương lai của giao thông tự động[6].

- Tesla trang bị cho xe của mình hai hệ thống hỗ trợ lái xe tiên tiến: Autopilot và Full Self-Driving (FSD). Autopilot là hệ thống hỗ trợ lái xe tiêu chuẩn, cung cấp các tính năng như Traffic-Aware Cruise Control (điều chỉnh tốc độ và khoảng cách an toàn với xe phía trước), Autosteering (hỗ trợ giữ làn đường), Auto Lane Change (tự động chuyển làn), Autopark (hỗ trợ đỗ xe tự động) và Summon (triệu hồi xe). Trong khi đó, Full Self-Driving (FSD) là một tùy chọn nâng cao, hướng tới mức độ tự động hóa cao hơn dưới sự giám sát của người lái. FSD (Supervised) bao gồm tất cả các tính năng của Autopilot, đồng thời bổ sung thêm Navigate on Autopilot (điều hướng tự động trên đường cao tốc), Auto Lane Change, Autopark, Summon, Autosteering on City Streets (hỗ trợ lái trên đường

phố nội đô) và Traffic and Stop Sign Control (nhận diện và phản ứng với đèn giao thông và biển báo dừng)[7].

- Cách tiếp cận của Tesla đối với công nghệ tự lái đã trải qua nhiều giai đoạn phát triển. Ban đầu, Tesla sử dụng radar và cảm biến siêu âm kết hợp với camera để thu thập thông tin về môi trường xung quanh. Tuy nhiên, công ty đã dần chuyển sang chiến lược "Tesla Vision" tập trung vào camera, loại bỏ radar và cảm biến siêu âm trên các mẫu xe mới hơn. Hiện tại, Tesla đang tập trung vào việc phát triển hệ thống FSD dựa trên học sâu "end-to-end". Để hỗ trợ cho các thuật toán AI phức tạp, Tesla đã phát triển bộ xử lý AI tùy chỉnh của riêng mình, được gọi là FSD Computer[8][9].

- Sự khác biệt rõ rệt trong chiến lược cảm biến của Tesla so với Waymo là một điểm đáng chú ý. Waymo sử dụng một bộ cảm biến đa dạng bao gồm LiDAR, radar và camera. Việc Waymo sử dụng rộng rãi LiDAR, radar và camera cho thấy niềm tin của họ vào sự dư thừa cảm biến để đảm bảo nhận thức môi trường mạnh mẽ. Ngược lại, việc Tesla nhấn mạnh vào thị giác ngũ ý sự tin tưởng vào khả năng xử lý hình ảnh tiên tiến và mạng nơ-ron để trích xuất đủ thông tin chỉ từ dữ liệu camera. Sự khác biệt cơ bản này sẽ ảnh hưởng đến tất cả các khía cạnh tiếp theo của công nghệ, từ thuật toán phần mềm đến xác thực an toàn.

#### b) Waymo:

- Waymo, trước đây là dự án xe tự lái của Google, là một công ty hàng đầu trong lĩnh vực công nghệ xe tự lái, được đổi tên vào tháng 12 năm 2016 và trở thành công ty con của Alphabet (công ty mẹ của Google). Dự án này bắt đầu vào năm 2009 tại phòng thí nghiệm Google X. Nguồn gốc từ Google X cho thấy tính chất đầy tham vọng, mang tính "đột phá" và nguồn lực đáng kể mà công ty có thể đã tiếp cận ngay từ đầu. Sự chuyển đổi từ một dự án nghiên cứu sang một công ty con độc lập cho thấy sự trưởng thành của công nghệ và sự chuyển hướng sang mục tiêu thương mại hóa, đồng thời ngũ ý một nền tảng vững chắc trong nghiên cứu và phát triển[10].

- Sứ mệnh của Waymo là làm cho việc di chuyển trở nên an toàn và dễ dàng hơn cho mọi người và mọi thứ, hướng tới cải thiện khả năng tiếp cận di chuyển cho tất cả mọi người đồng thời cứu sống hàng ngàn người bằng cách giảm thiểu tai nạn giao thông do lỗi của con người gây ra. Tuyên bố sứ mệnh này cung cấp một khuôn khổ để hiểu các lựa chọn và ưu tiên công nghệ của Waymo, đặc biệt liên quan đến an toàn và khả năng tiếp cận. Sự nhấn mạnh vào an toàn cho thấy một sự tập trung mạnh mẽ vào tính dự phòng, thử nghiệm nghiêm ngặt và xác nhận dựa trên dữ liệu của công nghệ. Công nghệ cốt lõi cho phép khả năng tự lái của Waymo bao gồm: một bộ cảm biến tiên tiến, phần mềm điều khiển bằng trí tuệ nhân tạo (AI) và bản đồ chi tiết[11].

- Các nguyên tắc và kiến trúc chung của Waymo Driver: một hệ thống tích hợp đầy đủ các cảm biến và khả năng tính toán. Waymo đã phát triển một hệ thống tích hợp duy nhất gồm các cảm biến (LiDAR, camera, radar) và máy tính trên xe (khả năng tính toán) được thiết kế để hoạt động

cùng nhau, mang lại cho Waymo Driver một cái nhìn toàn diện về thế giới xung quanh. Thuật ngữ "Waymo Driver" như một hệ thống thống nhất nhấn mạnh cách tiếp cận toàn diện đối với việc lái xe tự động, trong đó phần cứng và phần mềm được thiết kế cùng nhau để có hiệu suất tối ưu. Điều này trái ngược với các cách tiếp cận có thể dựa nhiều hơn vào các thành phần có sẵn hoặc một kiến trúc ít tích hợp hơn.

- Waymo mô tả hoạt động công nghệ lái xe tự động của mình thông qua ba bước cơ bản: "Sense, Solve, Go". Khuôn khổ này cung cấp một sự hiểu biết cao về cách Waymo Driver nhận thức, xử lý thông tin và hành động để điều hướng một cách tự động[12]:

- Sense: Waymo Driver sử dụng kết hợp LiDAR, camera và radar để cảm nhận môi trường xung quanh, có thể nhìn thấy xa tới ba sân bóng đá theo mọi hướng. Cách tiếp cận đa cảm biến này đảm bảo một hệ thống nhận thức mạnh mẽ có khả năng hoạt động trong nhiều điều kiện môi trường khác nhau.
- Solve: Hệ thống sử dụng trí tuệ nhân tạo (AI) và các thuật toán học máy để xử lý dữ liệu, xác định các đối tượng, dự đoán hành vi của chúng và tính toán một lộ trình an toàn trong thời gian thực. Giai đoạn này nhấn mạnh vai trò quan trọng của AI trong việc diễn giải lượng lớn dữ liệu cảm biến và đưa ra các quyết định lái xe thông minh. Việc đề cập đến xử lý thời gian thực làm nổi bật nhu cầu về khả năng tính toán và các thuật toán hiệu quả.
- Go: Sau khi xác định được một lộ trình an toàn, hệ thống điều khiển chuyển động tiên tiến của Waymo Driver cho phép tăng tốc và phanh mượt mà, ngay cả trong điều kiện giao thông đông đúc. Bước cuối cùng này nhấn mạnh sự kiểm soát vật lý của các bộ phận điều khiển xe dựa trên các quyết định của AI. Sự mượt mà và các chuyển động lái xe tự nhiên rất quan trọng đối với sự thoải mái và chấp nhận của hành khách.

### c) Cruise và Chuyển Hướng Công Nghệ:

- Cruise LLC, ban đầu được biết đến với tên gọi Cruise Automation, là một công ty con của General Motors (GM) tập trung vào việc phát triển và triển khai công nghệ xe tự lái. Với mục tiêu ban đầu là cách mạng hóa giao thông vận tải thông qua công nghệ không người lái tiên tiến, Cruise đã nhanh chóng trở thành một trong những công ty hàng đầu trong lĩnh vực này. Tuy nhiên, bối cảnh hiện tại cho thấy một sự thay đổi đáng chú ý trong chiến lược của Cruise, với việc tạm dừng dịch vụ taxi robot và tập trung vào việc tích hợp công nghệ tự lái vào các dòng xe cá nhân của GM.

- Hệ thống lái tự động của Cruise được xây dựng dựa trên một kiến trúc tích hợp bao gồm phần cứng (cảm biến, bộ xử lý) và phần mềm (thuật toán, hệ thống điều khiển) phối hợp nhịp nhàng để nhận thức, lập kế hoạch và hành động. Cruise đã vạch ra một quy trình làm việc gồm năm bước chính: Hiệu chuẩn Camera, Hiệu chuẩn Thời gian/Kết hợp Cảm biến, Gán nhãn/Học chủ động, Dữ liệu lớn/Quản lý đội xe và các Tác vụ Nhận thức (Phát hiện, Theo dõi, Phân loại, Phân đoạn)[13]. Các thành phần và hệ thống quan trọng trong nền tảng công nghệ của họ bao gồm:

- Nhận thức (Perception): Sử dụng một bộ cảm biến đa phương thức để hiểu rõ môi trường xung quanh.
- Dự đoán (Prediction): Dự đoán hành vi của những người tham gia giao thông khác bằng cách sử dụng trí tuệ nhân tạo và học máy.
- Lập kế hoạch (Planning): Xác định các quỹ đạo lái xe an toàn và hiệu quả dựa trên nhận thức và dự đoán[14].
- Điều khiển (Controls): Thực hiện các hành động đã lên kế hoạch thông qua các bộ phận chấp hành của xe (vô lăng, ga, phanh).
- Bản đồ (Mapping): Tạo và sử dụng bản đồ độ nét cao cho việc định vị và điều hướng[15].

- Cruise đã trải qua một quá trình phát triển công nghệ đáng kể, bắt đầu với các bộ công cụ trang bị thêm cho xe hiện có, sau đó tiến tới các phương tiện hoàn toàn tự lái như Cruise AV và Origin. Ban đầu, Cruise tập trung vào bộ công cụ RP-1, được thiết kế cho hệ thống lái tự động trên đường cao tốc. Sau đó, công ty đã chuyển hướng sang phát triển phần mềm cho xe tự lái hoàn toàn. Cruise AV, dựa trên nền tảng Chevrolet Bolt, đã trở thành nền tảng thử nghiệm và phát triển ban đầu cho dịch vụ taxi robot của họ. Tiếp theo là Cruise Origin, một phương tiện được thiết kế riêng cho mục đích tự lái, loại bỏ hoàn toàn các bộ phận điều khiển thủ công và tập trung vào sự thoải mái và hiệu quả cho hành khách. Tuy nhiên, gần đây, dự án phát triển Origin đã bị hủy bỏ, thay vào đó, Cruise sẽ tập trung vào việc điều chỉnh thế hệ xe Bolt EV tiếp theo cho công nghệ tự lái[16][17].

#### d) NVIDIA - Hỗ Trợ Phần Cứng:

- NVIDIA DRIVE là một nền tảng máy tính được phát triển bởi Nvidia, tập trung vào việc cung cấp các chức năng hỗ trợ lái xe và xe tự lái, được vận hành bởi công nghệ học sâu. Nền tảng này lần đầu tiên được giới thiệu tại Triển lãm Điện tử Tiêu dùng (CES) ở Las Vegas vào tháng 1 năm 2015. Mục tiêu cốt lõi của NVIDIA DRIVE là hiện thực hóa các cấp độ tự động hóa khác nhau, từ các hệ thống hỗ trợ lái xe tiên tiến (ADAS) cho đến những chiếc xe hoàn toàn không người lái. Tham vọng này được thể hiện rõ qua tầm nhìn của CEO NVIDIA về một tương lai với hàng tỷ dặm đường được di chuyển bởi các phương tiện tự hành mỗi năm, cho thấy quy mô to lớn mà họ hướng đến. Việc NVIDIA gia nhập thị trường xe tự lái từ khá sớm, vào năm 2015, cho thấy một cam kết chiến lược dài hạn đối với lĩnh vực này. Thay vì phản ứng với các xu hướng thị trường hiện tại, NVIDIA đã chủ động xây dựng nền tảng và tích lũy kinh nghiệm trong một thời gian dài, cho phép họ phát triển một danh mục sản phẩm toàn diện theo thời gian. Hơn nữa, việc tập trung vào học sâu ngay từ đầu đã làm nổi bật thế mạnh cốt lõi của NVIDIA trong lĩnh vực điện toán tăng tốc bằng GPU và niềm tin của họ vào trí tuệ nhân tạo như là yếu tố then chốt để hiện thực hóa khả năng tự lái. Việc tận dụng vị thế dẫn đầu của họ trong điện toán AI mang lại cho NVIDIA một lợi thế đáng kể so với các đối thủ có thể có nền tảng ô tô truyền thống vững chắc hơn nhưng lại thiếu chuyên môn về AI[18][19][20].

- Lịch sử phát triển của nền tảng NVIDIA DRIVE đã trải qua nhiều cột mốc quan trọng, bắt đầu với các nền tảng dựa trên kiến trúc Maxwell, bao gồm Drive CX, được thiết kế cho khoang lái kỹ thuật số, và Drive PX, một nền tảng phát triển ban đầu nhắm đến các loại xe bán tự lái và tự lái. Tiếp theo đó là Drive PX 2 dựa trên kiến trúc Pascal, với nhiều cấu hình khác nhau cho AutoCruise và AutoChauffeur, thậm chí đã được Tesla áp dụng. Sự ra đời của kiến trúc Volta và Turing đã dẫn đến dòng sản phẩm DRIVE AGX, bao gồm Xavier và Pegasus, với sức mạnh tính toán AI ngày càng tăng. Các hệ thống dựa trên kiến trúc Ampere và DRIVE Thor trong tương lai, dựa trên kiến trúc Blackwell, thể hiện sự phát triển không ngừng về khả năng phần cứng. Sự tiến bộ liên tục từ Maxwell đến Blackwell cho thấy khả năng cải tiến và lặp lại nhanh chóng của NVIDIA trong việc nâng cao sức mạnh xử lý và thiết kế kiến trúc, đặc biệt được điều chỉnh cho các khối lượng công việc liên quan đến xe tự lái. Trong một lĩnh vực đòi hỏi tài nguyên điện toán ngày càng tăng cho các mô hình AI phức tạp và xử lý dữ liệu cảm biến thời gian thực, sự phát triển phần cứng không ngừng này là vô cùng quan trọng. Việc Tesla áp dụng Drive PX 2 từ sớm đã nhấn mạnh thành công ban đầu và tầm ảnh hưởng của NVIDIA trong ngành, ngay cả với các công ty hiện đang theo đuổi các giải pháp nội bộ. Sự hợp tác ban đầu này đã cung cấp sự xác nhận có giá trị trong thế giới thực cho nền tảng của NVIDIA và giúp củng cố uy tín của họ trong lĩnh vực ô tô[21].

- Kiến trúc tổng thể của nền tảng NVIDIA DRIVE dựa trên chiến lược "ba máy tính": DGX để huấn luyện các mô hình AI, Omniverse để mô phỏng và tạo dữ liệu tổng hợp, và DRIVE AGX cho điện toán trên xe. Nền tảng này là một giải pháp toàn diện, bao gồm phần cứng, phần mềm (DriveOS, Drive AV, DriveWorks) và một hệ thống an toàn toàn diện (NVIDIA Halos). DRIVE Hyperion đóng vai trò là một nền tảng tham chiếu tích hợp cảm biến để tạo mẫu và xác thực. Chiến lược "ba máy tính" nhấn mạnh sự phức tạp của quá trình phát triển xe tự lái và cách tiếp cận toàn diện của NVIDIA, bao trùm toàn bộ vòng đời phát triển từ thu thập và huấn luyện dữ liệu đến triển khai và xác thực. Cách tiếp cận toàn diện này giúp NVIDIA khác biệt so với các công ty chỉ tập trung vào phần cứng trên xe hoặc phần mềm mô phỏng. Việc nhấn mạnh vào một nền tảng toàn diện và sự ra mắt của Halos cho thấy sự tập trung ngày càng tăng của NVIDIA vào các khía cạnh quan trọng về an toàn và độ tin cậy, vốn là những yếu tố cần thiết để xe tự lái được chấp nhận rộng rãi. Nhận thức được sự giám sát chặt chẽ của công chúng và các cơ quan quản lý đối với vấn đề an toàn của xe tự lái, NVIDIA đang chủ động xây dựng một khuôn khổ để giải quyết những lo ngại này[22][23].

## 6) CÁC THÀNH PHẦN CHÍNH CỦA XE TỰ LÁI

Xe tự lái gồm các thành phần chính sau: Nhận thức, Dự đoán, Lập kế hoạch đường đi và Điều khiển.

## 6.1 Cảm biến phần cứng: Camera vs. LiDAR[24]:

- Các phương pháp học sâu đặc biệt phù hợp để phát hiện và nhận dạng đối tượng trong hình ảnh 2D cũng như đám mây điểm 3D, được thu thập từ camera video và cảm biến LiDAR (Light Detection and Ranging).

- Trong lĩnh vực xe tự hành, cảm biến LiDAR đóng vai trò quan trọng trong việc cung cấp mô hình 3D của môi trường xung quanh dưới dạng đám mây điểm. Hiệu suất của LiDAR được đánh giá dựa trên các yếu tố như phạm vi quét, góc quan sát, độ phân giải và tốc độ khung hình. Chẳng hạn, các cảm biến như Velodyne có góc quét ngang 360°, trong khi các phương tiện di chuyển với tốc độ cao cần tầm quét tối thiểu 200m để đảm bảo phản ứng kịp thời với các tình huống giao thông. Độ chính xác của việc phát hiện vật thể 3D phụ thuộc vào độ phân giải cảm biến, với các loại LiDAR tiên tiến có thể đạt độ chính xác tới 3cm.

- Hiện nay, có nhiều tranh luận về việc sử dụng camera hay LiDAR làm công nghệ cảm biến chính cho xe tự hành. Hai công ty dẫn đầu trong lĩnh vực này - Tesla và Waymo - theo đuổi các chiến lược khác nhau. Waymo tập trung phát triển xe tự hành cấp độ 5, với hơn 10 triệu dặm đã vận hành hoàn toàn tự động. Trong khi đó, Tesla triển khai hệ thống AutoPilot như một phần của hệ thống hỗ trợ lái nâng cao (ADAS), cho phép người dùng tùy chỉnh bật hoặc tắt khi cần. Lợi thế của Tesla nằm ở lượng dữ liệu khổng lồ thu thập được từ các phương tiện do khách hàng sử dụng, với tổng số dặm lái lên đến hơn 1 tỷ.

- Về công nghệ cảm biến, Tesla chủ yếu dựa vào hệ thống camera, trong khi Waymo sử dụng LiDAR làm công nghệ cốt lõi. Mỗi phương pháp đều có điểm mạnh và hạn chế riêng. LiDAR có độ chính xác cao và hoạt động tốt ngay cả trong điều kiện thiếu sáng, nhưng chi phí cao và dễ bị ảnh hưởng bởi thời tiết xấu như mưa lớn. Ngược lại, camera có chi phí thấp hơn nhưng không thể đo độ sâu một cách trực tiếp và gặp khó khăn khi hoạt động trong bóng tối hoặc điều kiện thời tiết bất lợi.

