ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

NHẬN DẠNG

BÁO CÁO ĐỒ ÁN

VEHICLE TRACKING
WITH SPEED ESTIMATION AND COUNTING

GVHD: Ts. Dương Việt Hằng

Lớp: CS338.P22

Nhóm 9

Họ và tên	MSSV
Phạm Hồ Trúc Linh	22520777
Huỳnh Trung Nghĩa	22520945

□□ Tp. Hồ Chí Minh, 06/2025 □□

Mục lục

Lời mở đầu

Trong bối cảnh đô thị hóa diễn ra nhanh chóng, cơ sở hạ tầng giao thông ngày càng được đầu tư và mở rộng. Tuy nhiên, đi kèm với sự phát triển đó là hàng loạt vấn đề đáng lo ngại về an toàn giao thông như ùn tắc kéo dài, tai nạn do vi phạm tốc độ, và quá tải trong quản lý luồng phương tiện. Điều này đặt ra yêu cầu cấp thiết về các giải pháp giám sát hiệu quả, thông minh và có khả năng mở rộng trong thực tế.

Trước nhu cầu đó, các hệ thống giao thông thông minh (Intelligent Transportation Systems – ITS) đang được ứng dụng rộng rãi, đồng thời cũng cần tiếp tục được cải tiến để theo kịp thực tiễn. Đồ án này đề xuất một hệ thống giám sát giao thông ứng dụng thị giác máy tính, với khả năng tự động theo dõi, ước lượng tốc độ và đếm phương tiện từ video giám sát.

Hệ thống tích hợp các thành phần xử lý hiện đại, cho phép phát hiện và duy trì định danh phương tiện xuyên suốt các khung hình, đồng thời trích xuất thông tin vận tốc và thống kê dữ liệu như số lượng phương tiện theo loại, theo hướng di chuyển,... Hệ thống được thử nghiệm trên các video thực tế với điều kiện ánh sáng và mật độ giao thông đa dạng. Kết quả cho thấy độ chính xác cao, hoạt động ổn định và đáp ứng tốt yêu cầu thời gian thực.

Kết quả này cho thấy tiềm năng lớn của việc ứng dụng học sâu và thị giác máy tính trong giám sát và quản lý giao thông. Đồng thời, đây cũng là nền tảng để mở rộng thêm các chức năng nâng cao như phát hiện hành vi vi phạm, cảnh báo ùn tắc hoặc tích hợp vào hệ thống điều phối giao thông thông minh.

1. Giới thiệu đề tài (Introduction)

Trong bối cảnh xã hội đô thị hóa, đường phố được mở rộng và lượng phương tiện tham gia giao thông gia tăng không ngừng, việc xây dựng một hệ thống giám sát và quản lý giao thông là vô cùng cấp thiết. Các hệ thống giao thông thông minh (Intelligent Transportation Systems – ITS) đóng vai trò quan trọng trong việc quản lí và nâng cao chất lượng điều hành giao thông. Trong đó, việc **theo dõi, đếm và ước lượng tốc độ phương tiện** từ dữ liệu video là một tác vụ quan trọng mang tính nền tảng.

Tuy nhiên, bài toán này phải đối mặt với nhiều **thách thức thực tiễn** như:

- **Sự đa dạng về chủng loại phương tiện** (như ô tô con, xe máy, xe buýt, xe tải…) với **hình dạng và kích thước khác nhau** dễ dẫn đến sự nhầm lẫn trong quá trình phát hiện và phân loại, đặc biệt trong các tình huống phương tiện chen chúc hoặc che khuất lẫn nhau.
- Các yếu tố tác động từ bên ngoài như **điều kiện ánh sáng thay đổi liên tục** (giữa ngày và đêm, trong bóng râm hoặc vùng ánh sáng gắt), kết hợp với ảnh hưởng của thời tiết như **mưa lớn, sương mù hoặc ánh sáng phản chiếu từ mặt đường**, có thể làm giảm chất lượng hình ảnh đầu vào và ảnh hưởng trực tiếp đến hiệu quả xử lý của mô hình.
- **Chất lượng video từ các camera giám sát** không đồng đều (độ phân giải thấp, hình ảnh mờ, nhiễu...), cùng với **góc quay không lý tưởng**, bị khuất hoặc thay đổi theo từng vị trí lắp đặt, khiến việc theo dõi chính xác phương tiện trong toàn bộ khung hình trở nên khó khăn, khó đạt được sự đồng nhất trong toàn bộ hệ thống.
- **Sự thay đổi liên tục về tốc độ và hành vi di chuyển của phương tiện** như tăng/giảm tốc, chuyển làn hoặc quay đầu khiến hệ thống cần có khả năng thích ứng linh hoạt và theo dõi chính xác trong thời gian thực.

Những yếu tố này đòi hỏi hệ thống phải có khả năng xử lý mạnh mẽ, độ chính xác cao và hoạt động ổn định trong môi trường thực tế.