- Một số nghiên cứu gần đây[25] đã tìm cách tái tạo dữ liệu giống LiDAR từ hình ảnh. Các nhà khoa học tại Đại học Cornell đã phát triển phương pháp sử dụng ước lượng độ sâu từ ảnh để tái tạo đám mây điểm 3D, gọi là pseudo-LiDAR. Dữ liệu này có thể được sử dụng trong các mô hình học sâu 3D như PointNet hoặc AVOD để phân tích. Nếu phương pháp này đạt hiệu quả cao, nó có thể mở ra cơ hội ứng dụng rộng rãi hơn cho xe tự hành, vì loại bỏ được sự phụ thuộc vào cảm biến LiDAR - một trong những thành phần phần cứng đắt đỏ nhất trên xe tự động.

- Bên cạnh camera và LiDAR, các cảm biến khác như radar và siêu âm cũng được tích hợp để nâng cao khả năng nhận diện môi trường. Ví dụ, xe Waymo sử dụng ba cảm biến LiDAR, năm radar và tám camera, trong khi Tesla trang bị tám camera, 12 cảm biến siêu âm và một radar trước.

Bảng 3: So sánh giữa LiDAR, Camera và AI kết hợp trong xe tự hành

Tiêu chí	LiDAR	Camera	AI kết hợp (Camera + AI)
<b>Nguyên lý hoạt động</b>	Dùng tia laser để đo khoảng cách và tạo bản đồ 3D	Dùng ánh sáng nhìn thấy để thu ảnh màu	Dùng AI để tái tạo bản đồ 3D từ hình ảnh
<b>Độ chính xác về chiều sâu</b>	Cao (sai số dưới 3cm)	Kém hơn, cần ước tính độ sâu	Cải thiện bằng học sâu và thị giác máy tính
<b>Hoạt động trong bóng tối</b>	Hoạt động tốt	Kém, cần đèn hồng ngoại hoặc hỗ trợ	Cải thiện bằng AI và cảm biến hồng ngoại
<b>Ảnh hưởng bởi thời tiết</b>	Bị ảnh hưởng bởi mưa, sương mù, bụi	Bị ảnh hưởng mạnh bởi sương mù, mưa lớn	AI có thể giúp cải thiện trong điều kiện xấu
<b>Trường nhìn (FoV)</b>	360° (với LiDAR quay)	Hạn chế theo góc của camera	Mở rộng bằng nhiều camera và AI
<b>Chi phí</b>	Cao	Thấp	Trung bình (camera rẻ nhưng cần xử lý AI)
<b>Tính ổn định và bảo trì</b>	Dễ hỏng do có bộ phận chuyển động	Bền hơn do không có bộ phận chuyển động	Phụ thuộc vào AI và phần cứng xử lý
<b>Ứng dụng thực tế</b>	Waymo sử dụng chính	Tesla chủ yếu dựa vào camera	Kết hợp cả hai để tận dụng ưu điểm của từng công nghệ

## 6.2 Nhận diện làn đường:

- Vì mỗi camera đều có đặc điểm riêng, nên việc hiệu chỉnh (calibration) là cần thiết để thu được hình ảnh không bị méo, sạch và sẵn sàng cho các bước xử lý tiếp theo. Do đó, 20 hình ảnh từ các góc nhìn khác nhau của một mẫu bàn cờ (chessboard) được chụp lại khi mẫu này được treo trên tường. Mẫu bàn cờ mang lại cấu trúc có độ tương phản cao giúp dễ dàng phát hiện các góc, từ đó tính toán hệ số biến dạng (distortion factor).

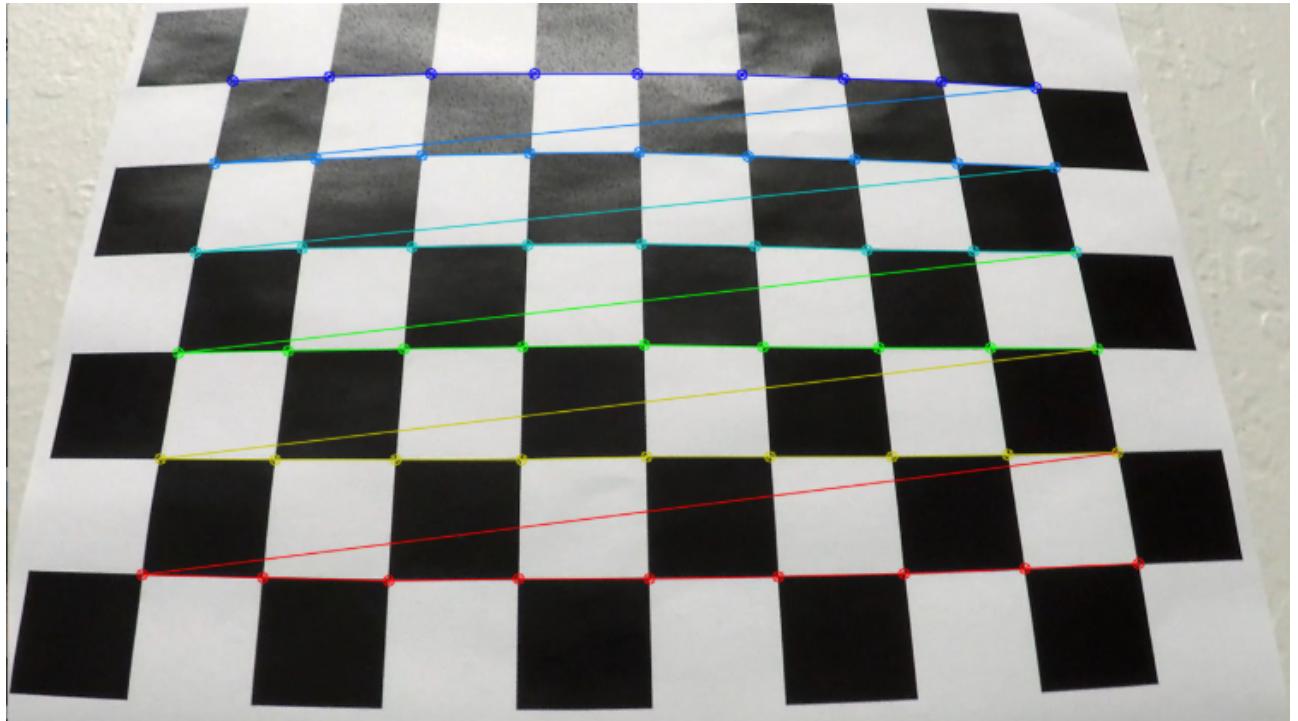
- Một vòng lặp được thực hiện trên tất cả các ảnh hiệu chỉnh. Trong mỗi lần lặp:

- Ảnh được chuyển sang ảnh xám (grayscale).

- Sau đó, lệnh `findChessboardCorners()` của OpenCV được gọi để phát hiện các góc trong ảnh.
- Nếu số lượng góc phát hiện được là chính xác – ví dụ đối với một bàn cờ có 10 hàng và 7 cột, số lượng góc cần tìm là:

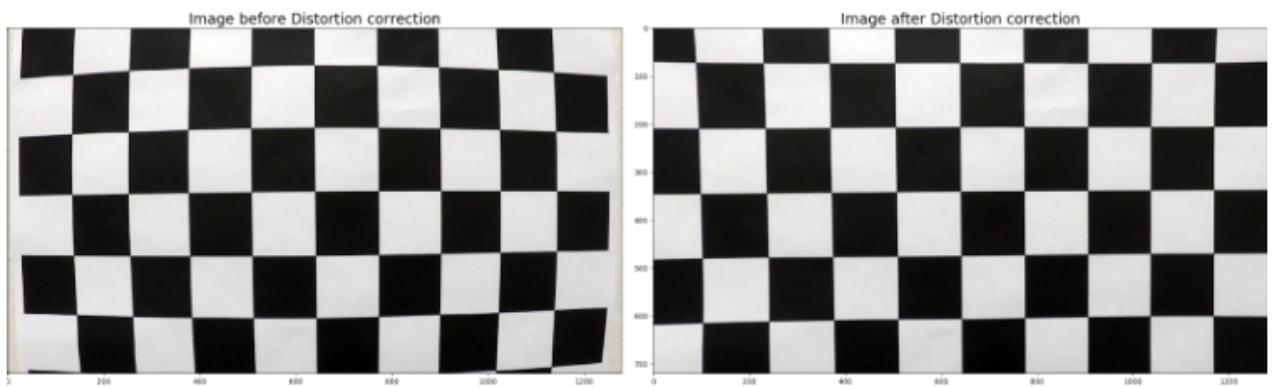
$$(10 - 1) \times (7 - 1) = 54(10 - 1) \times (7 - 1) = 54(10 - 1) \times (7 - 1) = 54\text{ góc}.$$

- Kết quả sẽ được lưu vào một vector điểm ảnh (image points). Đồng thời, một vector điểm vật thể (object points) cũng được tạo ra, chứa các tọa độ không bị biến dạng, với giá trị tọa độ  $z = 0$ , vì tất cả các điểm này nằm trên cùng một mặt phẳng.



- Sau khi tất cả các ảnh hiệu chỉnh đã thêm kết quả của chúng vào vector điểm ảnh (image points) và vector điểm vật thể (object points), hai vector này sẽ được sử dụng trong lệnh `calibrateCamera()` của OpenCV. Lệnh này sẽ trả về, ngoài các giá trị khác, ma trận hiệu chỉnh (calibration matrix) và các hệ số biến dạng (distortion coefficients).

- Kết quả thu được như hình sau:



- Tương tự đối với các ảnh từ nhiều góc của camera ta xác định làn đường theo các bước sau:

#### Bước 1: Loại bỏ méo ảnh đầu vào (Undistort input image)

- Trước tiên, ảnh đầu vào sẽ được hiệu chỉnh (undistort) dựa trên thông tin đã thu được từ bước tiền xử lý, cụ thể là ma trận hiệu chỉnh và các hệ số biến dạng.



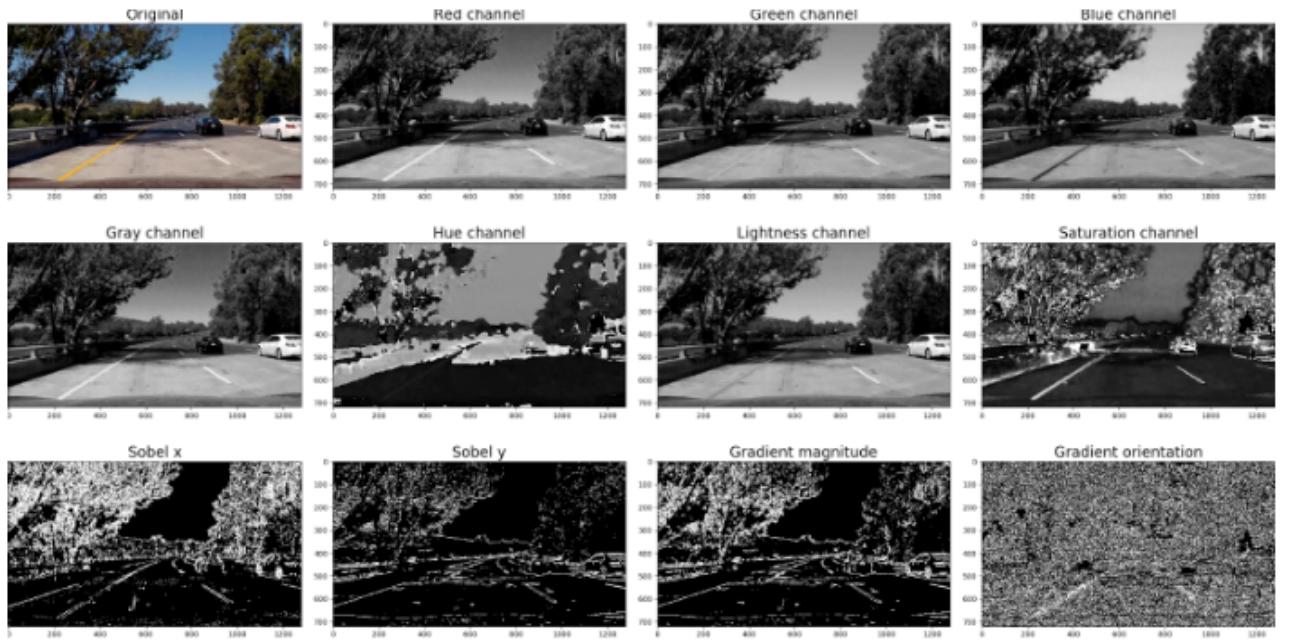
- Đặc biệt, nếu bạn nhìn vào các cạnh của cả hai hình ảnh, bạn có thể thấy rõ hiệu ứng của việc hiệu chỉnh biến dạng (distortion correction).

#### Bước 2: Nghiêng hóa ảnh đã hiệu chỉnh để thu được thông tin làn đường

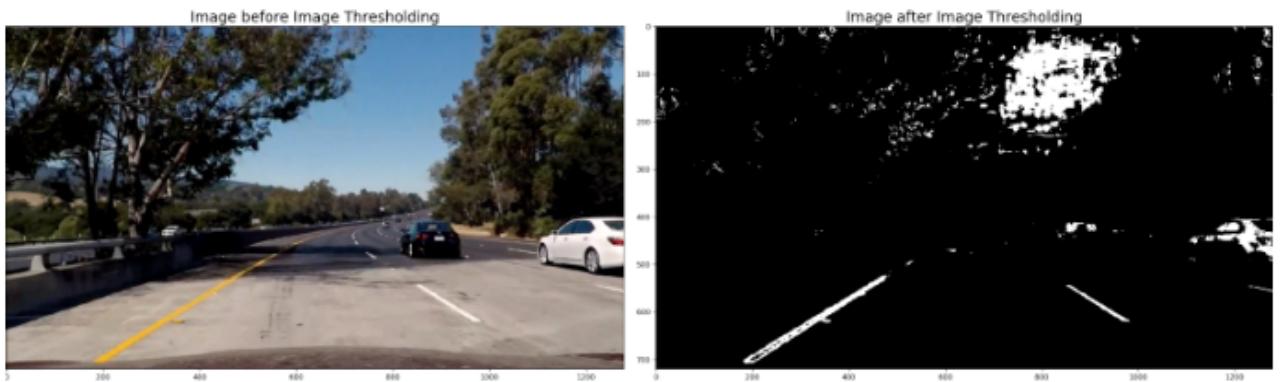
- Trong bước này, ảnh sau khi đã được hiệu chỉnh (undistorted) sẽ được phân tích để trích xuất thông tin về làn đường.

- Cụ thể:

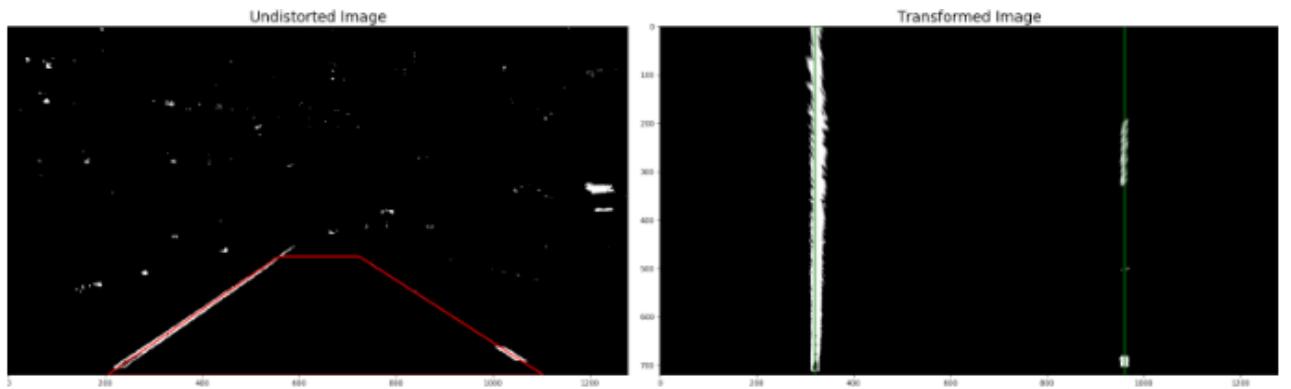
- Mỗi kênh màu trong các không gian màu RGB và HLS sẽ được hiển thị để kiểm tra.
- Ngoài ra, toán tử Sobel được áp dụng theo cả hướng x và y trên phiên bản ảnh xám (grayscale).
- Đồng thời, các đặc trưng như biên độ gradient (Gradient Magnitude) và hướng gradient (Gradient Orientation) cũng được tính toán.



- Một tổ hợp giữa kênh Đỏ (Red Channel) và kênh Độ bão hòa (Saturation Channel) được sử dụng. Cụ thể, mỗi kênh sẽ được lọc bằng một ngưỡng giá trị tối thiểu, và sau đó hai kết quả được kết hợp lại bằng toán tử "AND".

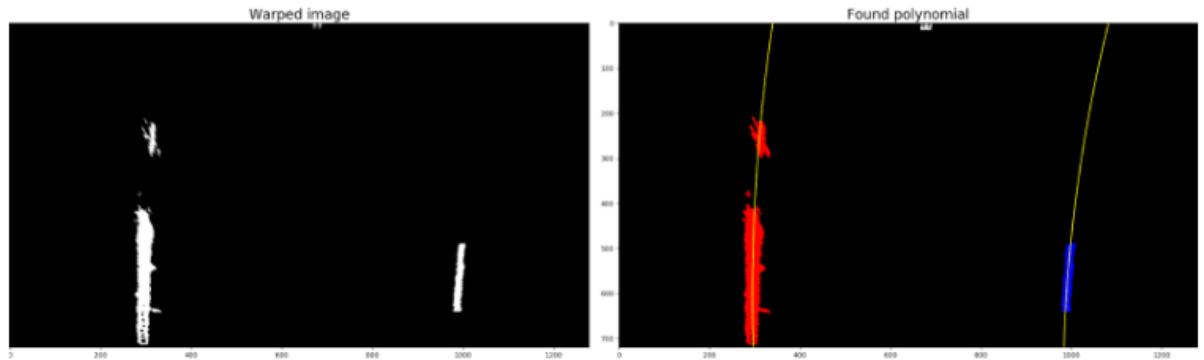


### Bước 3: Biến đổi phôi cảnh



#### Bước 4: So khớp Polynomial

- Tiếp theo, một đa thức bậc hai (polynomial bậc hai) được khớp (fit) thông qua các cửa sổ trượt (sliding windows) đã xác định trước đó.
- Kết quả của việc khớp đa thức có thể được thấy tại đây:



#### Bước 5: Tính toán độ cong và độ lệch trung tâm

- Độ cong của làn đường (curvature) và độ lệch làn đường (lane offset) có thể được xác định bằng cách sử dụng đa thức bậc hai đã khớp có dạng:

$$f(y) = Ay^2 + By + C$$

- Trong đó:

- y: là trục dọc (chiều sâu hoặc chiều dài xe đang đi),
- f(y): là vị trí theo trục ngang (trục x),
- A, B, C : là các hệ số của đường cong, được xác định từ việc fit đa thức vào các điểm làn đường đã phát hiện được trong ảnh.

- Điều quan trọng cần lưu ý là: việc chuyển đổi từ đơn vị pixel sang đơn vị mét trong thế giới thực đã được thực hiện để có kết quả chính xác hơn.

- Ví dụ: Giả sử các hệ số như sau: A = 0.0003, B = -0.2, C = 300

⇒ Tính độ cong (curvature) ?

- Công thức tính bán kính độ cong tại một điểm y:

$$R = \frac{(1 + (2Ay + B)^2)^{3/2}}{|2A|}$$

- Giả sử y = 720 (điểm gần đáy ảnh – thường là điểm gần xe nhất), thay vào ta được:

$$R = \frac{1 + (2 \times 0.0003 \times 720 - 0.2)^2)^{3/2}}{|2 \times 0.0003|}$$

$$R \approx \frac{1.081}{0.0006} \approx 1803.02 \text{ pixels}$$

⇒ Nếu cần tính theo đơn vị mét, cần thêm bước chuyển đổi pixel sang mét theo tỷ lệ của ảnh (thường khoảng 30m chiều dọc và 3.7m chiều ngang).

- Ví dụ: Giả sử ảnh có chiều rộng 1280 pixels, thì tâm ảnh là 640. Ta giả sử xe đang ở tâm ảnh.

⇒ Tính độ lệch làn đường (lane offset)/Vị trí làn đường trung tâm tại đáy ảnh ?

$$f(720) = A(720)^2 + B(720) + C$$

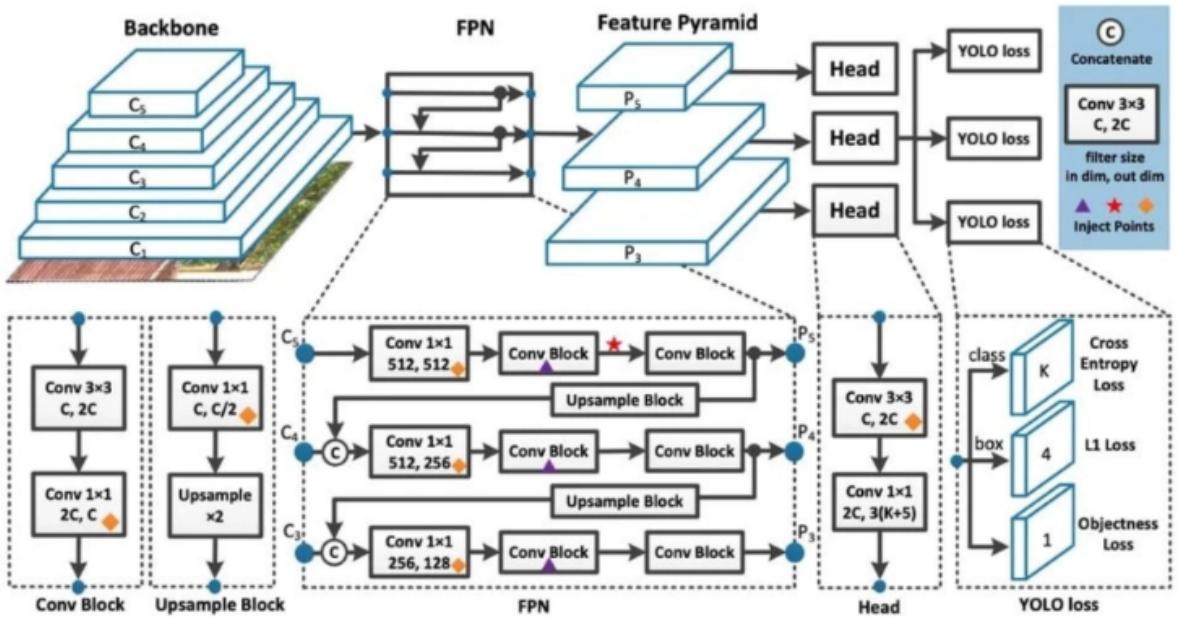
$$f(720) = 0.0003 \times 720^2 - 0.2 \times 720 + 300 = 311.52$$

$$\text{Offset} = 640 - 311.52 = 328.48 \text{ pixels}$$

- Giả sử 1 pixel = 3.7 / 700 m (tùy thuộc vào tỷ lệ ảnh), ta có thể chuyển sang mét nếu cần.

### 6.3 Nhận diện vật thể:

- Mặc dù độ chính xác của Lidar là khá cao nhưng nó đòi hỏi về tài nguyên các phần cứng và tốn khá nhiều chi phí. Do đó ở bài báo cáo này chúng tôi sẽ sử dụng theo phương pháp Kết hợp Camera và các mô hình học. Ở đây để thực hiện nhận dạng, phân loại và truy vết chúng tôi sử dụng mô hình YOLOv8.



Hình 2: Kiến trúc YOLOv8

#### - Input:

- Hình ảnh/Video: Các mô hình của YOLOv8 được thiết kế để nhận đầu vào là ảnh tĩnh hoặc các khung hình liên tục từ video.
- Dữ liệu tiền xử lý: Trước khi đưa vào mạng, dữ liệu thường được resize (điều chỉnh kích thước) về kích thước chuẩn mà mô hình yêu cầu (ví dụ:  $640 \times 640$  pixel), đồng thời thực hiện chuẩn hóa (normalization) giá trị pixel và có thể áp dụng thêm các kỹ thuật augmentation nhằm tăng cường tính đa dạng của dữ liệu huấn luyện.

#### - Output:

- Bounding Boxes (Hộp giới hạn): Mỗi đối tượng được xác định sẽ có một bounding box chứa tọa độ vị trí (x, y) điểm góc trên bên trái và kích thước chiều rộng, chiều cao.
- Nhãn (Labels): Mỗi bounding box đi kèm với nhãn xác định loại đối tượng, như “xe hơi”, “xe tải”, “xe máy”,... trong trường hợp nhận diện phương tiện.
- Điểm số độ tin cậy (Confidence Score): Mỗi dự đoán kèm theo một giá trị xác suất (từ 0 đến 1) thể hiện mức độ tin cậy của mô hình đối với phát hiện đó.
- Các thông số bổ sung cho tracking (nếu áp dụng): Trong ứng dụng truy vết, các thuật toán phụ trợ (ví dụ: Deep SORT, ByteTrack) thường sử dụng kết quả từ YOLOv8 (bounding boxes và nhãn) làm đầu vào để gán ID duy nhất cho từng đối tượng, từ đó liên kết đối tượng qua các khung hình.

- YOLOv8 là mô hình single-stage detector, nghĩa là tất cả các thao tác nhận dạng đối tượng được thực hiện trong một lần truyền xuôi (forward pass) qua mạng neural. Điều này giúp giảm thiểu độ trễ và tăng tốc độ xử lý, rất cần thiết trong các ứng dụng thời gian thực như giám sát giao thông.

- **Anchor-Free Detection:** Một trong những cải tiến quan trọng của YOLOv8 là chuyển sang kiến trúc anchor-free. Thay vì dựa vào các anchor boxes đã định trước (một tập hợp các hộp với tỷ lệ cố định), mô hình này trực tiếp dự đoán các điểm trung tâm của đối tượng cũng như kích thước hộp bao quanh. Điều này giúp đơn giản hóa việc xử lý, giảm thiểu việc điều chỉnh các hyperparameters và thường cho kết quả tốt hơn đối với các đối tượng có kích thước khác nhau.

- **Tích Hợp Cơ Chế Feature Pyramid:** Để phát hiện được cả đối tượng lớn và nhỏ, YOLOv8 áp dụng cơ chế Feature Pyramid Networks (FPN) hoặc các biến thể cải tiến như PAN (Path Aggregation Network) nhằm thu thập thông tin đa cấp độ từ các feature maps khác nhau. Điều này giúp nâng cao khả năng phát hiện, đặc biệt khi đối tượng có kích thước hoặc tỷ lệ khác nhau trong cùng một khung hình.

- **Kết Hợp với Thuật Toán Tracking:** Trong các ứng dụng truy vết, sau khi YOLOv8 nhận diện đối tượng từ từng khung hình, các thuật toán tracking sẽ được áp dụng để liên kết các đối tượng được phát hiện qua các khung hình liên tiếp. Một số thuật toán tracking thường dùng bao gồm:

- Deep SORT: sử dụng thông tin về vị trí (bounding boxes) kết hợp với embeddings học được từ hình ảnh của đối tượng để gán các ID duy nhất và theo dõi chúng qua thời gian.

- YOLOv8 có thể được chia thành ba phần chính:

### 1. Backbone:

- Mục đích: Trích xuất đặc trưng từ ảnh đầu vào. Các lớp convolution được thiết kế theo kiến trúc hiện đại, tích hợp các block như CSP (Cross-Stage Partial) và các layer Focus giúp giảm nhiễu loạn dữ liệu, tối ưu hóa tốc độ xử lý và tăng khả năng học hỏi các đặc trưng quan trọng.

- Đặc điểm: Backbone chịu trách nhiệm tạo ra một bộ các feature maps với nhiều cấp độ trừu tượng khác nhau nhằm phục vụ cho quá trình dự đoán ở các giai đoạn tiếp theo.

### 2. Neck:

- Mục đích: Kết hợp và tối ưu hóa các feature maps thu được từ Backbone để phù hợp với việc phát hiện đối tượng ở nhiều tỷ lệ khác nhau.

- Cơ chế:

- FPN/PAN: Các kiến trúc FPN hoặc PAN được sử dụng để tổng hợp đặc trưng từ các cấp độ khác nhau, giúp mô hình học được thông tin từ cả các đặc điểm tổng quát lẫn chi tiết nhỏ.
- Fusion Layers: Các lớp hợp nhất giúp cân bằng giữa thông tin chi tiết và thông tin tổng quát.

### 3. Head:

- Mục đích: Dự đoán cuối cùng cho từng vùng được xác định, bao gồm việc định vị (bounding box), phân loại (nhân đối tượng) và tính toán điểm số độ tin cậy.

- Cơ chế:

- Anchor-Free Prediction: Khác với các phiên bản trước, YOLOv8 thường sử dụng cách tiếp cận anchor-free để trực tiếp dự đoán các vị trí trung tâm và kích thước bounding box.
- Loss Functions: Các hàm mất mát (loss functions) được thiết kế để tối ưu hóa đồng thời quá trình dự đoán vị trí, phân loại và giảm thiểu lỗi. Các hàm này có thể bao gồm các thành phần như IoU loss (Intersection over Union), classification loss và confidence loss.

#### 6.4 Phân đoạn ngữ nghĩa:

- U-Net là một kiến trúc mạng nơ-ron được thiết kế đặc biệt cho bài toán segmentation – phân vùng từng pixel của ảnh vào một lớp cụ thể. Cái tên "U-Net" bắt nguồn từ hình dạng chữ U của kiến trúc mạng, gồm:

- Encoder (bên trái) – trích xuất đặc trưng (feature extraction).
- Decoder (bên phải) – phục hồi kích thước không gian và tạo ảnh đầu ra.
- Skip Connections (mũi tên xanh) – kết nối giữa encoder và decoder giúp giữ lại thông tin chi tiết không gian.

- Input: Ảnh RGB kích thước 960x640.

- Output: Mask nhị phân kích thước 960x640.

##### 1. Encoder (Downsampling path – đường xuống):

- Mỗi khối gồm:

- 2 tầng Conv2D (Conv + BatchNorm + ReLU): trích xuất đặc trưng.
- 1 tầng Max Pooling (mũi tên đỏ): giảm kích thước ảnh, tăng chiều sâu (số kênh feature maps).

- Mục tiêu: giữ lại thông tin ngữ nghĩa cao nhưng giảm độ phân giải dần.

- Ví dụ:

- Ảnh đầu vào: kích thước  $256 \times 256 \times 3$
- Sau mỗi khối: kích thước giảm còn  $128 \times 128 \times 128$ , rồi  $64 \times 64 \times 64$ , v.v. trong khi số kênh tăng:  $64 \rightarrow 128 \rightarrow 256 \rightarrow 512$

##### 2. Bottleneck (điểm đáy chữ U):

- Đây là lớp sâu nhất, mang thông tin trừu tượng nhất của ảnh.

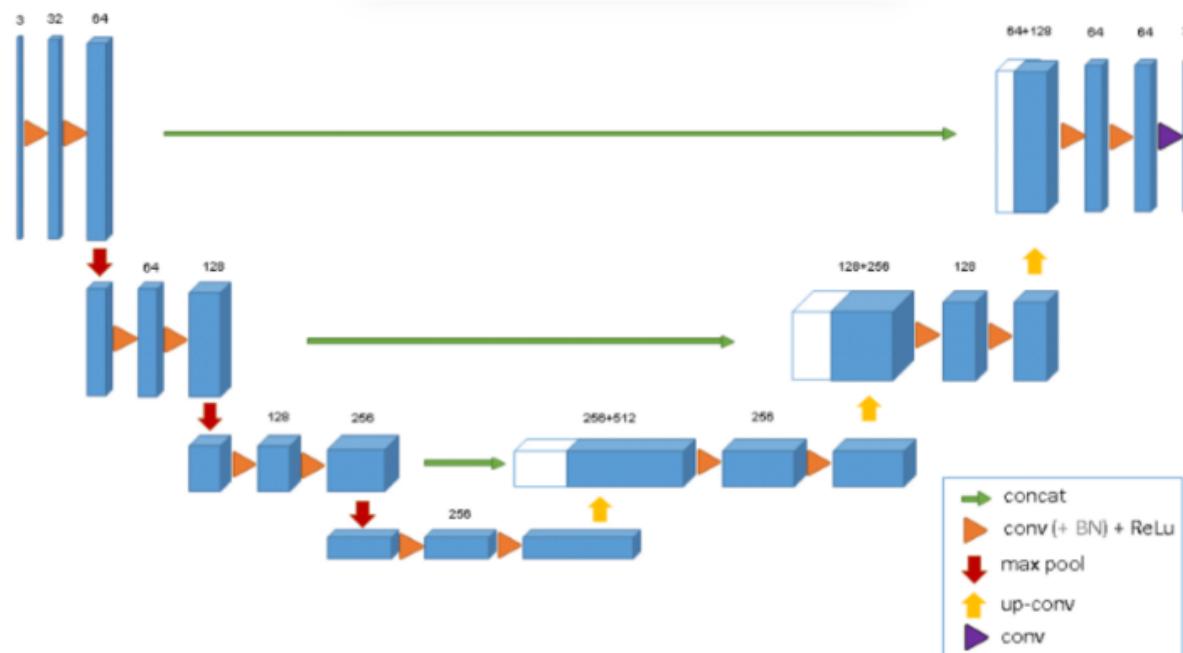
- Có thể coi là "nút cổ chai", giữ lại những đặc trưng quan trọng nhất trước khi phục hồi lại ảnh.

### 3. Decoder (Upsampling path – đường lên):

- Mỗi khôi gồm:

- Up-convolution (mũi tên vàng): dùng ConvTranspose2D để phóng to kích thước ảnh (giống như upsampling).
- Concatenation (mũi tên xanh lá): nối đặc trưng từ encoder tương ứng để khôi phục chi tiết không gian.
- 2 tầng Conv2D: để học lại đặc trưng từ ảnh có kích thước lớn hơn.

⇒ Skip connections giúp mô hình giữ được chi tiết (edges, biên...) đã bị mất trong quá trình pooling.



Hình 3: Kiến trúc Unet

- YOLOv8, mặc dù rất mạnh mẽ trong phát hiện đối tượng (object detection) và thậm chí đã bổ sung tính năng instance segmentation với YOLO-Seg, nhưng khi so với U-Net trong phân đoạn ngữ nghĩa, vẫn có một số lý do chính khiến YOLOv8 kém hơn:

#### (a) Mục đích và Kiến trúc của YOLO và U-Net:

- U-Net được thiết kế đặc biệt cho semantic segmentation. Nó sử dụng một mạng encoder-decoder với các skip connections giúp giữ lại thông tin chi tiết của ảnh gốc khi phân đoạn. Điều này rất quan trọng để phân loại chính xác từng pixel vào các lớp cụ thể (ví dụ: đường, cây, người, vách kẽ,...).

- YOLOv8 (và các phiên bản YOLO trước đó) được phát triển chủ yếu cho object detection và instance segmentation, với mục tiêu phát hiện đối tượng và xác định vị trí của chúng. Mặc dù YOLOv8 cải thiện khả năng phân đoạn với YOLO-Seg, nhưng nó chủ yếu tìm kiếm các vùng chứa đối tượng, chứ không phân loại từng pixel trong ảnh một cách chi tiết như U-Net.

(b) **Phân đoạn pixel-level:**

- U-Net có khả năng phân đoạn ở mức độ pixel-level, tức là phân loại từng pixel trong ảnh vào một lớp nhất định. Điều này rất quan trọng trong các ứng dụng như xe tự lái, nơi mà việc nhận diện chính xác từng phần của môi trường xung quanh là cần thiết (ví dụ: phân biệt rõ ràng các đối tượng như vỉa hè, đường, hoặc các vật thể động).
- YOLOv8 không tập trung vào phân loại từng pixel mà chủ yếu tìm kiếm các vùng chứa đối tượng. Mặc dù có sự hỗ trợ cho segmentation trong YOLO-Seg, nhưng nó vẫn hạn chế trong việc phân loại chi tiết các vùng không phải đối tượng, đặc biệt là trong các trường hợp môi trường phức tạp, chẳng hạn như đường phố với nhiều yếu tố cần phân đoạn.

(c) **Chất lượng phân đoạn trong ngữ nghĩa:**

- U-Net sử dụng skip connections để kết nối thông tin từ các tầng encoder xuống các tầng decoder, giúp cải thiện chất lượng phân đoạn chi tiết. Điều này giúp mô hình duy trì được thông tin về các chi tiết nhỏ, quan trọng trong phân đoạn ngữ nghĩa.
- YOLOv8 không tập trung vào duy trì thông tin chi tiết như U-Net, mà tập trung vào việc nhanh chóng phát hiện đối tượng và vẽ các bounding box. Mặc dù YOLOv8 có thể phân đoạn các đối tượng (instance segmentation), nhưng đối với các tác vụ phân đoạn ngữ nghĩa với các lớp không phải đối tượng, hiệu quả có thể không cao.

(d) **Độ chính xác trong môi trường xe tự lái:**

- Trong môi trường xe tự lái, phân đoạn chính xác các đối tượng và cảnh vật là rất quan trọng. U-Net và các biến thể của nó được tối ưu hóa cho các nhiệm vụ này, với độ chính xác cao trong việc phân đoạn các vùng như đường, cây, biển báo, người đi bộ,... mà không cần phải nhận diện từng đối tượng riêng biệt.
- YOLOv8 lại có xu hướng ưu tiên phát hiện đối tượng hơn là phân đoạn chi tiết các lớp ngữ nghĩa, điều này khiến mô hình không phải là sự lựa chọn lý tưởng khi cần phân đoạn chi tiết môi trường xung quanh.

(e) **Yêu cầu tài nguyên và hiệu suất:**

- Mặc dù YOLOv8 có thể xử lý nhanh hơn, nhờ vào việc tối ưu hóa cho việc phát hiện đối tượng và khả năng chạy trên phần cứng giới hạn, nhưng trong trường hợp phân đoạn ngữ nghĩa chi tiết, U-Net lại có thể cung cấp kết quả chính xác hơn về độ phân giải và chi tiết phân đoạn.

## 6.5 Lập kế hoạch đường đi cho xe tự hành:

- Lập kế hoạch đường đi (Path Planning) là một trong những nhiệm vụ cốt lõi của xe tự hành, giúp phương tiện xác định lộ trình tối ưu từ điểm xuất phát đến điểm đích. Để đảm bảo an toàn và hiệu suất hoạt động, hệ thống phải tính toán quỹ đạo di chuyển sao cho tránh được các chướng ngại vật, đồng thời thích ứng với điều kiện giao thông thay đổi liên tục.