Mục tiêu của đồ án là xây dựng một hệ thống thị giác máy tính có khả năng **phát hiện phương tiện**, **theo dõi liên tục qua nhiều khung hình**, **ước lượng tốc độ** theo đơn vị thực (km/h, m/s) và **đếm phương tiện** theo chiều và loại một cách **chính xác và gần với thời gian thực**. Hệ thống được thiết kế hướng tới khả năng áp dụng trong thực tế, phù hợp cho các ứng dụng như giám sát giao lộ, thu thập thống kê giao thông, hoặc hỗ trợ phát hiện vi phạm tốc độ.

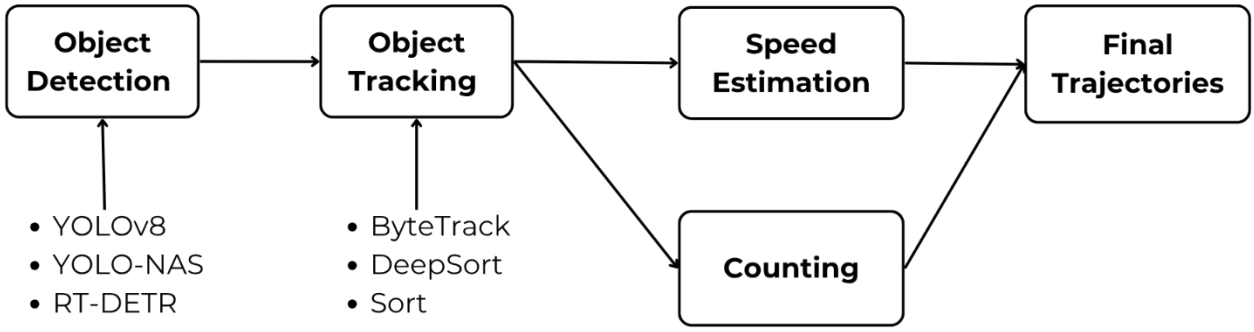
Đóng góp chính của đồ án bao gồm:

- Triển khai **pipeline hoàn chỉnh**, thử nghiệm so sánh kết quả thông qua việc kết hợp các mô hình phát hiện đối tượng mạnh mẽ (YOLOv8, YOLO-NAS, RT-DETR) với thuật toán theo dõi hiệu quả (Sort, DeepSort, ByteTrack).
- Ứng dụng **biến đổi phối cảnh** để chuyển đổi khoảng cách pixel sang đơn vị thực, từ đó ước lượng vận tốc phương tiện chính xác.
- Thiết kế **hệ thống đếm phương tiện** linh hoạt theo hướng di chuyển và phân loại, có khả năng xử lý các trường hợp đặc biệt như xe dừng hoặc quay đầu.

Hệ thống được thử nghiệm trên các video thực tế ở nhiều điều kiện môi trường khác nhau. Kết quả thực nghiệm cho thấy hệ thống đạt độ chính xác cao trong phát hiện và theo dõi phương tiện, tốc độ được ước lượng ổn định, tốc độ xử lý gần thời gian thực, thể hiện tính khả thi cao trong ứng dụng thực tế.

2. Phương pháp thực hiện (Methods)

2.1. Pipeline:



Hình 1. Pipeline tổng quan

Pipeline của hệ thống được thiết kế theo kiến trúc modular gồm bốn thành phần chính: **Phát hiện đối tượng (Object Detection)**, **Theo dõi đối tượng (Object Tracking)**, và các ứng dụng sau đó bao gồm **Ước lượng tốc độ (Speed Estimation)** và **Đếm phương tiện (Counting)**. Các kết quả từ từng giai đoạn được tổng hợp để tạo thành **Final Trajectories** – tập hợp các quỹ đạo đầy đủ kèm theo thông tin vận tốc và thống kê đếm. Cụ thể:

- **Object Detection – Phát hiện phương tiện:** Đây là bước đầu tiên trong pipeline, sử dụng các mô hình học sâu hiện đại như **YOLOv8**, **YOLO-NAS**, và **RT-DETR** để phát hiện các đối tượng là phương tiện giao thông (bao gồm xe máy, xe hơi, xe buýt, và xe tải) trong từng khung hình của video. Các mô hình này được lựa chọn do đạt hiệu quả cao cả về độ chính xác và tốc độ, đặc biệt phù hợp với yêu cầu xử lý gần thời gian thực.

- **Object Tracking – Theo dõi đối tượng:** Sau khi phát hiện, hệ thống sử dụng các thuật toán theo dõi như **ByteTrack** và **DeepSort** để gán định danh (ID) cho từng phương tiện và duy trì ID đó qua nhiều khung hình. Việc theo dõi chính xác là cơ sở cho việc tính vận tốc và đếm đúng phương tiện, đặc biệt trong các tình huống có nhiều đối tượng chồng lấp, bị che khuất.

- **Speed Estimation – Ước lượng tốc độ:** Dựa trên thông tin vị trí của từng phương tiện trong nhiều khung hình và thời gian giữa các khung, hệ thống áp dụng **biến đổi phối cảnh (homography)** để chuyển đổi khoảng cách trong ảnh sang đơn vị thực (mét, km/h), từ đó ước lượng tốc độ di chuyển thực tế của từng xe.