- Theo nghiên cứu của S. Shalev-Shwartz [26] và đồng nghiệp trong báo cáo nghiên cứu "Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving" năm 2016, môi trường lái xe tự động là một hệ thống đa tác nhân, trong đó xe tự hành không chỉ di chuyển theo quỹ đạo cố định mà còn cần tương tác thông minh với các phương tiện khác, người đi bộ, và các yếu tố động khác trong giao thông. Điều này đòi hỏi hệ thống phải có khả năng xử lý các tình huống phức tạp như vượt xe, nhập làn, nhường đường, hay di chuyển trong các khu vực giao thông không có cấu trúc rõ ràng.

- Một cách tiếp cận phổ biến để lập kế hoạch đường đi là sử dụng hàm phần thưởng (reward function). Trong đó:

- Nếu xe gặp tai nạn, hệ thống nhận một phần thưởng âm lớn  $R(\bar{s}) = -r$ , nhằm tránh các tình huống nguy hiểm.
- Với các quỹ đạo khác, phần thưởng nằm trong khoảng  $R(\bar{s}) \in [-1, 1]$ , phản ánh chất lượng của hành động.

- Mục tiêu là tìm ra chính sách điều hướng tối ưu để thực hiện các thao tác lái xe phức tạp một cách mượt mà và an toàn.

- Để đạt được khả năng phản ứng nhanh và chính xác, thuật toán lập kế hoạch đường đi cần tốc độ xử lý cao, giúp xe có thể đưa ra quyết định trong thời gian thực. Khảo sát trong [27] chia quá trình lập kế hoạch đường đi thành ba cấp độ chính:

- **Lập kế hoạch nhiệm vụ (Mission Planning):** Xác định tuyến đường tổng thể dựa trên bản đồ và dữ liệu điều hướng.
- **Lập kế hoạch hành vi (Behavior Planning):** Dưa ra quyết định chiến lược như vượt xe, dừng lại hay thay đổi làn đường.
- **Lập kế hoạch chuyển động (Motion Planning):** Tính toán quỹ đạo chi tiết cho xe, đảm bảo di chuyển mượt mà và an toàn.

- Mặc dù nghiên cứu của S. D. Pendleton và đồng nghiệp trong bài nghiên cứu "Perception, Planning, Control, and Coordination for Autonomous Vehicles" năm 2017 đã đề cập đến các kỹ thuật truyền thống trong lập kế hoạch đường đi, nhưng chưa khai thác sâu về ứng dụng của Deep Learning trong lĩnh vực này. Ngày càng có nhiều nghiên cứu tập trung vào việc sử dụng mô hình học sâu để cải thiện hiệu suất và độ chính xác của hệ thống điều hướng. Hai phương pháp nổi bật trong hướng tiếp cận này là **Học Bắt Chước** (Imitation Learning - IL) và **Học Tăng Cường Sâu** (Deep Reinforcement Learning - DRL).

## 6.6 Học bắt chước và tăng cường học sâu:

- Học Bắt Chước (IL) là một phương pháp học máy trong đó hệ thống xe tự hành được huấn luyện để bắt chước hành vi lái xe của con người dựa trên dữ liệu thu thập từ các tài xế thực tế [28]. Mô hình sẽ quan sát các hành động mà con người thực hiện trong những tình huống khác nhau, sau đó học cách mô phỏng những hành động đó.

- Các nghiên cứu như NeuroTrajectory [29] sử dụng mạng nơ-ron tích chập (CNN) để phân tích dữ liệu lái xe, từ đó dự đoán quỹ đạo di chuyển tối ưu của xe trong một khoảng thời gian nhất định. Phương pháp này cho phép xe tự hành tái tạo hành vi của con người, giúp xe có phong cách lái xe tự nhiên hơn.

- Ngoài ra, Học Bắt Chước cũng có thể được tiếp cận dưới dạng Học Tăng Cường Ngược (Inverse Reinforcement Learning - IRL). Thay vì trực tiếp sao chép hành vi của con người, IRL cố gắng học ra hàm phần thưởng ẩn từ dữ liệu lái xe thực tế [30]. Bằng cách này, xe có thể hiểu được các nguyên tắc lái xe tối ưu thay vì chỉ sao chép hành động.

### - **Ưu điểm:**

- Có thể học từ dữ liệu thực tế, giúp xe mô phỏng cách lái xe tự nhiên của con người.
- Ít phụ thuộc vào mô hình mô phỏng, giảm nguy cơ sai lệch khi triển khai thực tế.

### - **Hạn chế:**

- Dữ liệu thực tế thường thiếu các tình huống hiếm gặp như xe chạy sai làn, tai nạn giao thông,... khiến mô hình khó phản ứng với những tình huống chưa từng gặp.
- Nếu dữ liệu huấn luyện có lỗi hoặc hành vi lái xe không tối ưu, hệ thống cũng có thể học theo những lỗi này.

- Khác với IL, phương pháp Học Tăng Cường Sâu (DRL) huấn luyện mô hình bằng cách thử nghiệm và điều chỉnh hành động trong môi trường mô phỏng. Hệ thống sẽ thực hiện nhiều hành động khác nhau, quan sát hậu quả và điều chỉnh chiến lược để tối ưu hóa phần thưởng.

- Trong DRL, môi trường thực tế được mô hình hóa thành một môi trường mô phỏng ảo [31]. Tuy nhiên, một thách thức lớn của phương pháp này là hàm mục tiêu có thể gây ra sai lệch trong mô hình, dẫn đến các quyết định không an toàn.

- Một giải pháp hiệu quả được đề xuất trong [32] là sử dụng chính sách kết hợp (hybrid policy) gồm hai phần:

- **Chính sách học được (learnable policy):** Dùng deep learning để học cách tối ưu hóa hàm phần thưởng, bao gồm các yếu tố như an toàn, thoái mái, cơ hội vượt xe, v.v.
- **Chính sách không học được (non-learnable policy):** Áp đặt các ràng buộc cứng về an toàn, đảm bảo xe luôn di chuyển trong phạm vi an toàn cho phép.

### - **Ưu điểm:**

- Có thể khám phá nhiều tình huống lái xe khác nhau trong môi trường mô phỏng, bao gồm cả các trường hợp nguy hiểm mà dữ liệu thực tế không có.
- Không phụ thuộc vào dữ liệu lái xe thực tế, giúp mô hình có khả năng học hỏi liên tục.

**- Hạn chế:**

- Khi triển khai trong môi trường thực tế, mô hình DRL có thể có độ chệch cao, dẫn đến quyết định không phù hợp.
- Quá trình huấn luyện thường tốn nhiều thời gian và tài nguyên tính toán.

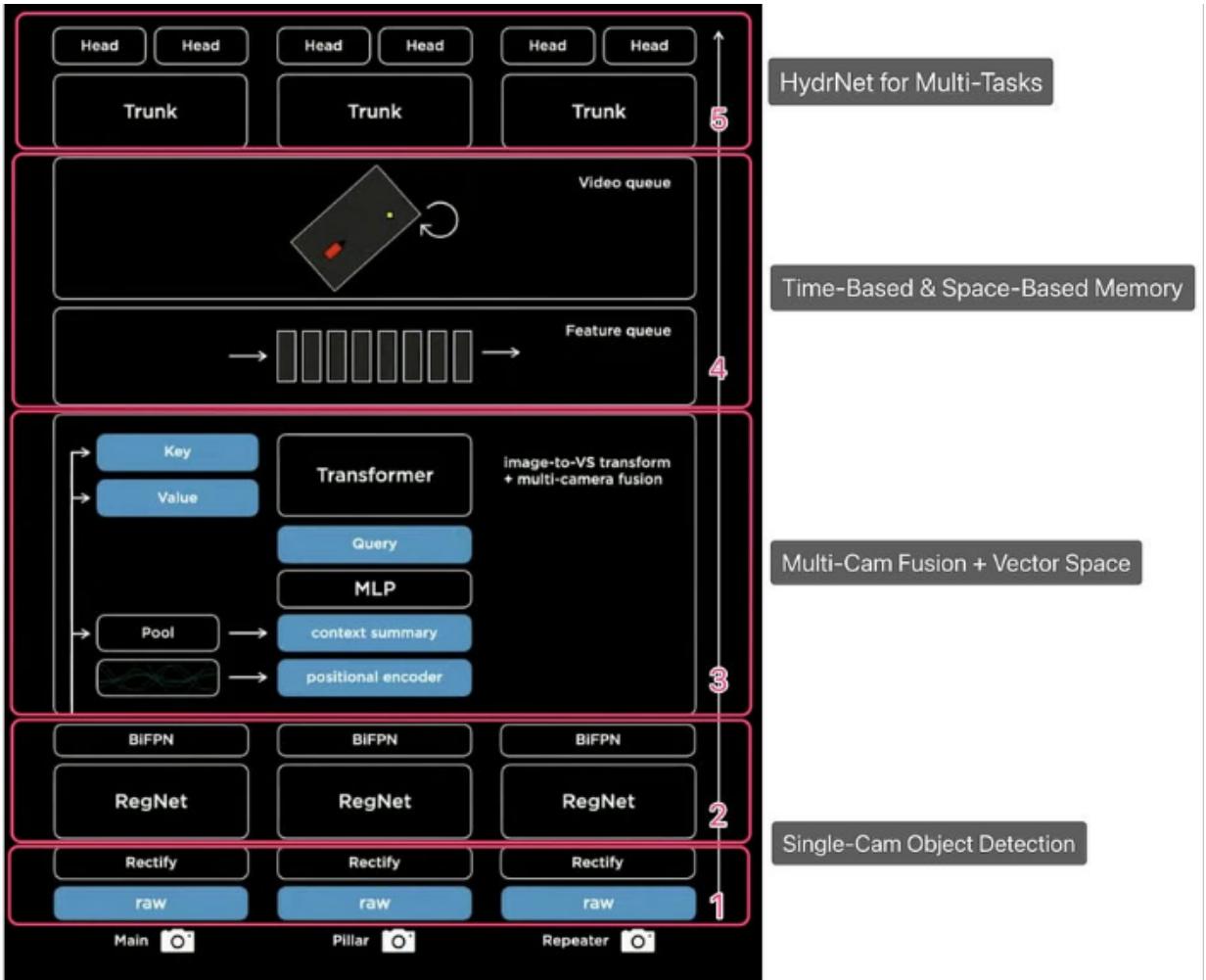
## 6.7 Ví dụ chi tiết trong TESLA:

- Lý do vì sao nhóm tập trung vào Tesla đó là do Tesla sử dụng Camera chứ không phải LiDAR hay Radar. Có những lý do sau đây:

- Cung cấp thông tin trực quan giống mắt người, đủ để nhận diện vật thể, đọc biển báo, và hiểu ngữ cảnh, dù khó hơn trong việc đo độ sâu trực tiếp.
- Triết lý của Elon Musk: Musk lập luận rằng con người lái xe chỉ dựa vào mắt (tương đương camera) và não (tương đương mạng nơ-ron), nên xe tự lái cũng có thể làm được mà không cần cảm biến bổ sung. Camera cho phép hệ thống học cách “nhìn” và hiểu môi trường giống con người, như nhận diện biển báo “Đừng lại” hoặc dự đoán ý định của người đi bộ.
- Tiến bộ trong AI bù đắp hạn chế.
- Camera rẻ hơn nhiều so với LiDAR và radar, đơn giản hóa sản xuất.

### 6.7.1 Thu thập và gán nhãn dữ liệu:

- Nguồn dữ liệu: Tesla thu thập dữ liệu từ hàng ngàn xe Tesla đang lưu thông trên đường hàng ngày.
- Khó khăn: Để huấn luyện một mạng nơ-ron dự đoán độ sâu, cần một bộ dữ liệu khổng lồ gồm hàng triệu video đã được gán nhãn.
  - Giải pháp: Tesla sử dụng phương pháp gán nhãn "bán giám sát" (semi-supervised) bằng cách sử dụng mạng nơ-ron khác để tự động gán nhãn cho dữ liệu (Tesla không công khai - có thể là Transformer).
  - Gán nhãn offline: Quá trình gán nhãn được thực hiện offline, nghĩa là không trực tiếp trên xe. Điều này cho phép sử dụng các mô hình sử dụng mô hình phức tạp hơn, không ảnh hưởng đến hiệu suất lái xe và kết hợp nhiều nguồn dữ liệu. Mục tiêu là tạo ra "ground truth label" (nhãn chính xác cao) cho dữ liệu.



Hình 4: Kiến trúc tổng quát các tác vụ trong xe tự lái của Tesla

#### 6.7.2 Quy trình trong mạng học sâu của Tesla - Full Self-Driving (FSD)[33][34][35]:

##### Tầng 1: Camera Input và Rectification (Camera xử lý thô):

- Như đã nói ở trên, Tesla sử dụng tám camera để cung cấp tầm nhìn 360 độ với phạm vi lên đến 250 mét. Các camera được đặt ở phía trước (camera góc hẹp, camera chính, camera góc rộng), hai bên hông xe giữa và các phía sau (trực tiếp phía sau, và phía sau hông).
- Input: raw (ảnh thô) → Méo hình (do ống kính góc rộng – fisheye), Khác biệt về màu sắc, độ sáng giữa các camera, Lỗi đồng bộ (một số ảnh chụp sớm/trễ hơn).
- Để giải quyết vấn đề này, Tesla tạo ra một "camera ảo" (virtual camera) - ám chỉ việc chuẩn hóa dữ liệu từ nhiều nguồn thành một định dạng thống nhất. Mục tiêu là chuyển đổi dữ liệu hình ảnh từ tất cả 8 camera thành một định dạng thống nhất, giống như được chụp bởi một camera duy nhất, camera ảo này.
- Rectify (chuẩn hóa ảnh):

- Dùng thông số hiệu chỉnh từ quá trình camera calibration (ma trận nội suy + thông số biến dạng) để khử méo hình.
- Mỗi camera có hệ tọa độ riêng, được chuyển về hệ tọa độ xe (ego-centric frame).
- Resize về kích thước chuẩn đầu vào của mạng RegNet (thường là 224x224 hoặc 640x640 tùy biến thể).
- Normalization: đưa pixel về khoảng [0,1] hoặc [-1,1].

⇒ Đảm bảo các ảnh từ 8 camera chụp tại cùng một thời điểm hoặc được cǎn chỉnh chính xác dựa trên timestamp.

- Mục tiêu:

- Khử méo (Do camera góc rộng), đồng bộ, và đảm bảo rằng ảnh từ các camera có thể được xử lý nhất quán trong hệ thống học sâu.
- Chuyển đổi ảnh từ dạng góc rộng bị biến dạng sang hình ảnh đã hiệu chỉnh để phù hợp với các mạng học sâu.

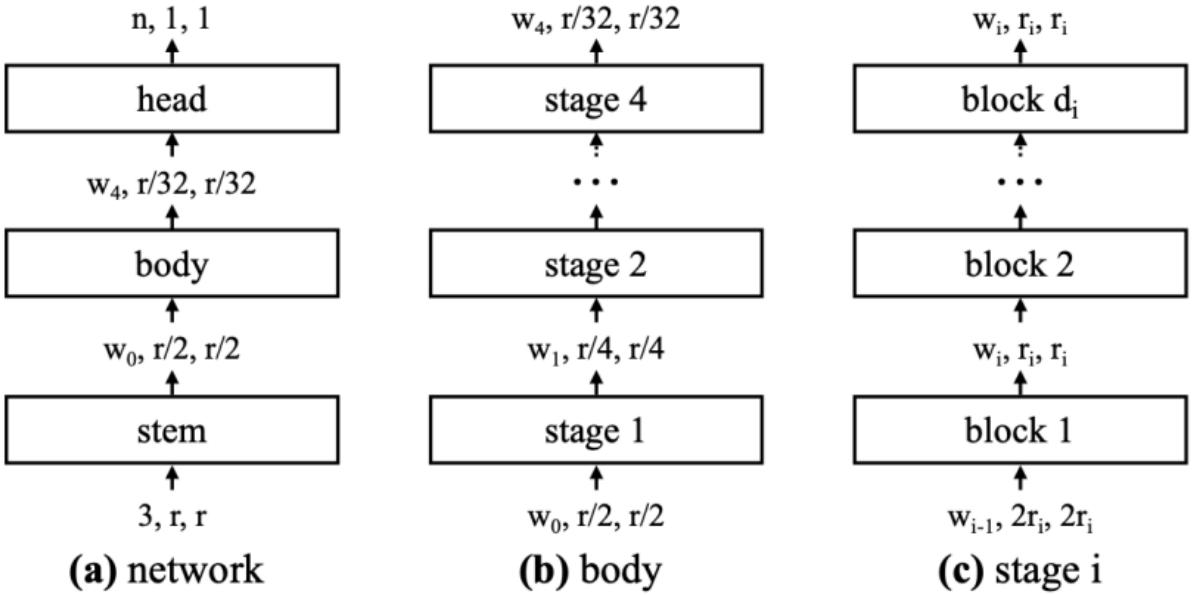
## Tầng 2: Single-Camera Object Detection (Xử lý từng camera riêng lẻ):

### (a) RegNet:

- RegNet không phải là một mô hình cụ thể mà là một không gian thiết kế (design space) chứa nhiều mô hình con với các đặc điểm chung, được gọi là RegNetX và RegNetY (hai biến thể chính). Các mô hình này được tạo ra bằng cách giới hạn không gian thiết kế để đảm bảo tính hiệu quả và khả năng mở rộng.
- Mục tiêu: Tìm ra các kiến trúc mạng có hiệu suất tốt (độ chính xác cao) và hiệu quả tính toán (FLOPs thấp, tốc độ nhanh) trên nhiều quy mô khác nhau (từ nhỏ đến lớn).
- Cách tiếp cận: Thay vì thiết kế thủ công một mạng cụ thể, nhóm nghiên cứu sử dụng phương pháp tìm kiếm không gian thiết kế (design space search), tập trung vào các đặc điểm chung của các mạng tốt để tạo ra các mô hình tổng quát hóa tốt.

- Đặc điểm nổi bật:

- Khả năng mở rộng: Các mô hình RegNet có thể được điều chỉnh để phù hợp với các ràng buộc tài nguyên khác nhau (ví dụ: số FLOPs, độ trễ).
- Tính đơn giản: Kiến trúc được thiết kế để dễ triển khai và tối ưu hóa trên nhiều phần cứng.
- Tính tổng quát: RegNet hoạt động tốt trên nhiều tác vụ thị giác máy tính, đặc biệt là phân loại ảnh.