- **Counting – Đếm phương tiện:** Hệ thống sử dụng các **vùng đếm (counting zones)** hoặc các **đường đếm ảo (counting line)** được định nghĩa thủ công để đếm số lượng phương

tiện di chuyển qua từng khu vực theo hướng và loại. Việc kết hợp với kết quả theo dõi giúp giảm lỗi đếm trùng hoặc đếm sai khi phương tiện dừng, quay đầu hoặc di chuyển chậm.

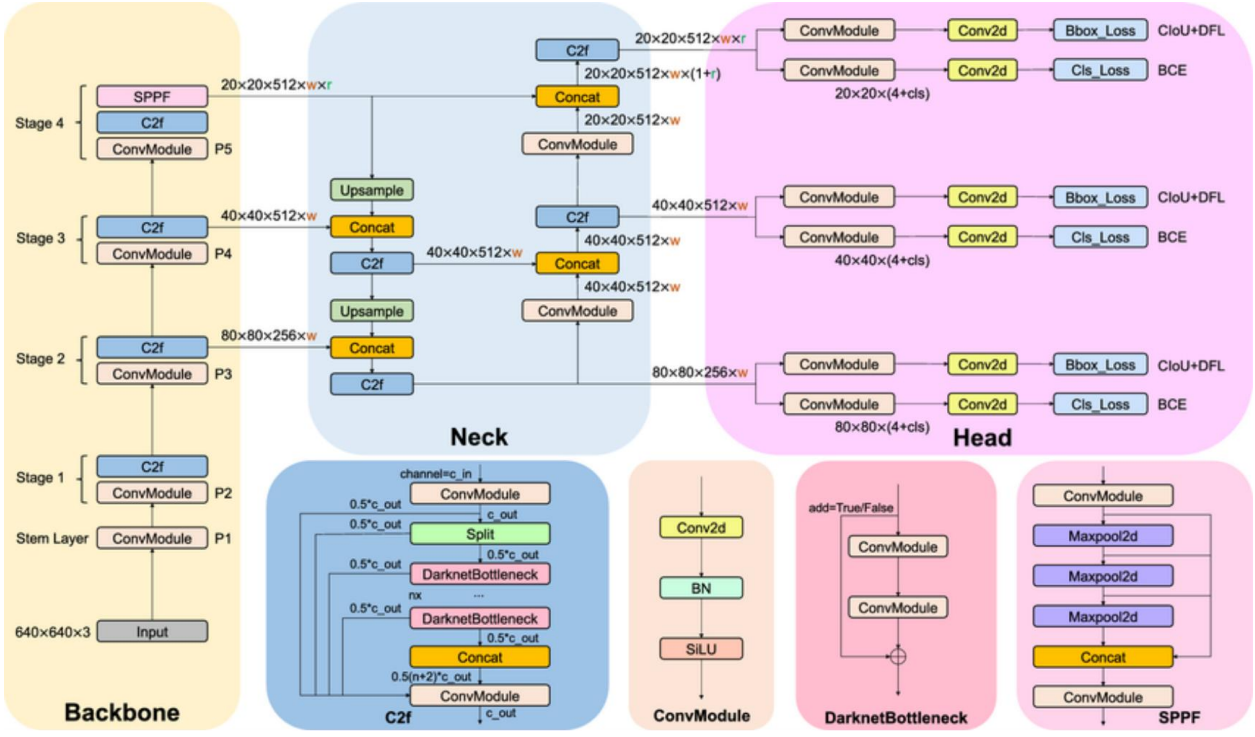
- **Final Trajectories – Tổng hợp kết quả:** Các kết quả từ ba thành phần trên được tổng hợp thành quỹ đạo cuối cùng của từng phương tiện, bao gồm các thông tin: **loại phương tiện, ID, vận tốc, và số lần đếm theo hướng đi**. Tập dữ liệu này có thể được sử dụng cho các ứng dụng như phân tích giao thông, phát hiện vi phạm hoặc làm đầu vào cho các hệ thống dự đoán tắc nghẽn.

2.2. Phát hiện đối tượng (Object detection)

Phát hiện đối tượng là bước khởi đầu và giữ vai trò then chốt trong pipeline xử lý video giao thông. Đây là giai đoạn xác định và phân loại các phương tiện giao thông trong từng khung hình, làm tiền đề, cung cấp đầu vào cho việc theo dõi đối tượng. Đồ án lựa chọn khảo sát qua 3 mô hình cùng thuộc nhà phát triển YOLO bao gồm: YOLOV8, YOLO-NAS, RT-DETR. Cả ba đều được giới thiệu trong cùng khoảng thời gian (2023–2024), kế thừa và phát triển từ dòng mô hình YOLO nổi tiếng, với mục tiêu cải thiện đồng thời hiệu năng và tốc độ xử lý.

2.2.1 YOLOv8 - (You Only Look Once v8) – CNN-based, anchor-free

YOLOv8 do Ultralytics phát triển, được ra mắt vào tháng 01 năm 2023, là thế hệ mới của dòng YOLO, tập trung vào tốc độ suy luận (inference speed) và độ chính xác cao. YOLOv8 được tối ưu tốt cho deployment trong môi trường thực tế với GPU hạn chế.



Hình 2. Kiến trúc YOLOv8

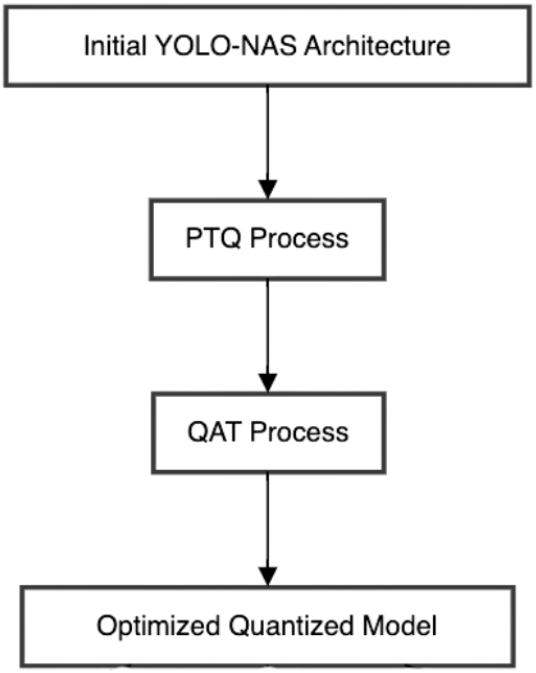
YOLOv8 có cấu trúc đơn giản gồm ba thành phần chính: **backbone** (trích xuất đặc trưng), **neck** (kết hợp thông tin từ nhiều tầng) và **head** (dự đoán đầu ra). Mô hình chia ảnh thành một lưới các ô nhỏ và dự đoán trực tiếp các bounding box, lớp đối tượng và độ tin cậy (confidence score) trên từng ô. Việc **sử dụng kiến trúc anchor-free**, giúp tăng tốc độ suy luận và cải thiện khả năng tổng quát hóa, đặc biệt trong các tình huống dữ liệu chưa từng gặp.

Ưu điểm nổi bật của YOLOv8 là **khả năng inference nhanh**, phù hợp cho các ứng dụng thời gian thực như giám sát giao thông. Ngoài ra, mô hình được Ultralytics hỗ trợ mạnh mẽ về tài liệu, công cụ và cộng đồng, giúp người dùng dễ dàng tùy biến và huấn luyện lại trên dữ liệu riêng. Cơ chế anchor-free cũng góp phần giảm sai số do lựa chọn anchor không tối ưu, đơn giản hóa quá trình tinh chỉnh mô hình.

Tuy nhiên, YOLOv8 vẫn gặp **hạn chế** trong các bối cảnh có **đối tượng chồng lấp cao** hoặc **vật thể nhỏ**, nơi độ chính xác có thể giảm. Bên cạnh đó, hiệu suất của mô hình phụ thuộc đáng kể vào chất lượng và sự đa dạng của dữ liệu huấn luyện, đòi hỏi cần đầu tư kỹ trong khâu chuẩn bị dữ liệu để đạt được kết quả tốt nhất.

2.2.2 YOLO NAS (Neural Architecture Search) – CNN with NAS optimization

YOLO-NAS, được giới thiệu vào tháng 4 năm 2023, bởi **Deci.ai** – đơn vị chuyên tối ưu mô hình AI cho triển khai thực tế – phối hợp cùng **Ultralytics**, nhà phát triển dòng mô hình YOLO. Điểm nổi bật của YOLO-NAS nằm ở phương pháp thiết kế mô hình hiện đại, dựa trên **Neural Architecture Search (NAS)** – một kỹ thuật tự động hóa quá trình tìm kiếm kiến trúc mạng nơ-ron tối ưu thay vì thiết kế thủ công như truyền thống.



Hình 3. Sơ đồ quá trình tối ưu của YOLO-NAS

NAS hoạt động bằng cách **khám phá không gian kiến trúc mạng (search space)** và sử dụng **proxy training** để đánh giá nhanh hiệu suất của các kiến trúc ứng viên. Mục tiêu là tìm ra mô hình tốt nhất thỏa mãn đồng thời các tiêu chí như:

- Hiệu suất cao (mAP, accuracy)
- Tốc độ suy luận nhanh
- Độ trễ thấp trên thiết bị mục tiêu (GPU, CPU, mobile)
- Kích thước mô hình nhỏ.

Kết quả của quá trình tối ưu này là một mô hình với **backbone và head được tự động lựa chọn**, đạt sự cân bằng hiệu quả giữa tốc độ và độ chính xác, đồng thời phù hợp với từng loại thiết bị phần cứng cụ thể.