Hình 5: Kiến trúc cơ bản của RegNet

- RegNet tuân theo cấu trúc CNN hiện đại, bao gồm các giai đoạn (stages) xử lý tuần tự:

i. **Phần (a) - Network (Mạng tổng thể) - Stem:**

- Đầu vào: Dữ liệu đầu vào có kích thước  $3, r, r$ , nghĩa là một ảnh RGB (3 kênh) với độ phân giải không gian  $r \times r$ .
- Stem: Lớp đầu tiên (stem) nhận đầu vào và giảm độ phân giải xuống  $r/2 \times r/2$ , đồng thời tăng số kênh lên  $w_0$ . Đây là một lớp tích chập đơn giản (thường là conv 3x3 với stride=2).
- Body: Phần thân của mạng, chứa các stage (giai đoạn), được mô tả chi tiết hơn ở phần (b).
- Head: Lớp cuối cùng, nhận đầu ra từ body với kích thước  $w_4, r/32, r/32$  (độ phân giải rất thấp do đã qua nhiều stage giảm kích thước) và tạo ra đầu ra cuối cùng với kích thước  $n, 1, 1$ , thường là một lớp global average pooling theo sau bởi một lớp fully connected để phân loại thành  $n$  lớp.

ii. **Phần (b) - Body (Thân mạng):**

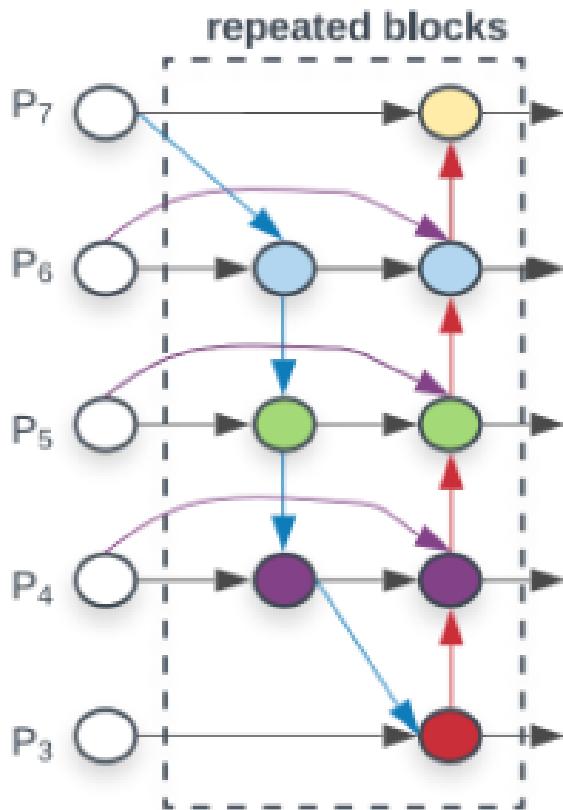
- Body được chia thành 4 stage (giai đoạn), từ stage 1 đến stage 4. Mỗi stage có các đặc điểm sau:
  - Stage 1: Nhận đầu ra từ stem ( $w_0, r/2, r/2$ ) và tạo ra đầu ra với số kênh  $w_1$  và độ phân giải  $r/4 \times r/4$ .
  - Stage 2: Tiếp tục giảm độ phân giải xuống  $r/8 \times r/8$ , số kênh tăng lên  $w_2$ .
  - Stage 3: Độ phân giải tiếp tục giảm.
  - Stage 4: Đầu ra cuối cùng của body là  $w_4, r/32, r/32$ , cho thấy độ phân giải giảm dần qua các stage (mỗi stage giảm độ phân giải xuống một nửa, từ  $r/2$  đến  $r/32$ , tức là qua 4

lần giảm).

### iii. Phản (c) - Stage i (Giai đoạn i) - Head:

- Mỗi stage bao gồm một số block (khối), từ block 1 đến block  $d_i$ , trong đó  $d_i$  là số block trong stage thứ i.
- Đầu vào của stage i có kích thước  $w_{i-1}, 2r_i, 2r_i$  và đầu ra có kích thước  $w_i, r_i, r_i$  cho thấy độ phân giải giảm một nửa (do stride=2 ở block đầu tiên của stage).
- Số kênh thay đổi từ  $w_{i-1}$  (đầu vào) sang  $w_i$  (đầu ra), phù hợp với việc tăng số kênh qua các stage.
- Các block trong stage có cùng cấu trúc, thường là residual block (cho RegNetX) hoặc bottleneck block với Squeeze-and-Excitation (cho RegNetY), như đã mô tả trong phần kiến trúc RegNet trước đó.

### (b) BiFPN:



Hình 6: Kiến trúc cơ bản của BiFPN

- BiFPN nhận đầu vào là một tập hợp các đặc trưng đa tỷ lệ từ backbone (thường là EfficientNet), được ký hiệu là  $(P_3, P_4, P_5, P_6, P_7)$ , trong đó:

- $P_3$  là đặc trưng ở độ phân giải cao nhất (tầng thấp nhất).
- $P_7$  là đặc trưng ở độ phân giải thấp nhất (tầng cao nhất).

- Quá trình hợp nhất đặc trưng trong BiFPN diễn ra như sau:

i. **Đường từ trên xuống (Top-Down Pathway):**

- Bắt đầu từ tầng cao nhất ( $P_3$ ) và truyền thông tin xuống các tầng thấp hơn ( $P_6, P_5, \dots, P_3$ ).
- Tại mỗi mức, đặc trưng từ tầng cao hơn được lấy mẫu lên (upsampling) và hợp nhất với đặc trưng ở tầng hiện tại.

ii. **Đường từ dưới lên (Bottom-Up Pathway):**

- Sau khi hoàn thành đường từ trên xuống, BiFPN thực hiện một đường từ dưới lên, bắt đầu từ tầng thấp nhất ( $P_3$ ) và truyền thông tin lên các tầng cao hơn ( $P_3, \dots, P_7$ ).
- Tại mỗi mức, đặc trưng từ tầng thấp hơn được lấy mẫu xuống (downsampling) và hợp nhất với đặc trưng ở tầng hiện tại.

iii. **Hợp nhất đặc trưng có trọng số (Weighted Feature Fusion):**

- Khi hợp nhất đặc trưng từ nhiều nguồn (ví dụ: đặc trưng từ tầng hiện tại, tầng trên, và tầng dưới), BiFPN sử dụng cơ chế trọng số để điều chỉnh mức độ đóng góp của từng đặc trưng.
- Trọng số được học trong quá trình huấn luyện và được chuẩn hóa để đảm bảo tổng trọng số bằng 1. Công thức hợp nhất tại một nút có thể được biểu diễn như sau:

$$O = \sum_i \frac{w_i}{\sum_j w_j + \epsilon} \cdot I_i$$

- trong đó:

- $O$ : Đầu ra sau khi hợp nhất.
- $I_i$ : Đặc trưng đầu vào từ nguồn thứ i.
- $w_i$ : Trọng số học được cho nguồn thứ i.
- $\epsilon$ : Một hằng số nhỏ (thường là  $10^{-4}$ ) để tránh chia cho 0.

- Để giảm chi phí tính toán, BiFPN sử dụng một biến thể nhanh của cơ chế trọng số, gọi là Fast Normalized Fusion, trong đó các trọng số được giới hạn trong khoảng [0,1] và được chuẩn hóa bằng cách sử dụng ReLU để đảm bảo không âm.

iv. **Lắp lại BiFPN:**

- Một tầng BiFPN (gồm một đường từ trên xuống và một đường từ dưới lên) được lắp lại nhiều lần để tăng cường khả năng hợp nhất đặc trưng.
- Số lần lắp (số tầng BiFPN) được xác định bởi phương pháp compound scaling, ví dụ: EfficientDet-D0 sử dụng 3 tầng BiFPN, trong khi EfficientDet-D7 sử dụng 8 tầng.
- Sau khi ảnh camera di qua Rectify  $\Rightarrow$  RegNet  $\Rightarrow$  BiFPN, Ta có một tập đặc trưng không gian (spatial features) giàu thông tin từ từng camera riêng lẻ. Đây là dữ liệu đã được chuẩn hóa và khuếch đại thông tin cần thiết cho tầng tiếp theo: Transformer fusion đa camera (tầng 3).

### Tầng 3: Multi-Cam Fusion và Vector Space Transformation (Hợp nhất đa camera và biến đổi không gian vector):

#### (a) Positional Embedding (Nhúng vị trí cho từng pixel):

- Positional Embedding (nhúng vị trí) là một kỹ thuật trong học sâu, thường được sử dụng trong các mô hình như Transformer, để mã hóa thông tin về vị trí hoặc thứ tự của các phần tử trong dữ liệu. Trong ngữ cảnh của Tesla FSD, Positional Embedding được áp dụng để gắn thông tin vị trí không gian thực tế (3D) và thời gian (temporal) cho từng pixel hoặc đặc trưng (feature) từ các camera, giúp hệ thống hiểu được chúng nằm ở đâu trong thế giới thực tế.

- Trong Tầng 3, mục tiêu chính là hợp nhất thông tin từ 8 camera để tạo ra một không gian 3D thống nhất (vector space) đại diện cho môi trường xung quanh xe. Tuy nhiên, mỗi camera cung cấp một góc nhìn 2D riêng biệt, với các đặc điểm sau:

- Khác biệt về góc nhìn: Một vật thể (ví dụ: một chiếc xe khác) có thể xuất hiện ở các vị trí khác nhau trên ảnh của camera trước, camera hông, hoặc camera sau.
- Méo hình: Camera góc rộng (fisheye) gây biến dạng, làm lệch vị trí thực tế của các vật thể.
- Không gian thực tế phức tạp: Để xe tự lái hiểu được một vật thể ở đâu trong không gian 3D (cách xe bao xa, ở hướng nào), hệ thống cần ánh xạ các pixel 2D từ camera sang tọa độ 3D thực tế.

- Positional Embedding giải quyết vấn đề này bằng cách:

- Ánh xạ không gian: Gắn cho mỗi pixel hoặc đặc trưng một "nhãn" định vị, cho biết nó tương ứng với điểm nào trong không gian 3D thực tế.
- Kết nối đa camera: Giúp mô hình nhận ra rằng một pixel ở camera A và một pixel ở camera B có thể đại diện cho cùng một vật thể (như cùng một chiếc xe), dựa trên vị trí 3D chung của chúng.
- Hỗ trợ Transformer: Transformer sử dụng cơ chế attention để so sánh và kết hợp đặc trưng từ các camera. Positional Embedding cung cấp thông tin vị trí để attention biết cách "nhìn" đúng vào các đặc trưng liên quan.

- Ví dụ: Nếu camera trước thấy một chiếc xe ở phía bên trái ảnh, và camera hông trái cũng thấy chiếc xe đó ở phía bên phải ảnh, Positional Embedding giúp hệ thống hiểu rằng cả hai đang nhìn cùng một vật thể, dựa trên tọa độ 3D thực tế của nó.

- Cách thực hiện Positional Embedding:

- Tính tọa độ 3D: Sử dụng thông số camera (intrinsics: tiêu cự, biến dạng; extrinsics: vị trí, hướng) để ánh xạ mỗi pixel từ ảnh 2D sang tọa độ 3D trong hệ tọa độ xe (ego-centric).

- Gán vector định vị: Mỗi đặc trưng trong feature map (từ Tầng 2) được gắn một vector 3D (x, y, z), đánh dấu vị trí thực tế của nó, ví dụ: một vật thể ở (5m, -2m, 0m).
  - Thêm temporal embedding: Gắn thông tin thời gian (như timestamp hoặc thứ tự frame) để ghi nhớ chuyển động của vật thể giữa các frame.
  - Kết hợp: Vector không gian và thời gian được thêm vào đặc trưng gốc, tạo đầu vào cho Transformer.
- Kết quả: Chuyển dữ liệu từ ảnh 2D riêng lẻ của 8 camera thành biểu diễn 3D trừu tượng (x, y, z), giúp Transformer hợp nhất đặc trưng từ các camera, nhận diện cùng một vật thể qua các góc nhìn.

(b) **Transformer / Cross-Camera Attention Block:**

- Mục đích: Hợp nhất đặc trưng từ 8 camera bằng cách so sánh và kết hợp thông tin, tạo ra biểu diễn không gian 3D thống nhất.
- Cách thực hiện:
  - Sử dụng cơ chế self-attention và cross-view attention trong Transformer.
  - So sánh đặc trưng: Xác định mối liên hệ giữa các pixel/dặc trưng từ các camera khác nhau (ví dụ: pixel ở camera A và B cùng chỉ một vật thể).
  - Kết hợp thông tin: Gộp dữ liệu từ các góc nhìn, giải quyết vấn đề như che khuất (vật thể chỉ rõ ở camera B, không thấy ở camera A) hoặc khác biệt góc nhìn (Cùng một xe nhìn khác nhau ở camera trước và sau, ta phải hiểu đó là cùng 1 xe).
  - Dựa trên Positional Embedding: Sử dụng tọa độ 3D từ bước trước để biết đặc trưng nào tương ứng với cùng vị trí thực tế.
- Kết quả: Tạo ra vectorized 3D space feature, một tập hợp đặc trưng hợp nhất, mô tả toàn cảnh môi trường xung quanh xe, sẵn sàng cho các bước tiếp theo.

(c) **Vector Space Transformation (Biểu diễn không gian vector):**

- Vector Space Transformation là bước chuyển đổi các đặc trưng (features) từ nhiều nguồn (trong trường hợp này là đặc trưng từ 8 camera sau khi qua Transformer/Cross-Camera Attention) thành một biểu diễn không gian vector 3D thống nhất. Biểu diễn này được gọi là một "bản đồ thế giới 3D trừu tượng", trong đó mỗi điểm trong không gian được mô tả bằng một vector đặc trưng đa chiều (x, y, z). Vector này chứa thông tin về:

- Vị trí của vật thể trong không gian thực tế (theo tọa độ xe).
- Thuộc tính của vật thể (như loại vật thể: xe, người đi bộ, biển báo; màu sắc; kích thước).
- Ngữ cảnh môi trường (như khoảng cách, hướng, chuyển động).

- Tính chất:

- Mọi điểm trong bản đồ là một vector đặc trưng đa chiều, mang thông tin về vật thể (class), vị trí, chuyển động, v.v.

- Không phụ thuộc vào góc nhìn camera, nghĩa là biểu diễn này là egocentric (dựa trên hệ tọa độ của xe, không dựa vào góc nhìn cụ thể của từng camera).
- Phương pháp: Có thể dùng MLP hoặc phép biến đổi tuyến tính để chuẩn hóa vector, tổ chức vào không gian 3D (voxel grid hoặc liên tục).
- Điểm khác nhau giữa bước **b)** và **c)** **đó là:**

Bảng 4: Bảng so sánh giữa Vector Space Transformation và Transformer/Cross Camera Attention Block

Thách thức	Mô tả chi tiết	Ví dụ cụ thể
Mục đích	Tập trung vào việc kết nối đặc trưng từ các camera	Tập trung vào việc tổ chức các đặc trưng đã hợp nhất thành một không gian 3D có cấu trúc
Quá trình xử lý	Sử dụng Transformer để so sánh và gộp đặc trưng dựa trên mối quan hệ không gian và ngữ cảnh	Tổ chức các đặc trưng thành một không gian vector, có thể dùng MLP hoặc phép biến đổi tuyến tính để chuẩn hóa.
Phạm vi tác động	Xử lý mối quan hệ giữa các camera, ví dụ: liên kết pixel từ camera A với pixel từ camera B	Xử lý toàn bộ không gian xung quanh xe, tạo biểu diễn tổng quát cho mọi vật thể trong môi trường

- Giả sử có một chiếc xe màu đỏ ở vị trí (5, -2, 0):

- Trong Cross-Camera Attention:
  - Camera trước cung cấp đặc trưng [v1] (xe đỏ) tại (5, -2, 0).
  - Camera hông trái cung cấp đặc trưng [v2] (xe đỏ) tại (5, -2, 0).
  - Transformer dùng attention để nhận ra [v1] và [v2] cùng chỉ một xe, tạo ra đặc trưng hợp nhất [v\_fused] tại (5, -2, 0), kết hợp thông tin từ cả hai camera (màu đỏ, hình xe, v.v.).
- Trong Vector Space Transformation:
  - Nhận [v\_fused] từ bước trên, hệ thống đặt nó vào một không gian 3D tại (5, -2, 0) với vector [0.9, 0.1, 5.0, -2.0, 0.0, 2.0, ...].
  - Vector này không chỉ chứa thông tin xe đỏ mà còn chuẩn hóa để biểu thị vị trí, tốc độ, hướng, và các thuộc tính khác trong một không gian tổng quát, không phụ thuộc vào camera nào đã thấy xe.

#### (d) Chuyển thành Bird's-Eye View (BEV):

- Bird's-Eye View (BEV) là một biểu diễn không gian 2D, nhìn từ trên cao xuống (như góc nhìn của một con chim), mô tả môi trường xung quanh xe trong hệ tọa độ của xe (ego-centric).

Trong BEV, mỗi điểm trên một lưới (grid) đại diện cho một khu vực trong không gian thực tế, chứa thông tin về:

- Vật thể: Xe, người đi bộ, biển báo, v.v.
- Làn đường: Vị trí và loại làn (liên, nét đứt).
- Khu vực trống: Không gian có thể di chuyển được.
- Các thuộc tính khác: Tốc độ, hướng, khoảng cách.

- Cách thực hiện:

- Input: Một không gian vector 3D từ bước Vector Space Transformation.
- Từ mỗi điểm ảnh (pixel) trong feature map của các camera (hoặc từ các đặc trưng trong không gian vector 3D), hệ thống dựng một chùm tia (ray) trong không gian thực tế. Chùm tia bắt đầu từ tâm camera và đi qua điểm ảnh trên cảm biến, được tính toán dựa trên **thông số nội tại - intrinsics** (Tiêu cự, tâm ảnh, ma trận biến dạng) và **thông số ngoại tại - extrinsics** (Vị trí và hướng của camera trên xe). Hệ thống xác định nơi chùm tia giao với mặt đường (ground plane, thường giả định  $z=0$ ), tạo ra một điểm  $(x, y)$  trong không gian thực tế.
- Gán vector đặc trưng vào lưới BEV (BEV grid - là một ma trận 2D với các ô (cells) đại diện cho các khu vực trong không gian thực tế). Nếu nhiều đặc trưng rơi vào cùng một ô, chúng có thể được kết hợp (ví dụ: lấy trung bình hoặc chọn đặc trưng mạnh nhất). Tesla có thể sử dụng một lớp học sâu (như ConvNet hoặc MLP) để chuẩn hóa các vector đặc trưng khi gán vào lưới, đảm bảo tính nhất quán. Các phương pháp như pillar-based (giống PointPillars) hoặc voxel-to-BEV projection (giống Lift-Splat-Shoot) có thể được áp dụng để ánh xạ đặc trưng.
- Từ đó hình thành một BEV feature map, nơi mỗi cell chứa một vector đặc trưng đa chiều mang thông tin về khu vực cụ thể trong không gian thật. Hay nói cách khác là đầu ra là BEV Grid – bản đồ từ trên cao chứa thông tin về vật thể, làn đường, khu vực trống, v.v.

#### Tầng 4: Time-Based và Space-Based Memory (Bộ nhớ không gian và thời gian):

- Time-Based và Space-Based Memory là một bước trong pipeline FSD của Tesla, tập trung vào việc lưu trữ và tích hợp thông tin từ các khung hình (frames) trước đó để hiểu môi trường xung quanh xe theo cả khía cạnh thời gian (temporal) và không gian (spatial). Tầng này hoạt động như một bộ nhớ ngắn hạn, giúp xe không chỉ phản ứng với dữ liệu hiện tại mà còn dự đoán xu hướng tương lai dựa trên lịch sử.

- Vấn đề trong thế giới thực:

- Xe di chuyển liên tục, camera cũng chuyển động nên ảnh thay đổi từng giây.
- Có những vật thể chỉ xuất hiện thoáng qua, nếu chỉ nhìn frame hiện tại thì dễ bỏ sót.

- Để theo dõi hành vi (trajectory) của vật thể, ta cần dữ liệu lịch sử.
- Trường hợp môi trường có thể bị che khuất tạm thời (occlusion), ta cần “trí nhớ” để suy luận.