Ngoài ra, YOLO-NAS còn được thiết kế hướng đến **khả năng triển khai linh hoạt** nhờ tích hợp các kỹ thuật **lượng tử hóa (quantization)**. Đây là quá trình chuyển đổi các tham số mạng từ định dạng 32-bit (float32) sang các định dạng nhẹ hơn như int8 hoặc float16 nhằm:

- Giảm kích thước mô hình,
- Tiết kiệm bộ nhớ,
- Tăng tốc độ suy luận,
- Đáp ứng yêu cầu triển khai trên thiết bị tài nguyên hạn chế.

YOLO-NAS hỗ trợ cả **quantization-aware training (QAT)** và **post-training quantization (PTQ)**, đồng thời điều chỉnh cấu trúc mô hình để tương thích với lượng tử hóa, hạn chế sai số số học (numerical instability). Nhờ đó, mô hình hoạt động hiệu quả trên **thiết bị nhúng, edge device và mobile**, rất phù hợp trong các hệ thống giám sát giao thông thông minh – nơi hạ tầng phần cứng có thể bị giới hạn.

Theo báo cáo từ **Deci.ai**, YOLO-NAS cho thấy **hiệu suất vượt trội hơn YOLOv8**, với mức cải thiện mAP khoảng 1–2% trong các thử nghiệm trên cùng điều kiện. Một số phiên bản nhỏ như **YOLO-NAS-S** còn có thể đạt tốc độ suy luận nhanh hơn YOLOv8, phù hợp với các ứng dụng yêu cầu gần thời gian thực.

Tổng thể, YOLO-NAS mang lại nhiều **ưu điểm** đáng chú ý:

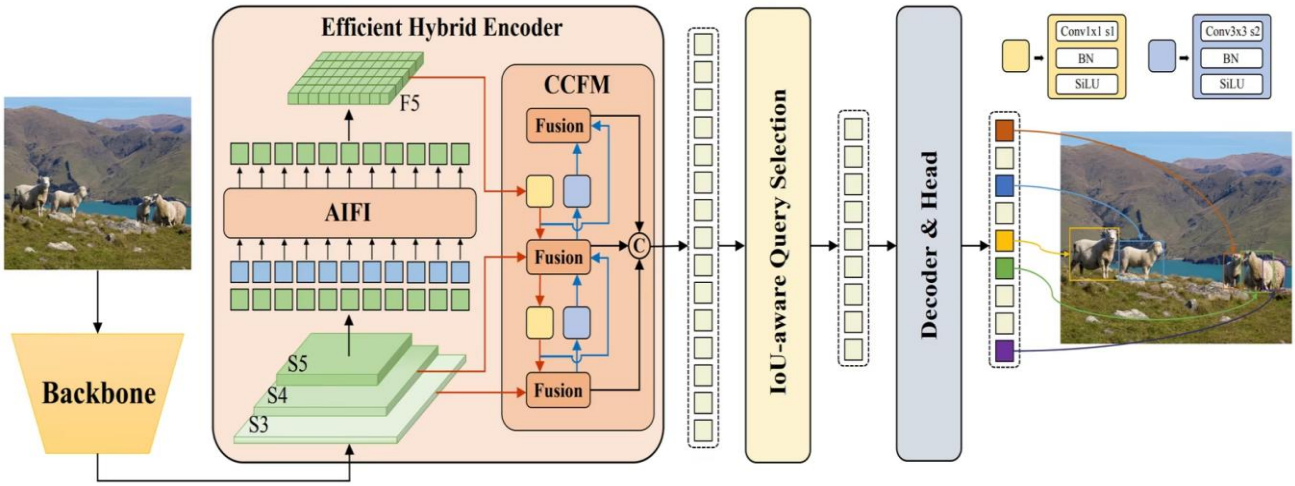
- Tối ưu tự động theo phần cứng mục tiêu.
- Hiệu suất cao và ổn định trên nhiều độ phân giải ảnh.
- Khả năng thích ứng tốt với tình huống giao thông phức tạp.

Tuy nhiên, mô hình cũng có một số hạn chế: **kích thước lớn** hơn các phiên bản YOLO truyền thống gây khó khăn khi triển khai trên thiết bị siêu nhỏ, và **quá trình huấn luyện phức tạp hơn**. Đồng thời, vì là phương pháp còn mới, **cộng đồng hỗ trợ hiện chưa phổ biến** như với YOLOv5/v8.

2.2.3 RT-DETR (Real-Time Detection Transformer)

RT-DETR là một trong những mô hình phát hiện đối tượng thế hệ mới dựa trên kiến trúc Transformer, được nhóm nghiên cứu của ByteDance giới thiệu vào tháng 6 năm 2023.

Mô hình này kế thừa nguyên lý hoạt động của DETR gốc với triết lý "end-to-end object detection without NMS", đồng thời cải tiến đáng kể về tốc độ suy luận và khả năng hội tụ để phù hợp hơn với các ứng dụng yêu cầu xử lý thời gian thực.