- Cấu trúc tầng 4:

- Video Queue: Lưu lại các khung hình trong quá khứ.
- Feature Queue: Lưu lại các đặc trưng đã được trích xuất từ Transformer trong quá khứ.
- Frame hiện tại: Dữ liệu ngay bây giờ, gồm video từ 8 camera và bản đồ BEV hiện tại.

- Mục tiêu:

- Mục tiêu: Tích lũy thông tin từ nhiều khung hình (frames) và kết hợp không gian (BEV) để hiểu toàn cảnh, bền vững theo thời gian.
- Giúp xe ghi nhớ lịch sử chuyển động của các vật thể, bối cảnh xung quanh.
- Tạo điều kiện cho việc dự đoán hành vi (prediction).

⇒ Giống như bộ nhớ ngắn hạn – hỗ trợ xe không chỉ phản ứng tức thì mà còn có thể dự đoán xu hướng tương lai (xe khác sẽ rẽ ? dừng ? tiếp tục ?).

#### (a) **Video Queue (Temporal Memory - Bộ nhớ thời gian):**

- Video Queue lưu trữ một chuỗi các khung hình video (frames) từ quá khứ gần, thường là vài giây gần nhất (ví dụ: 2-5 giây, tương ứng với 60-150 frames ở 30fps). Các khung hình này đến từ tất cả 8 camera, cung cấp góc nhìn 360 độ của môi trường tại các thời điểm trước đó.
  - Như một short-term memory cho phép hệ thống “xem lại” các khung hình trước để suy luận hành vi hoặc phát hiện vật thể đã xuất hiện nhưng không còn trong frame hiện tại (ví dụ: đoán xe sắp rẽ, người sắp băng đường...).
- ⇒ Dùng cho các task như: motion prediction, intent prediction, trajectory planning.

#### (b) **Feature Queue (Spatial Memory - Bộ nhớ không gian):**

- Lưu trữ các đặc trưng không gian (BEV features) từ những frame trước đó.
  - Mỗi frame trước tạo ra một BEV feature map (lưới 2D từ trên cao), và các map này được lưu vào hàng đợi theo thứ tự thời gian.
  - Cơ chế này giống như trí nhớ không gian liên tục trong một vùng — ví dụ: Đoạn đường phía sau xe nhưng camera không nhìn thấy → vẫn còn nhớ được từ frame trước.
- ⇒ Dùng để hỗ trợ các task cần hiểu toàn cảnh môi trường và liên tục như: Occupancy (chiếm dụng không gian), Free space prediction (Dự đoán không gian trống) và Planning & navigation (Lên kế hoạch đường đi).

#### (c) **Cách hoạt động:**

- Mỗi frame hiện tại sẽ được kết hợp với thông tin từ cả Video Queue và Feature Queue.

- Việc kết hợp có thể thông qua Spatial RNN.
- Output cuối cùng là BEV feature map có tích hợp thông tin thời gian và không gian → truyền lên HydraNet để thực hiện nhiều nhiệm vụ.

(d) **Kiến trúc SPATIAL RNN:**

- Spatial RNN thường được triển khai dưới dạng ConvLSTM, là các mô hình kết hợp tích chập (convolution) và cơ chế tuần tự (recurrence).
- Input: Một chuỗi các tensor 2D (như BEV feature maps) với kích thước (T, C, H, W), trong đó:
  - T: Số frame (ví dụ: 60 frame từ Feature Queue + frame hiện tại).
  - C: Số kênh đặc trưng (số chiều của vector đặc trưng trong mỗi ô).
  - H, W: Chiều cao và chiều rộng của lưới BEV (ví dụ: 100x100).
- Hidden State: Duy trì một trạng thái ẩn (hidden state) và trạng thái ô (cell state), cả hai đều là tensor 2D, để lưu trữ thông tin không gian và thời gian.
- Convolution: Thay vì fully connected layers như RNN truyền thống, ConvLSTM dùng tích chập để xử lý cục bộ các ô lân cận, giữ cấu trúc không gian.
- Các bước thực hiện:

i. **Bước 1: Nhìn lại các bản đồ BEV cũ (Feature Queue)**

- Spatial RNN bắt đầu bằng cách xem xét các bản đồ BEV trong Feature Queue, giống như lật lại một cuốn album ảnh giao thông:
  - Nó kiểm tra từng bản đồ BEV, từ bản cũ nhất (cách đây 2 giây) đến gần nhất (cách đây 1/30 giây).
  - Với mỗi bản đồ, Spatial RNN “nhớ” những gì quan trọng, bỏ qua thông tin cũ không còn liên quan và kết nối các ô gần nhau.

ii. **Bước 2: Thêm chi tiết từ video cũ (Video Queue)**

- Tiếp theo, Spatial RNN nhìn vào Video Queue để lấy thông tin chi tiết hơn, giống như xem lại đoạn phim quay chậm:
  - Các khung hình video chứa dữ liệu thô, như màu sắc, hình dạng, hoặc tín hiệu của vật thể (ví dụ: xe bật đèn xi-nhan).
  - Vì video có độ phân giải cao và đến từ 8 camera, Tesla có thể đã nén chúng thành các đặc trưng đơn giản hơn (như qua một mạng nơ-ron khác) trước khi đưa vào Spatial RNN.
  - Spatial RNN “so sánh” các đặc trưng video với bản đồ BEV:
    - Ví dụ: Video cách đây 0.5 giây cho thấy xe đỏ bật đèn xi-nhan trái. Spatial RNN thêm thông tin này vào ô (10, 4) trên bản đồ BEV, gợi ý xe có thể sắp rẽ.

- Hoặc video cách đây 1 giây cho thấy người đi bộ nhìn sang trái, ám chỉ họ có thể băng đường. Spatial RNN ghi nhớ điều này cho ô (6, 8).
- Spatial RNN làm việc này bằng cách:
- Chọn các chi tiết quan trọng từ video, như chuyển động bất ngờ hoặc tín hiệu hành vi.
  - Bỏ qua những thứ không cần thiết, như bóng cây lay động không ảnh hưởng đến lái xe.
  - Căn chỉnh video với bản đồ BEV, đảm bảo thông tin từ camera (như người đi bộ ở góc phải ảnh) được đặt đúng vào ô BEV tương ứng (như ô (6, 8)).
  - Kết quả là “ký ức” của Spatial RNN được làm giàu với các chi tiết hành vi từ video, như xe đó có ý định rẽ hoặc người đi bộ sắp di chuyển.

### iii. **Bước 3: Kết hợp với frame hiện tại**

- Bây giờ, Spatial RNN lấy dữ liệu từ frame hiện tại (video và bản đồ BEV ngay bây giờ) và gộp nó với “ký ức” từ các bước trước, giống như thêm một trang mới vào cuốn album:
  - Từ bản đồ BEV hiện tại: Spatial RNN thấy xe đó ở ô (10, 4), nhưng không thấy người đi bộ (có thể bị xe tải che khuất).
  - Từ video hiện tại: Camera cho thấy xe đó vẫn di chuyển thẳng, nhưng không có dấu hiệu người đi bộ.
- Spatial RNN so sánh dữ liệu hiện tại với ký ức của nó:
  - Về xe đó: Ký ức từ Feature Queue cho biết xe đã di chuyển từ ô (8, 4) đến (10, 4), và Video Queue cho thấy nó bật xi-nhan trái. Kết hợp với frame hiện tại, Spatial RNN xác nhận xe đang đi thẳng nhưng có thể rẽ trái sắp tới.
  - Về người đi bộ: Ký ức từ Feature Queue cho thấy người đi bộ ở ô (6, 8) cách đây 0.5 giây, và Video Queue cho thấy họ đang tiến gần đường. Dù frame hiện tại không thấy người đi bộ, Spatial RNN quyết định giữ thông tin này, vì họ có thể vẫn ở gần.
- Spatial RNN “vẽ lại” bản đồ BEV, cập nhật:
  - Ô (10, 4) có xe đó, với thông tin bổ sung rằng nó có thể rẽ trái.
  - Ô (6, 8) có người đi bộ, dù không thấy trong frame hiện tại, vì lịch sử cho thấy họ ở đó.

### iv. **Bước 4: Tạo bản đồ BEV mới**

- Cuối cùng, Spatial RNN tạo ra một bản đồ BEV mới, giống như một bức tranh tổng hợp. Bản đồ này chứa mọi thứ xe cần biết ngay bây giờ:
  - Vị trí hiện tại của xe đó, người đi bộ, làn đường, khu vực trống.
  - Lịch sử chuyển động, như xe đó đã di chuyển 2 mét trong 1 giây.
  - Chi tiết hành vi, như xe đó có thể rẽ trái hoặc người đi bộ có nguy cơ băng đường.

- Mỗi ô trên bản đồ là một mảnh thông tin, được làm giàu từ cả frame hiện tại và ký ức quá khứ.
- Bản đồ BEV này được gửi sang Tầng 5, nơi hệ thống dùng nó để dự đoán (như người đi bộ sẽ đi đâu) và lập kế hoạch (như giảm tốc độ để an toàn).

#### Tầng 5: HydraNet và Occupancy Network:

- Tầng 5 là giai đoạn cuối trong pipeline xử lý dữ liệu của Tesla FSD, nơi các thông tin từ các tầng trước (đặc biệt là BEV feature map tích hợp thời gian và không gian từ Tầng 4) được sử dụng để đưa ra các dự đoán và quyết định cụ thể. Tầng này bao gồm hai thành phần chính:

- HydraNet:
  - Một mạng nơ-ron multi-task (đa nhiệm), được thiết kế để thực hiện nhiều tác vụ cùng lúc, như phát hiện vật thể, dự đoán chuyển động, phân tích làn đường, và lập kế hoạch đường đi.
  - Được gọi là “HydraNet” vì nó có cấu trúc giống như con rắn Hydra trong thần thoại, với một trunk (thân chính) chia sẻ và nhiều heads (đầu) chuyên biệt cho từng tác vụ.
- Occupancy Network:
  - Một mạng nơ-ron chuyên biệt để dự đoán vùng trống (free space) và vùng chiếm dụng (occupied space) trong môi trường xung quanh xe, kèm theo độ chắc chắn (confidence) cho mỗi dự đoán.
  - Giúp xe hiểu nơi nào có thể di chuyển an toàn và nơi nào có vật thể hoặc chướng ngại vật.

##### (a) HydraNet:

- HydraNet là một mạng nơ-ron đa nhiệm, được thiết kế để thực hiện nhiều tác vụ cùng lúc từ một đầu vào chung. Cách hoạt động của nó có thể được mô tả như sau:

- Đầu vào:
  - Nhận BEV feature map từ Tầng 4, là một lưới 2D (ví dụ: 100x100 ô, mỗi ô 0.5m) chứa các vector đặc trưng tích hợp thời gian và không gian.
  - Mỗi ô trên lưới chứa thông tin về: Vật thể (xe, người đi bộ, biển báo), thuộc tính (tốc độ, hướng, màu sắc), lịch sử chuyển động (quỹ đạo, hành vi) và bối cảnh không gian (làn đường, khu vực trống).
- Kiến trúc: Kiến trúc mạng chia sẻ chung phần “trunk” (bộ trích xuất đặc trưng) và có nhiều “head” (phục vụ các nhiệm vụ khác nhau) được chia ra từ trunk:
  - Trunk (Thân chính): Một mạng nơ-ron sâu (thường là Convolutional Neural Network - CNN hoặc Transformer) xử lý BEV feature map để trích xuất các đặc trưng cấp cao hơn. Trunk này giống như một “bộ não chung”, phân tích toàn bộ bản đồ BEV để hiểu môi trường một cách tổng quát.

- Heads: gồm Object Detection Head, Motion Prediction Head, Lane Analysis Head và Path Planning Head. Các heads hoạt động đồng thời, mỗi head là một mạng nơ-ron nhỏ hơn, được huấn luyện để chuyên biệt hóa cho tác vụ của nó, nhưng tận dụng đặc trưng chung từ trunk để tiết kiệm tính toán.
- Các nhiệm vụ (multi-heads):
  - Dự đoán vị trí, tốc độ, hướng di chuyển của các đối tượng (prediction).
  - Phân tích làn đường, khoảng cách tới vật thể, đèn giao thông...
  - Lập kế hoạch đường đi (planning) phù hợp với quy định giao thông, an toàn, và mục tiêu đã định.
- Đầu ra: Một tập hợp các dự đoán cụ thể:
  - Vật thể: Danh sách xe, người đi bộ, biển báo với vị trí, tốc độ, hướng.
  - Làn đường: Bản đồ các làn, ngã tư, vạch kẻ đường.
  - Quỹ đạo: Dự đoán chuyển động của vật thể (như xe đó sẽ rẽ trái trong 2 giây).
  - Kế hoạch: Lộ trình đề xuất (như giảm tốc, giữ làn, rẽ phải).

⇒ Các dự đoán này được truyền đến hệ thống điều khiển để thực thi (như nhấn phanh, đánh lái).

(b) **Occupancy Network:**

- Occupancy Network là mạng nơ-ron chuyên biệt để dự đoán vùng trống (free space) và vùng chiếm dụng (occupied space) trong môi trường xung quanh xe, kèm theo độ chắc chắn (confidence) cho mỗi dự đoán.

Bảng 5: Bảng mô tả chức năng của Occupancy Network

Chức năng	Mô tả
Phân loại không gian chiếm dụng	Xác định vùng nào là vật thể, vùng nào là trống
Dự đoán xác suất occupancy	Mỗi điểm trong BEV được gán xác suất có vật thể
Tái tạo hình dạng vật thể 3D Làm nền tảng cho detection và path planning	Đặc biệt cho các vật thể mờ, bị che khuất Đưa ra input cực kỳ quan trọng cho các bước sau

- Giả sử camera trước bị lóa nắng, không rõ xe phía trước có một chiếc container không. Occupancy Network sẽ học cách suy luận từ bối cảnh, từ những camera bên hông
- ⇒ Vẫn có thể xác định rằng khu vực đó đang bị chiếm dụng, mặc dù không rõ hình dạng cụ thể.

- Điều này cực kỳ quan trọng khi đưa ra những quyết định sau: Có nên dừng lại không ? Có thể chuyển làn không ? Có xe phía sau góc khuất không ?
- Đầu vào: Cũng nhận BEV feature map từ Tầng 4, tương tự HydraNet. Map này chứa thông tin tích hợp thời gian và không gian về vật thể, làn đường, và bối cảnh.
- Quá trình xử lý:

i. **Phân tích BEV feature map:**

- Occupancy Network nhìn vào từng ô trên lưới BEV, kiểm tra vector đặc trưng để xác định trạng thái:
  - Vùng chiếm dụng: Ô có vật thể (xe, người, chướng ngại vật).
  - Vùng trống: Ô không có vật thể, xe có thể di chuyển.
  - Không chắc chắn: Ô không rõ trạng thái (do che khuất, nhiễu, hoặc thiếu dữ liệu).

ii. **Dự đoán kèm độ chắc chắn:**

- Mỗi ô được gán một xác suất  $\Rightarrow$  Độ chắc chắn giúp hệ thống đánh giá mức độ tin cậy của dự đoán, đặc biệt trong điều kiện khó.

iii. **Xử lý trường hợp phức tạp:**

- Che khuất: Nếu ô X không thấy người đi bộ do xe tải che, Occupancy Network dựa vào lịch sử từ Tầng 4 (Feature Queue) để dự đoán người đi bộ vẫn ở đó.
- Nhiễu: Trong điều kiện lóa nắng hoặc sương mù, mạng dùng độ chắc chắn thấp để báo hiệu cần thận trọng.
- Vật thể nhỏ: Như ỏ gà hoặc mảnh vỡ trên đường, được phát hiện nhờ đặc trưng chi tiết từ BEV map.
- Đầu ra: Một Occupancy Grid (lưới chiếm dụng), là một bản đồ BEV nơi mỗi ô được gán nhãn:
  - Trống (free): Xe có thể di chuyển.
  - Chiếm dụng (occupied): Có vật thể hoặc chướng ngại.
  - Không chắc chắn (unknown): Cần kiểm tra thêm hoặc tránh. $\Rightarrow$  Mỗi nhãn đi kèm xác suất (confidence score).

(c) **Phối hợp giữa HydraNet và Occupancy Network:**

- HydraNet tập trung vào chi tiết vật thể và kế hoạch:
  - Xác định xe, người, làn đường, và dự đoán hành vi cụ thể.
  - Dưa ra lộ trình di chuyển (như rẽ trái, dừng).
- Occupancy Network tập trung vào không gian tổng quát:
  - Xác định vùng an toàn để di chuyển, đảm bảo xe tránh chướng ngại vật.

- Hỗ trợ HydraNet bằng cách cung cấp bản đồ vùng trống/chiếm dụng, giúp Path Planning Head chọn lộ trình khả thi.

- Tích hợp:

- Occupancy Grid từ Occupancy Network có thể được dùng như một ràng buộc (constraint) cho Path Planning Head của HydraNet, đảm bảo lộ trình chỉ đi qua vùng trống.
- Dự đoán vật thể từ HydraNet (như xe ở ô (10, 4)) giúp Occupancy Network xác nhận trạng thái chiếm dụng.

#### (d) Occupancy Network có khả năng cao là một head của HydraNet

- Lý do chính:

- Tesla ưu tiên hiệu quả tính toán trên chip FSD, và một mạng đa nhiệm như HydraNet với occupancy làm một head sẽ tối ưu hơn về tài nguyên và tốc độ.
- Trong AI Day 2021, HydraNet được mô tả như một mạng xử lý nhiều tác vụ chính của FSD, và occupancy prediction là một tác vụ cốt lõi, phù hợp để tích hợp.
- Các phương pháp như BEVFormer và CenterPoint (tích hợp occupancy vào mạng chính) là xu hướng phổ biến, và Tesla có thể đi theo hướng này để đơn giản hóa pipeline.

- Cách hoạt động:

- BEV feature map vào HydraNet.
- Trunk xử lý, tạo đặc trưng chung.
- Occupancy Head (Occupancy Network) tạo Occupancy Grid (ô nào trống, ô nào chiếm dụng).
- Các heads khác (như path planning) dùng Occupancy Grid để lập kế hoạch.

⇒ Ưu điểm: Hiệu quả, nhanh, và tích hợp chặt chẽ.

## 7) THỰC NGHIỆM VÀ ĐÁNH GIÁ

**\*Lưu ý tất cả các thực nghiệm phải chạy trên GPU t100\***

### 7.1 Dataset:

- Udacity Object Detection Dataset - Part 2-1:

- Kích thước ảnh: 1920x1080 (Full HD).
- Số lượng ảnh: ~4070 khung hình.
- Loại đối tượng: chủ yếu là car.
- Môi trường: Đường phố, ánh sáng ban ngày.

- Số đối tượng/ảnh: Trung bình từ 2–10 xe mỗi khung hình.
  - Loại dữ liệu: Ảnh màu RGB trích xuất từ video quay bởi xe tự hành.
  - Kèm theo nhãn (label) dạng bounding box.
- Tập cardetection:
- Gồm: 1 video mp4 và 1 file ảnh chia làm tập train và test.
  - Loại đối tượng chủ yếu: các biển báo.
  - Môi trường: Ánh sáng tốt.
  - Loại dữ liệu: Ảnh RGB.
  - Kèm theo nhãn (label).

## 7.2 Các tác vụ:

### 7.2.1 Thực nghiệm nhận diện làn đường:

- Radius of curvature tại một điểm trên đường cong là bán kính của đường tròn tiếp xúc và có độ cong giống đường cong tại điểm đó.



- Nếu đường cong càng cong, thì bán kính độ cong càng nhỏ.

- Nếu đường cong gần như thẳng, thì bán kính độ cong rất lớn, tiến tới vô cực nếu là đường thẳng hoàn toàn.

- Trên hình trên nếu độ cong  $\geq 6000$  thì là đường thẳng, nếu dưới là đường cong. Còn độ lệch làn khá nhỏ và chính xác.

### 7.2.2 Thực nghiệm nhận diện, phân loại và truy vết vật thể:

- Ban đầu chúng tôi thử nghiệm trên mô hình SSD để nhận diện các phương tiện di chuyển, tuy nhiên độ chính xác không được cao chỉ khoảng ở mức 0,8 và không bounding box đủ tất cả các vật thể trong khung hình. Sau khi chuyển sang sử dụng YOLOv8 thì độ chính xác tăng lên đáng kể hơn đến 0,9.

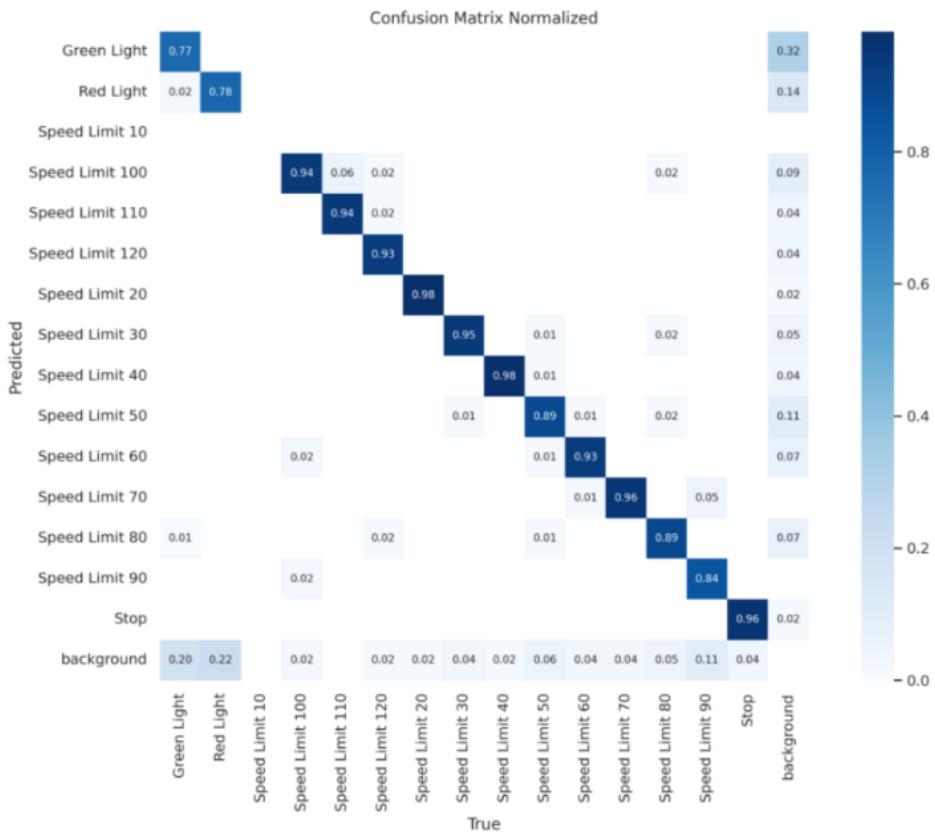


Hình 7: YOLOv8



Hình 8: SSD

- Đối với tác vụ phân loại biển báo: ban đầu thực nghiệm trên LeNet và kết quả đưa ra khá chính xác nhưng chỉ nằm trong top 4. Tuy nhiên trên thực tế ta có rất nhiều loại biển báo và đèn giao thông nên ta cần một mô hình có độ chính xác cao hơn và chúng tôi đã lựa chọn YOLOv8. Sau khi thực nghiệm trên tập dữ liệu chúng tôi thu được kết quả khá tốt được biểu thị bằng ma trận bên dưới



- Các giá trị trên đường chéo chính (ví dụ: 0.77 cho Green Light, 0.98 cho Speed Limit 20, v.v.) cho thấy mô hình có khả năng phân loại chính xác cao đối với hầu hết các lớp. Các giá trị này đều gần với 1, chứng tỏ mô hình đang phân loại đúng đắn số các trường hợp.

- Speed Limit 20 và Speed Limit 30 có tỷ lệ chính xác cao nhất (0.98 và 0.95 tương ứng), cho thấy mô hình phân loại tốt nhất với các giới hạn tốc độ này. Green Light và Red Light có sự nhầm lẫn cao với nhau (0.22 cho Green Light bị dự đoán là Red Light và ngược lại với tỷ lệ 0.78 cho Red Light bị dự đoán là Green Light). Điều này cho thấy mô hình có thể gặp khó khăn trong việc phân biệt giữa đèn xanh và đèn đỏ, có thể là do chúng trông rất giống nhau trong hình ảnh hoặc các đặc trưng không rõ ràng.

- Speed Limit 100 và Speed Limit 110 có ít sự nhầm lẫn, với tỷ lệ rất thấp trong các lớp khác. Điều này cho thấy mô hình phân loại tốt và có độ chính xác cao với các giới hạn tốc độ này.

- Lớp "background" cũng được phân loại tốt với tỷ lệ nhầm lẫn thấp (0.02), điều này có nghĩa là mô hình có thể phân biệt tốt giữa các đối tượng và các khu vực nền không có thông tin.

⇒ Mô hình hoạt động khá tốt với đa số các lớp, nhưng có thể cần cải thiện khả năng phân biệt giữa các lớp có sự tương đồng cao như "Green Light" và "Red Light". Bạn có thể thử cải thiện độ phân biệt giữa các lớp này bằng cách tăng cường dữ liệu hoặc sử dụng các kỹ thuật học sâu phù hợp hơn.

### 7.2.3 Thực nghiệm phân đoạn ngữ nghĩa:

- Dưới đây là một số kết quả phân đoạn ngữ nghĩa sử dụng Unet đối với các phương tiện:



- Ưu điểm:

- Mô hình nhận diện được phần lớn các đối tượng (xe ô tô) trên đường.
- Hình dạng mask khá khớp với đối tượng, cho thấy U-Net hoạt động tốt trong việc nhận diện biên dạng.
- Phân tách được nhiều đối tượng trong một khung hình, ngay cả khi các đối tượng gần nhau hoặc nhỏ

- Hạn chế:

- Một số đối tượng bị bỏ sót:
  - Có một vài ô tô ở xa bên phải không được gán mask dù có mặt trong Ground Truth.
  - Tương tự, hình hàng 3 có một số xe bên trái cũng bị bỏ qua.

- Mask không bao phủ hết kích thước đối tượng: Một vài mask hơi nhỏ hơn so với kích thước thực tế của ô tô, ví dụ như hình đầu tiên (xe ở giữa), có thể ảnh hưởng đến độ chính xác IoU (Intersection over Union).
- Lỗi nhận diện ở vùng sáng chói hoặc vùng tối:
  - Hình hàng 3 có ánh sáng gắt từ bên trái (chữ "STOP" trên mặt đường bị lóa), dẫn đến việc bỏ sót xe ở gần đó.
  - Điều này cho thấy mô hình có thể nhạy cảm với điều kiện ánh sáng không ổn định.
- Dưới đây là một số kết quả phân đoạn ngữ nghĩa sử dụng Unet đối với làn đường:



- Phân đoạn (segmentation) làn đường khá chính xác:
  - Vùng màu xanh bao phủ đúng phần mặt đường xe có thể chạy được.
  - Các ranh giới hai bên làn đường được xác định tương đối tốt, không bị lem sang vỉa hè hay khu vực đậu xe.
- Tách biệt tốt với các vật thể bên đường:
  - Mặt đường không bị dính vào các xe đậu bên phải hoặc bên trái.
  - Cho thấy mô hình segmentation hoạt động tốt trong môi trường đô thị có nhiều xe cộ.
- Phân đoạn có tính liên tục, không bị đứt đoạn: Vùng xanh liền mạch từ gần đến xa, phù hợp cho ứng dụng như điều hướng hoặc tự lái.

## 8) AN TOÀN VÀ ĐỘ TIN CẬY

- Đây là bài báo cáo về vai trò và tầm quan trọng của thị giác máy tính trong xe tự lái nên sẽ không trình bày chi tiết biện pháp an toàn và kiểm định an toàn của các hãng xe trên. Báo cáo này chỉ cung cấp một phân tích tổng quát về cách xe tự lái xử lý lỗi hệ thống, phản ứng khi xảy ra lỗi, các phương pháp khắc phục, và các ví dụ thực tế.

## **8.1 Xử Lý Lỗi Hệ Thống: Cơ Chế và Thách Thức:**

- Khi xe tự lái gặp lỗi, chẳng hạn như cảm biến LIDAR không hoạt động hoặc thuật toán dự đoán sai, hệ thống cần phát hiện và xử lý ngay lập tức. Theo NHTSA[36], Object and Event Detection and Response (OEDR) là một yếu tố quan trọng, đảm bảo xe có thể phát hiện các vật thể và sự kiện bất thường, như xe khác cắt ngang hoặc chướng ngại vật. Giám sát thời gian thực, chẳng hạn như phân tích dữ liệu từ cảm biến radar, camera, và GPS, giúp phát hiện lỗi ngay khi chúng xảy ra. Nếu lỗi nghiêm trọng, hệ thống có thể chuyển sang chế độ an toàn, như dừng xe ở lề đường, được gọi là "Fall Back (Minimal Risk Condition)." Điều này đặc biệt quan trọng ở các cấp độ tự lái cao (Level 4 và 5), nơi không có tài xế can thiệp.

- Các loại lỗi phổ biến bao gồm:

- Lỗi phần mềm: Dự đoán sai chuyển động.
- Lỗi cảm biến: Cảm biến bị che khuất hoặc không hoạt động, dẫn đến mất dữ liệu quan trọng.
- Lỗi phần cứng: Hỗn hót ở bộ xử lý hoặc bộ nhớ, ảnh hưởng đến hiệu suất tổng thể.

- Việc xử lý lỗi không chỉ là kỹ thuật mà còn liên quan đến quyết định kinh doanh, như liệu có cần thu hồi toàn bộ đội xe hay chỉ một số xe cụ thể, tùy thuộc vào mức độ ảnh hưởng.

## **8.2 Phản Ứng Khi Xảy Ra Lỗi: Chế Độ An Toàn và Giám Sát Thời Gian Thực:**

- Khi phát hiện lỗi, hệ thống xe tự lái thường chuyển sang chế độ an toàn, chẳng hạn như giảm tốc độ, dừng xe ở lề đường, hoặc cảnh báo người lái nếu xe ở cấp độ tự lái thấp (Level 2 hoặc 3). Theo NHTSA, "Fall Back (Minimal Risk Condition)" đảm bảo xe có thể tự đưa mình vào trạng thái rủi ro tối thiểu, như dừng xe an toàn mà không gây cản trở giao thông. Ví dụ, nếu cảm biến không hoạt động, hệ thống có thể sử dụng dữ liệu từ cảm biến khác, như radar thay cho LIDAR, để duy trì hoạt động, như Tesla đã làm trong một số trường hợp.

- Giám sát thời gian thực, chẳng hạn như OEDR, sử dụng dữ liệu từ nhiều cảm biến để phát hiện lỗi ngay lập tức, như dự đoán sai chuyển động của xe khác hoặc chướng ngại vật bất ngờ. Nếu xe ở cấp độ tự lái cao, hệ thống sẽ tự động xử lý, nhưng ở cấp độ thấp, nó sẽ cảnh báo tài xế, chẳng hạn qua âm thanh hoặc yêu cầu đặt tay lên vô-lăng, để đảm bảo an toàn.

## **8.3 Các Phương Pháp Khắc Phục: Cập Nhật OTA và Thu Hồi:**

- Để khắc phục lỗi, các nhà sản xuất thường sử dụng cập nhật phần mềm qua OTA, một phương pháp hiệu quả và tiết kiệm chi phí. OTA cho phép gửi bản vá lỗi trực tiếp đến xe mà không cần mang xe đến trung tâm bảo dưỡng, như Tesla đã làm khi thu hồi 362,000 xe vào năm 2023 để sửa lỗi phần mềm

Full Self-Driving Beta[37]. Waymo cũng sử dụng OTA để cập nhật phần mềm sau các vụ va chạm, như trường hợp 444 xe vào năm 2024[38].

- Tuy nhiên, nếu lỗi nghiêm trọng và không thể sửa qua OTA, nhà sản xuất có thể thực hiện thu hồi. Thu hồi có thể áp dụng cho toàn bộ đội xe nếu lỗi ảnh hưởng rộng, như Tesla, hoặc chỉ một số xe cụ thể nếu lỗi giới hạn ở một lô sản xuất, như Cruise thu hồi 950 xe vào năm 2023 sau vụ tai nạn người đi bộ ở San Francisco[39]. Quyết định thu hồi phụ thuộc vào mức độ nghiêm trọng, như lỗi gây tai nạn hoặc ảnh hưởng đến an toàn giao thông.

Bảng 6: Bảng So Sánh Các Trường Hợp Thu Hồi Thực Tế

Công Ty	Số Lượng Xe Thu Hồi	Năm	Lý Do Thu Hồi	Phương Pháp Khắc Phục
Waymo	444	2024	Lỗi dự đoán sai chuyển động xe kéo	Cập nhật phần mềm OTA
Tesla	362,000	2023	Lỗi phần mềm Full Self-Driving Beta	Cập nhật phần mềm OTA
Cruise	950	2023	Lỗi phần mềm sau tai nạn người đi bộ	Cập nhật phần mềm, thu hồi

#### 8.4 Tổng kết:

- An toàn và độ tin cậy của xe tự lái phụ thuộc vào khả năng xử lý lỗi hiệu quả, từ phát hiện qua giám sát thời gian thực đến phản ứng qua chế độ an toàn và khắc phục qua OTA hoặc thu hồi. Các ví dụ thực tế như Waymo, Tesla, và Cruise cho thấy công nghệ này đang phát triển, nhưng vẫn đối mặt với thách thức, đặc biệt trong các tình huống hiếm gặp (edge cases). Nghiên cứu tiếp tục tập trung vào cải thiện thuật toán, cảm biến, và tiêu chuẩn an toàn, như hướng dẫn của NHTSA, để đảm bảo xe tự lái có thể hoạt động an toàn trên đường phố.

### 9) THÁCH THỨC VÀ HẠN CHẾ CỦA THỊ GIÁC MÁY TÍNH TRONG XE TỰ LÁI

- Computer vision là công nghệ cốt lõi trong xe tự lái, cho phép xe nhận biết và phản ứng với môi trường xung quanh thông qua hình ảnh từ camera. Tuy nhiên, công nghệ này vẫn đối mặt với nhiều thách thức, đặc biệt trong các điều kiện thời tiết và môi trường phức tạp.

#### 1. Thách thức chính trong computer vision cho xe tự lái:

Bảng 7: Thách thức chính trong computer vision cho xe tự lái

Thách thức	Mô tả chi tiết	Ví dụ cụ thể
Điều kiện thời tiết xấu	Mưa lớn, sương mù, tuyết làm mờ hình ảnh, giảm khả năng phát hiện đối tượng và làn đường	Mưa tạo phản chiếu, sương mù giảm độ tương phản, tuyết che dấu hiệu giao thông
Môi trường đô thị phức tạp	Giao thông đông đúc, người đi bộ, xe đạp, và biển báo đa dạng gây khó khăn trong nhận diện	Xử lý ngã tư nhiều làn, vòng xuyến, hành vi không lường trước của phương tiện
Điều kiện ánh sáng yếu	Ban đêm hoặc khu vực thiếu sáng làm giảm độ tương phản, ánh hưởng đến phát hiện đối tượng	Không phát hiện được người đi bộ hoặc biển báo trong bóng tối
Thu thập dữ liệu huấn luyện	Cần dữ liệu đa dạng từ nhiều điều kiện, nhưng thu thập và gắn nhãn tốn thời gian, chi phí	Dữ liệu thiếu về thời tiết hiếm như bão tuyết
Xử lý thời gian thực	Yêu cầu xử lý hình ảnh nhanh để đưa ra quyết định tức thì, đòi hỏi phần cứng mạnh mẽ	Phải phản ứng nhanh trong tình huống khẩn cấp
Phát hiện và phân loại đối tượng	Khó khăn trong nhận diện đối tượng bị che khuất, kích thước khác nhau, hoặc trông giống nhau	Phân biệt xe tải và xe con trong điều kiện đông đúc
Dộ tin cậy trước tấn công đối nghịch	Hệ thống dễ bị lừa bởi các thay đổi nhỏ trong hình ảnh, gây phân loại sai	Thay đổi nhỏ trên biển báo làm xe hiểu sai

## 2. Phân tích thêm:

### (a) **Điều kiện thời tiết xấu:**

- Theo báo cáo từ Weather Creates Challenges For Next Generation Of Vehicles[40], mưa, sương mù, và tuyết làm giảm hiệu suất của camera và cảm biến, đặc biệt khi chúng không thể "nhìn thấy" các dấu hiệu đường. Một nghiên cứu từ How Self-Driving Cars Handle Inclement Weather?[41] cho thấy Tesla's Autopilot gặp khó khăn trong mưa lớn, không duy trì được làn đường.
- Ngoài ra, ánh sáng chói từ mặt trời, như được đề cập trong Applications of Computer Vision in Autonomous Vehicles[42], cũng gây nhiễu cho camera, làm giảm khả năng nhận diện.

### (b) **Môi trường đô thị phức tạp:**

- Trong các thành phố lớn, xe tự lái phải xử lý mật độ giao thông cao và hành vi không lường trước. Ví dụ, hệ thống cần nhận diện và dự đoán hành vi của người đi bộ tại ngã tư, điều này

đòi hỏi khả năng phân tích cảnh phức tạp.

(c) **Điều kiện ánh sáng yếu:**

- Vào ban đêm, hình ảnh từ camera có độ tương phản thấp, làm giảm khả năng phát hiện đối tượng. Nghiên cứu cho thấy các thuật toán cần cải thiện để xử lý ánh sáng yếu, tránh nguy cơ tai nạn.

(d) **Các thách thức bổ sung:**

- Ngoài các điểm trên, việc thu thập dữ liệu huấn luyện đại diện là một vấn đề lớn, vì cần dữ liệu từ nhiều điều kiện thời tiết và địa điểm khác nhau. Xử lý thời gian thực cũng đòi hỏi phần cứng mạnh mẽ, tăng chi phí. Phát hiện và phân loại đối tượng, đặc biệt trong điều kiện che khuất, là một thách thức khác. Cuối cùng, độ tin cậy trước tấn công đối nghịch, như thay đổi nhỏ trên hình ảnh gây phân loại sai, là một mối quan ngại an toàn.

**3. Hướng giải quyết:**

- Kết hợp cảm biến khác: Sử dụng LiDAR và radar để bổ sung cho camera, như LiDAR tạo bản đồ 3D chi tiết, radar phát hiện đối tượng trong mọi điều kiện thời tiết bởi vì LiDAR có thể giúp định vị khi lùn đường bị che bởi tuyết[43].

- Cải thiện thuật toán: Phát triển mô hình deep learning với data augmentation, như tạo dữ liệu giả cho điều kiện thời tiết xấu, và tối ưu hóa để chạy trên phần cứng nhúng.