Hình 4. Kiến trúc RT-DETR

Khác với các dòng YOLO vốn dựa trên kiến trúc CNN truyền thống, **RT-DETR** sử dụng kiến trúc **encoder-decoder dựa trên Transformer**, cho phép mô hình học trực tiếp **các mối quan hệ hình học** giữa các đối tượng trong ảnh. Nhờ đó, mô hình có thể đưa ra dự đoán chính xác mà không cần đến bước hậu xử lý NMS (Non-Maximum Suppression), giúp đơn giản hóa pipeline và giảm độ trễ.

Kiến trúc tổng quát:

- **Backbone**: Một mạng CNN như ResNet hoặc ConvNeXt được sử dụng để trích xuất đặc trưng không gian từ ảnh đầu vào.

- **Encoder**: Áp dụng attention toàn cục để mã hóa mối quan hệ giữa các vùng trong ảnh, với sự hỗ trợ của khối **AIFI (Axial Interaction for Feature Interaction)** giúp giảm chi phí tính toán so với attention đầy đủ.

- **Decoder**: Nhận vào tập các **object queries** và dự đoán trực tiếp bounding box và nhãn lớp tương ứng cho từng đối tượng.

- **CCFM (Cross-Channel Fusion Module) và Fusion Module:** Tăng cường việc kết hợp đặc trưng giữa các tầng và chiều kênh, giúp cải thiện khả năng phát hiện trong các tình huống phức tạp như vật thể chồng lấp hoặc bố cục không rõ ràng**.**

RT-DETR được huấn luyện theo cách tiếp cận **end-to-end**. Trong quá trình training, mô hình sử dụng **Hungarian matching** để ánh xạ một-đối-một giữa các object queries và ground truth, đảm bảo mô hình không tạo ra các dự đoán dư thừa và tăng tính ổn định.

Nhằm khắc phục điểm yếu về tốc độ của DETR gốc, RT-DETR có các cải tiến tối ưu:

- Rút gọn số tầng encoder và decoder,

- Giảm độ phân giải đầu vào mà vẫn giữ độ chính xác,

- Tối ưu hóa attention với các cơ chế như multi-scale deformable attention.

Nhờ các cải tiến này, RT-DETR đạt được sự cân bằng giữa **độ chính xác cao**, **khả năng tổng quát hóa tốt** và **tốc độ inference ổn định**, đặc biệt phù hợp cho các bài toán thị giác thời gian thực mà không đánh đổi quá nhiều về hiệu suất.

Với cấu trúc Transformer, RT-DETR có khả năng học tốt các mối quan hệ hình học phức tạp – một lợi thế trong các cảnh có nhiều vật thể chồng lấp hoặc phân bố không đồng đều. Ngoài ra, việc loại bỏ hoàn toàn NMS giúp giảm lỗi trùng lặp, đơn giản hóa pipeline và giảm độ trễ tổng thể.

Tuy nhiên, mô hình vẫn có **một số hạn chế**:

- **Chi phí tính toán cao** hơn so với các mô hình CNN như YOLOv8, đặc biệt khi triển khai trên thiết bị không tối ưu cho self-attention;

- **Quá trình triển khai phức tạp**, đòi hỏi phần cứng mạnh và nhiều tài nguyên huấn luyện;

- Mặc dù tốc độ đã được cải thiện so với DETR gốc, nhưng trong các môi trường hạn chế tài nguyên như **edge device** hoặc **thiết bị di động**, RT-DETR vẫn **chưa thể vượt qua** các mô hình CNN được thiết kế chuyên biệt cho thời gian thực.

2.3. Object tracking

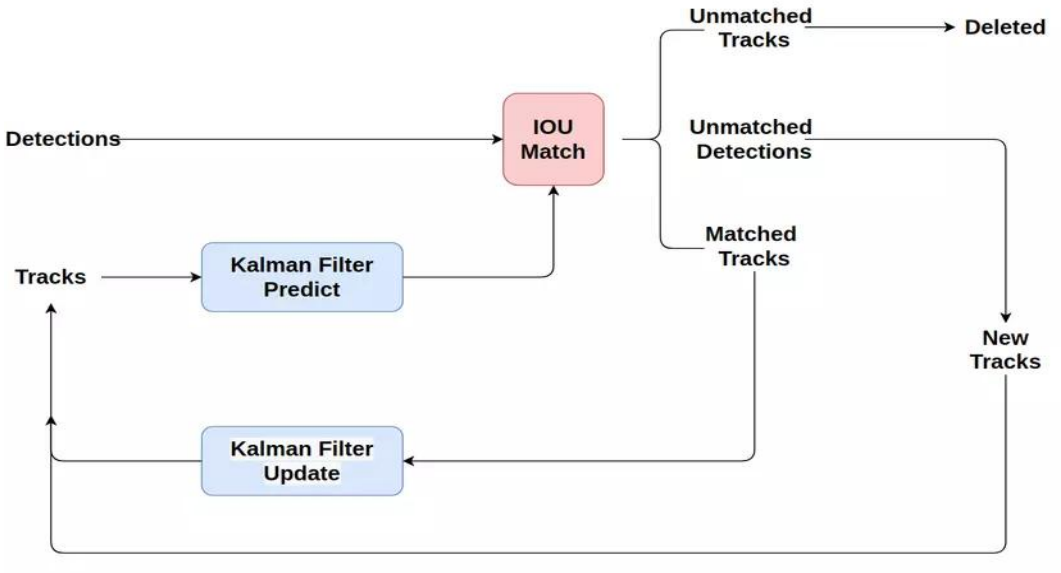
Object Tracking là bài toán theo dõi một hoặc nhiều đối tượng chuyển động theo thời gian trong một video. Hiểu một cách đơn giản nhất, nó là bài toán ở mức độ cao hơn so với object detection, khi đối tượng được xử lý không đơn giản là một hình ảnh mà là một chuỗi các hình ảnh (video). Object tracking gồm:

- ***Singer object tracking (SOT):*** Là những bài toán tập trung vào việc theo dõi một đối tượng duy nhất trong toàn bộ video. Và tất nhiên, để biết được cần theo dõi đối tượng nào, việc cung cấp một bounding box từ ban đầu là việc bắt buộc phải có. Các kỹ thuật tracking điển hình như BOOSTING, MIL, KCF, TLD, MEDIANFLOW, GOTURN, MOSSE, CSRT, ….

- ***Multiple object tracking (MOT):*** Là những bài toán khó hơn SOT, bài toán cố gắng phát hiện đồng thời theo dõi tất cả các đối tượng trong tầm nhìn, kể cả các đối tượng mới xuất hiện trong video.Và hướng tới các ứng dụng có tính mở rộng cao hơn. Các kỹ thuật tracking điển hình như SORT, DEEP SORT, BYTE TRACK, FAIRMOT, OC-SORT, BOT-SORT, ….

2.3.1. SORT

Giới thiệu chung: SORT là một phương pháp tracking đơn giản và hiệu quả, chủ yếu dựa vào việc phát hiện đối tượng qua các frame (ví dụ: bằng YOLO, RCNN), dự đoán vị trí của đối tượng trong các frame tiếp theo bằng Kalman Filter, và liên kết các detections với các track hiện có thông qua thuật toán Hungarian.



Hình 5. Luồng xử lý của SORT

Kalman Filter:

Để ứng dụng được Kalman Filter, việc xác định được các dạng biến cũng như mô hình ban đầu của quá trình là điều bắt buộc cần có. Với giả định các đối tượng chuyển động đều, và độc lập với các đối tượng khác, một track được xác định bằng

$$t = [x, y, s, r, v_x, v_y, v_s]$$

Trong đó:

- *t:* có ma trận hiệp phương sai ban đầu được khởi tạo với giá trị lớn để thể hiện sự không chắc chắn của trạng thái.
- *x, y:* lần lượt là tọa độ của tâm đối tượng (ở đây là tâm bounding box).
- *s:* là diện tích của bounding box.
- *r:* là tỷ lệ aspect ratio của bounding box.
- v_x, v_y, v_s: lần lượt là các giá trị vận tốc tương ứng của x, y, s.

Dựa trên thông tin của một track đã có ở frame trước để dự đoán vị trí bounding box ở vị trí tiếp theo, ta có phương trình dự đoán:

$$t_k = F_{k-1} t_{k-1} + u_{k-1}$$

tương đương với

$$\begin{bmatrix} x_k \\ y_k \\ s_k \\ r_k \\ \dot{x}_k \\ \dot{y}_k \\ \dot{s}_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ s_{k-1} \\ r_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \\ \dot{s}_{k-1} \end{bmatrix} + u_{k-1}$$

Để tính toán sự khác biệt giữa phép đo thực tế và phép đo dự đoán, ta có phương trình tính toán phần dư:

$$z_k = d_k - H_{k-1} t_k + w_k$$

tương đương với

$$\begin{bmatrix} x'_k \\ y'_k \\ s'_k \\ r'_k \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ s_k \\ r_k \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} x_k \\ y_k \\ s_k \\ r_k \\ \dot{x}_k \\ \dot{y}_k \\ \dot{s}_k \end{bmatrix} + w_k$$

Sau khi tính toán phần dư, ta tiến hành cập nhật lại các giá trị của một track, dùng để dự đoán vị trí của bounding box ở frame tiếp theo, ta có phương trình cập nhật:

$$t_{k+1} = t_k + K_k z_k$$

tương đương với

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ s_{k+1} \\ r_{k+1} \\ \dot{x}_{k+1} \\ \dot{y}_{k+1} \\ \dot{s}_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ s_k \\ r_k \\ \dot{x}_k \\ \dot{y}_k \\ \dot{s}_k \end{bmatrix} + \begin{bmatrix} k & 0 & 0 & 0 \\ 0 & k & 0 & 0 \\ 0 & 0 & k & 0 \\ 0 & 0 & 0 & k \\ k & 0 & 0 & 0 \\ 0 & k & 0 & 0 \\ 0 & 0 & k & 0 \end{bmatrix} * \begin{bmatrix} x'_k \\ y'_k \\ s'_k \\ r'_k \end{bmatrix}$$