- Sử dụng bản đồ chi tiết: Bản đồ độ nét cao (HD maps) giúp xe định vị chính xác, ngay cả khi visual bị che lấp.

- Nâng cấp công nghệ cảm biến: Nghiên cứu radar giống LiDAR để xuyên qua sương mù và mưa, như được nghiên cứu tại Đại học California San Diego[44].

## 10) KẾT LUẬN VÀ DỰ ĐOÁN

### 10.1 Tổng hợp:

- Thị giác máy tính là một thành phần cốt lõi trong công nghệ xe tự lái, đóng vai trò như "đôi mắt" của phương tiện, cho phép chúng nhận diện và hiểu môi trường xung quanh. Thông qua camera và cảm biến, nó thu thập hình ảnh và video, sau đó xử lý để đưa ra quyết định lái xe an toàn trong nhiều điều kiện khác nhau. Các nhiệm vụ cụ thể bao gồm:

- Phát hiện vật thể: Nhận diện xe cộ, người đi bộ, biển báo và chướng ngại vật trong thời gian thực, sử dụng các thuật toán như YOLO (You Only Look Once) để xử lý nhanh chóng.
- Theo dõi vật thể: Giám sát các vật thể di động, chẳng hạn như xe khác và người đi bộ, để lập kế hoạch đường đi và tránh va chạm, thường sử dụng Deep SORT để duy trì nhận diện ngay cả khi bị che khuất.

- Phân đoạn ngữ nghĩa: Chia hình ảnh thành các pixel được gán nhãn (đường, người đi bộ, v.v.) để hiểu chi tiết môi trường, hỗ trợ điều hướng chính xác.
- Ước lượng độ sâu: Đo khoảng cách đến các vật thể, cải thiện nhận thức về cấu trúc 3D, thường sử dụng camera stereo hoặc kỹ thuật ánh sáng.
- Phân tích dữ liệu LIDAR: Tạo bản đồ 3D để đo khoảng cách và tốc độ chính xác, đáng tin cậy trong mọi điều kiện ánh sáng.
  - Sự tích hợp của thị giác máy tính với các cảm biến khác như radar, GPS và cảm biến siêu âm tạo ra một bức tranh toàn diện về môi trường, đảm bảo an toàn và hiệu quả. Đặc biệt, việc sử dụng học sâu, chẳng hạn như mạng nơ-ron tích chập, đã nâng cao độ chính xác trong nhận diện hình ảnh, cho phép xử lý dữ liệu trong thời gian thực. Ví dụ, trong phát hiện làn đường, các lớp tích chập học các đặc điểm, trong khi các lớp cuối cùng tính toán hệ số phương trình làn đường, hỗ trợ điều hướng chính xác.
  - Tầm quan trọng của thị giác máy tính nằm ở khả năng giúp xe tự lái đưa ra quyết định an toàn, đặc biệt trong các tình huống phức tạp như giao thông đông đúc hoặc điều kiện thời tiết xấu. Tuy nhiên, vẫn còn nhiều thách thức:
- Cảm biến: Không cảm biến nào hoàn hảo; cần nhiều loại cảm biến (radar, LIDAR) để đảm bảo dự phòng. Radar hoạt động tốt trong sương mù hoặc mưa nhưng không chi tiết hình dạng, trong khi LIDAR bị ảnh hưởng bởi thời tiết và có tầm ngắn. Các công cụ mô phỏng như Ansys giúp tối ưu hóa thiết kế cảm biến, chẳng hạn mô phỏng ảnh hưởng của sương mù lên LIDAR.
- Trí tuệ nhân tạo nhận thức: Cảm biến tạo ra khối lượng dữ liệu lớn cần xử lý thời gian thực thông qua học máy và AI để nhận diện đối tượng như biển dừng, người chạy bộ. Đào tạo trong thế giới thực tốn kém và không an toàn; mô phỏng Ansys cung cấp cách đào tạo nhanh, an toàn và tiết kiệm chi phí.
- Quyết định an toàn: Xe phải đưa ra quyết định an toàn dựa trên hàng ngàn tham số (giao thông, người đi bộ, thời tiết, giao tiếp V2X). An toàn đòi hỏi thử nghiệm, thiết kế lại và xác nhận thuật toán trên hàng tỷ kịch bản lái xe, chỉ khả thi qua thử nghiệm vòng lặp phần cứng với Ansys.
  - Những thách thức này nhấn mạnh rằng, mặc dù thị giác máy tính đã tiến bộ, việc đạt được cấp độ tự chủ 5, nơi xe hoạt động hoàn toàn độc lập mà không cần sự can thiệp của con người, vẫn là một mục tiêu phức tạp và xa vời.

## 10.2 Dự đoán:

- Dự đoán tương lai cho thấy sự phát triển của AI và thị giác máy tính sẽ thúc đẩy xe tự lái tiến gần hơn đến cấp độ 5. Các xu hướng chính bao gồm:

- Cải tiến cảm biến: LIDAR và radar tiên tiến hơn, kết hợp với camera chất lượng cao, sẽ cải thiện khả năng nhận thức trong mọi điều kiện, chẳng hạn như sương mù, mưa, hoặc ban đêm.

- AI và học máy mạnh mẽ hơn: Các thuật toán học sâu, như mạng nơ-ron tích chập, sẽ cải thiện nhận diện đối tượng và dự đoán hành vi của người dùng đường khác. Việc tích lũy dữ liệu lái xe từ các phương tiện trên thực tế sẽ hỗ trợ đào tạo mô hình học máy, giúp xử lý tốt hơn các kịch bản không thể đoán trước.
  - Tính toán biên và truyền dữ liệu nhanh hơn: Sự phát triển của 5G và tính toán biên sẽ hỗ trợ truyền dữ liệu nhanh hơn cho giao tiếp V2X (xe với mọi thứ), cung cấp thông tin bổ sung để hỗ trợ quyết định.
  - Tích hợp với cơ sở hạ tầng thông minh: Thành phố thông minh với hệ thống giao thông thông minh sẽ hỗ trợ xe tự lái, giảm ùn tắc và cải thiện an toàn.
- Mặc dù vậy, đạt được cấp độ 5 vẫn đòi hỏi nhiều rào cản, bao gồm:
- Thách thức kỹ thuật: Cần xử lý khôi lượng dữ liệu lớn trong thời gian thực, đòi hỏi sức mạnh tính toán vượt trội.
  - Rào cản quy định: Cần cập nhật luật pháp để cho phép xe tự lái hoạt động hoàn toàn, đặc biệt ở các khu vực đông đúc như thành phố.
  - Chấp nhận xã hội: Công chúng cần tin tưởng vào công nghệ, đặc biệt khi lo ngại về mất việc làm, quyền riêng tư và an ninh mạng.

- Theo một báo cáo từ GlobalData[45], triển khai có ý nghĩa của xe tự lái cấp độ 5 có thể xảy ra trước năm 2035, do lạm phát tăng, ảnh hưởng từ cuộc xâm lược Ukraine của Nga lên ngành bán dẫn, và chi phí cao cho cơ sở hạ tầng. Tuy nhiên, các công ty như Waymo đang thử nghiệm dịch vụ robotaxi ở cấp độ 4 tại Arizona và California, cho thấy tiến bộ đáng kể.

### **10.3 Kết luận:**

- Tóm lại, thị giác máy tính là nền tảng cho xe tự lái, và sự phát triển của AI sẽ tiếp tục thúc đẩy tiến bộ. Mặc dù cấp độ 5 vẫn là mục tiêu xa, với dự đoán triển khai có ý nghĩa có thể đến năm 2035, các tiến bộ trong công nghệ đang dần thu hẹp khoảng cách. Xe tự lái không chỉ hứa hẹn cải thiện an toàn và hiệu quả mà còn mang lại lợi ích xã hội sâu rộng, cách mạng hóa cách chúng ta di chuyển.

## References

- [1] Anzhella Pankratova. Computer Vision in Self-Driving Cars. Nov. 19, 2023. URL: <https://www.opencv.ai/blog/computer-vision-in-self-driving-cars>.
- [2] AUTOPILOT REVIEW. SAE Self-Driving Levels 0 to 5 for Automation – What They Mean. URL: <https://www.autopilotreview.com/self-driving-cars-sae-levels/>.
- [3] WIKIPEDIA. Self-driving car. URL: [https://en.wikipedia.org/wiki/Self-driving\\_car](https://en.wikipedia.org/wiki/Self-driving_car).
- [4] Darrell Etherington. It's time to admit self-driving cars aren't going to happen. TechCrunch, Oct. 27, 2022. URL: <https://techcrunch.com/2022/10/27/self-driving-cars-arent-going-to-happen/>.
- [5] Sean Tucker. Self-Driving Cars: Everything You Need To Know. Kelley Blue Book, Sept. 1, 2024. URL: <https://www.kbb.com/car-advice/self-driving-cars/>.
- [6] Richard K. Hy. Tesla's Self-Driving Features- What's the Difference? EGLET LAW, Feb. 28, 2024. URL: <https://www.egletlaw.com/teslas-self-driving-features-whats-the-difference/>.
- [7] Tesla Team. Autopilot and Full Self-Driving (Supervised). TESLA. URL: <https://www.tesla.com/support/autopilot>.
- [8] autopilot review. Tesla Hardware 3 (Full Self-Driving Computer) Detailed. URL: <https://www.autopilotreview.com/tesla-custom-ai-chips-hardware-3/>.
- [9] autopilot review. Tesla Hardware 4 (AI4) – Full Details and Latest News. URL: <https://www.autopilotreview.com/tesla-hardware-4-rolling-out-to-new-vehicles/>.
- [10] WAYMO. URL: <https://waymo.com/waymo-driver/>.
- [11] How our cars drive. GOOGLE. URL: <https://support.google.com/waymo/answer/9190838?hl=en>.
- [12] Waymo. Sense, Solve, and Go: The Magic of the Waymo Driver. 2023. URL: [https://www.youtube.com/watch?v=hA\\_-MkUONfw](https://www.youtube.com/watch?v=hA_-MkUONfw).
- [13] CRUISE. “The Cruise Origin: Driverless Testing Program Guidance for First Responders”. In: Cruise Origin - Law Enforcement Interaction Plan - 02.2024 (February 2024). URL: [https://downloads.ctfassets.net/95kuvdv8zn1v/qShYj4CQSqzWoLNtnCHmq/323d186bfe2cd33d9f777f501b3f38fd/Cruise\\_Origin\\_-\\_Law\\_Interaction\\_Plan\\_-02.2024\\_.pdf](https://downloads.ctfassets.net/95kuvdv8zn1v/qShYj4CQSqzWoLNtnCHmq/323d186bfe2cd33d9f777f501b3f38fd/Cruise_Origin_-_Law_Interaction_Plan_-02.2024_.pdf).
- [14] Stephen Edelstein. How GM's Cruise self-driving cars navigate around double-parked vehicles. digitaltrends, July 3, 2019. URL: <https://www.digitaltrends.com/cars/gm-cruise-self-driving-cars-navigate-around-double-parked-vehicles/>.
- [15] Erin Antcliffe. 3 Ways Cruise HD Maps Give Our Self-Driving Vehicles An Edge. Medium, Nov. 13, 2019. URL: <https://medium.com/cruise/hd-maps-self-driving-cars-b6444720021c>.

- [16] Caleb Miller. GM's Cruise Ditches Origin Robotaxi for a Self-Driving Next-Gen Bolt. CARAND-DRIVER, July 24, 2024. URL: <https://www.caranddriver.com/news/a61677537/gm-cruise-origin-robotaxi-dead/>.
- [17] Dan Ammann. THE CRUISE ORIGIN STORY. Position: CEO, Cruise. IoT AUTOMOTIVE NEWS. URL: <https://iot-automotive.news/the-cruise-origin-story/>.
- [18] NVIDIA DRIVE Hyperion Platform Achieves Critical Automotive Safety and Cybersecurity... NVIDIA, Jan. 6, 2025. URL: <https://nvidianews.nvidia.com/news/nvidia-drive-hyperion-platform-achieves-critical-automotive-safety-and-cybersecurity-milestones-for-av-development>.
- [19] Michael Kahn. NVIDIA's Automotive Platform Could Transform the Future of Robotic Vehicles. The Weekly Driver, Mar. 3, 2025. URL: <https://theweeklydriver.com/2025/03/nvidias-automotive-platform-transforms-the-future-of-robotic-vehicles/>.
- [20] PYMNTS. Nvidia and AWS Work on Autonomous Vehicle Tech; How Long Before Consumers Trust It? PYMNTS, Jan. 7, 2025. URL: <https://www.pymnts.com/news/artificial-intelligence/2025/nvidia-ai-boost-development-autonomous-vehicle-tech/>.
- [21] DPCCars. NVIDIA CEO Jensen Huang AI Autonomous Vehicle Development at CES 2025. 2025. URL: <https://www.youtube.com/watch?v=vqmW1BQ1cbw>.
- [22] End-to-End Solutions for Autonomous Vehicles. NVIDIA DEVELOPER. URL: <https://developer.nvidia.com/drive>.
- [23] Driving Next-Generation AV Breakthroughs. NVIDIA. URL: <https://www.nvidia.com/en-us/self-driving-cars/in-vehicle-computing/>.
- [24] Peide Wang. “Research on Comparison of LiDAR and Camera in Autonomous Driving”. In: Journal of Physics: Conference Series. Vol. 2093. Zhuhai, China: IOP Publishing Ltd, Sept. 24–26, 2021, p. 012032. DOI: [10.1088/1742-6596/2093/1/012032](https://doi.org/10.1088/1742-6596/2093/1/012032).
- [25] Yan Wang, Wei-Lun Chao, and Divyansh Garg et al. “Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019, pp. 9120–9129. URL: [https://openaccess.thecvf.com/content\\_CVPR\\_2019/html/Wang\\_Pseudo-LiDAR\\_From\\_Visual\\_Depth\\_Estimation\\_Bridging\\_the\\_Gap\\_in\\_3D\\_CVPR\\_2019\\_paper.html](https://openaccess.thecvf.com/content_CVPR_2019/html/Wang_Pseudo-LiDAR_From_Visual_Depth_Estimation_Bridging_the_Gap_in_3D_CVPR_2019_paper.html).
- [26] S. Shalev-Shwartz, S. Shammah, and A. Shashua. “Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving”. In: arXiv arXiv:1610.03295 (2016). URL: <https://arxiv.org/abs/1610.03295>.
- [27] S. D. Pendleton et al. “Perception, Planning, Control, and Coordination for Autonomous Vehicles”. In: Machines 5.1 (2017), p. 6. DOI: [10.3390/machines5010006](https://doi.org/10.3390/machines5010006).

- [28] D. A. Pomerleau. “ALVINN: An Autonomous Land Vehicle in a Neural Network”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 1989, pp. 305–313. URL: <https://proceedings.neurips.cc/paper/1988/hash/812b4ba287f5ee0bc9d43bbf5bbe87fb-Abstract.html>.
- [29] L. Yu et al. “Intelligent Land-Vehicle Model Transfer Trajectory Planning Method Based on Deep Reinforcement Learning”. In: *Sensors* 18.9 (2018), p. 2905. DOI: 10.3390/s18092905.
- [30] Tianhao Zhang et al. “Learning Deep Control Policies for Autonomous Aerial Vehicles with MPC-guided Policy Search”. In: *(ICRA)* (May 2016), pp. 528–535. DOI: 10.1109/ICRA.2016.7487175.
- [31] V. Mnih et al. “Human-level Control Through Deep Reinforcement Learning”. In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. DOI: 10.1038/nature14236.
- [32] Hao Zhu et al. “Overview of Environment Perception for Intelligent Vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems*. Vol. 18. 10. 2017, pp. 2584–2601. DOI: 10.1109/TITS.2017.2658662.
- [33] Ilija Radosavovic et al. “Designing Network Design Spaces”. In: *Computer Vision Foundation (CVF)* (2020), pp. 10428–10435. URL: [https://openaccess.thecvf.com/content\\_CVPR\\_2020/papers/Radosavovic\\_Designing\\_Network\\_Design\\_Spaces\\_CVPR\\_2020\\_paper.pdf](https://openaccess.thecvf.com/content_CVPR_2020/papers/Radosavovic_Designing_Network_Design_Spaces_CVPR_2020_paper.pdf).
- [34] Mingxing Tan, Ruoming Pang, and Quoc V. Le. “EfficientDet: Scalable and Efficient Object Detection”. In: *arXiv* (2020). DOI: arXiv:1911.09070v7.
- [35] Ian Greer. *Tesla Full Self-Driving Technical Deep Dive*. Jan. 11, 2023. URL: [https://www.iangreer.io/tesla-fsd-technical-deep-dive/?utm\\_source=chatgpt.com#hydranets](https://www.iangreer.io/tesla-fsd-technical-deep-dive/?utm_source=chatgpt.com#hydranets).
- [36] Alexandra S. Mueller, Jessica B. Cicchino, and David S. Zuby. “What humanlike errors do autonomous vehicles need to avoid to maximize safety?” In: *Journal of Safety Research* 75 (Dec. 2020), pp. 310–318. DOI: 10.1016/j.jsr.2020.10.005.
- [37] David Shepardson. *Tesla recalls 362,000 U.S. vehicles over Full Self-Driving software*. Feb. 17, 2023. URL: <https://www.reuters.com/business/autos-transportation/tesla-recalls-362000-us-vehicles-over-full-self-driving-software-2023-02-16/>.
- [38] David Shepardson. *Waymo recalls 444 self-driving vehicles over software error*. Feb. 16, 2024. URL: <https://www.reuters.com/technology/waymo-updates-software-over-400-recalled-vehicles-nhtsa-2024-02-15/>.
- [39] Nathan Bernier. *Cruise recalls 950 self-driving vehicles over software glitch*. Nov. 8, 2023. URL: <https://www.houstonpublicmedia.org/articles/news/transportation/2023/11/08/469138/cruise-recalls-950-self-driving-vehicles-over-software-glitch/>.

- [40] Jim Foerster. Weather Creates Challenges For Next Generation Of Vehicles. Forbes, Nov. 22, 2019.  
URL: <https://www.forbes.com/sites/jimfoerster/2019/11/22/weather-creates-challenges-for-next-generation-of-vehicles/>.
- [41] How Self-Driving Cars Handle Inclement Weather? WITHERITE LAW GROUP. URL: <https://www.witheritelaw.com/news/how-self-driving-cars-handle-inclement-weather/>.
- [42] Xingshuai Dong and Massimiliano L. Cappuccio. "Applications of Computer Vision in Autonomous Vehicles: Methods, Challenges and Future Directions". In: arXiv (2024). DOI: [arXiv:2311.09093v3](https://arxiv.org/abs/2311.09093v3).
- [43] Alex Perrone. How Will Self-Driving Cars Fare in Bad Weather? ENDURANCE, Jan. 28, 2021.  
URL: <https://www.endurancewarranty.com/learning-center/tech/self-driving-cars-bad-weather/>.
- [44] Sanksshep Mahendra. Can Self-Driving Cars See in Bad Weather? Artificial Intelligence, Nov. 19, 2022. URL: <https://www.aiplusinfo.com/blog/can-self-driving-cars-see-in-bad-weather/>.
- [45] Kurt Robson. Fully self-driving cars unlikely before 2035, experts predict. Verdict, Mar. 9, 2023.  
URL: <https://www.verdict.co.uk/fully-self-driving-cars-unlikely-before-2035-experts-predict/>.