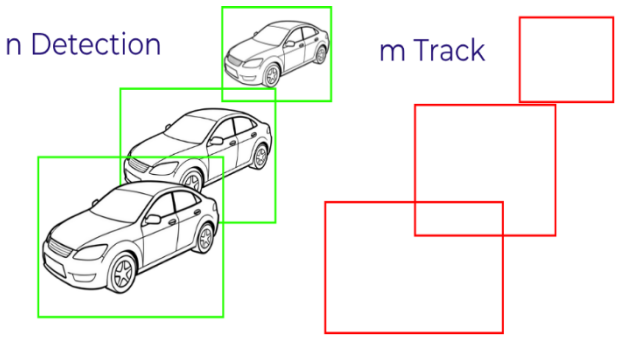
Trong đó:

- *t:* là thông tin của một track.
- *d:* là thông tin của một detection.
- *F: là* Ma trận chuyển trạng thái (State Transition Matrix).
- *H: là* Ma trận quan sát (Observation matrix).
- u_k, w_k: là các tham số nhiễu, được khởi tạo tuân theo phân phối chuẩn có mean = 0 và covariance không đổi.

Hungarian Algorithm:

Thuật toán Hungarian là một phương pháp tối ưu được đề xuất vào năm 1955 để giải bài toán phân công (assignment problem). Trong bài toán theo dõi đối tượng (object tracking), thuật toán này được sử dụng để ghép nối giữa các bounding box được phát hiện tại khung hình hiện tại với các track đang theo dõi sao cho tổng chi phí ghép nối là nhỏ nhất.

Cụ thể, mỗi bounding box (object) và mỗi track (dự đoán vị trí trước đó) được so khớp với nhau thông qua một ma trận chi phí, thường dựa trên độ chênh lệch vị trí hoặc độ tương đồng. Thuật toán đảm bảo mỗi object chỉ được gán cho một track và ngược lại, từ đó giúp hệ thống duy trì định danh chính xác của đối tượng qua nhiều khung hình.



Cơ chế hoạt động của SORT:

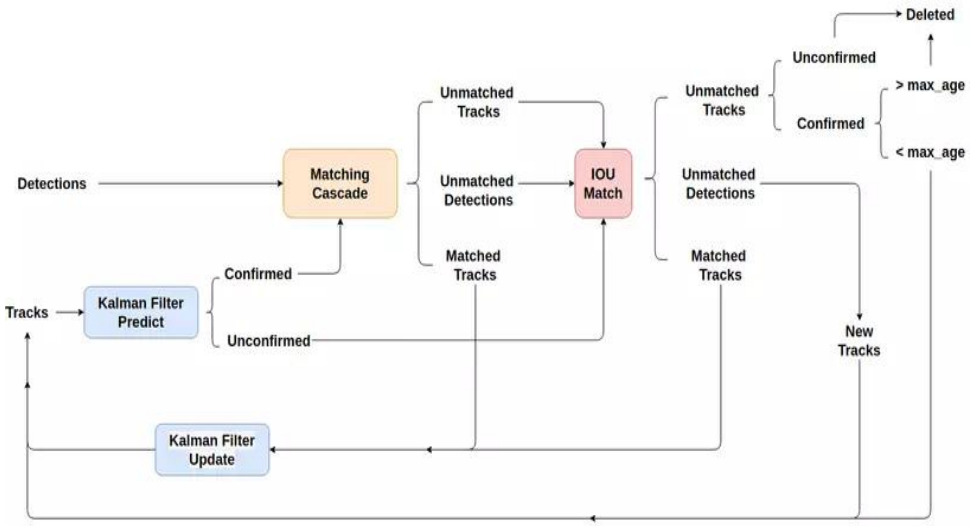
- Phát hiện đối tượng → sử dụng mạng neural như YOLO.
- Dự đoán vị trí mới dựa vào trạng thái của Kalman Filter.
- Liên kết detections với các track dựa trên tiêu chí khoảng cách (IoU).
- Cập nhật trạng thái của track sau khi liên kết thành công hoặc khi mất detections sẽ để vòng đời của track hết hạn sau một số frame không có detection.

Hạn chế: Không tích hợp các đặc trưng đặc trưng của đối tượng, chỉ dựa vào proximity của vị trí. Khi đối tượng chồng chéo hoặc bị che khuất, khả năng liên kết chính xác giảm (gây nhầm lẫn hoặc mất đối tượng).

Giải pháp nâng cao: Deep SORT ra đời để giải quyết hạn chế này bằng cách tích hợp đặc trưng hình ảnh của từng đối tượng, giúp phân biệt các đối tượng có vị trí gần nhau hơn, đồng thời xử lý tốt hơn các tình huống che khuất hay bị trùng lặp.

2.3.2. DeepSort

Giới thiệu chung: Deep SORT nâng cấp từ SORT bằng cách thêm một mạng neural học đặc trưng (deep feature extractor) để mã hoá đặc trưng của đối tượng, giúp liên kết chính xác hơn trong các tình huống phức tạp.

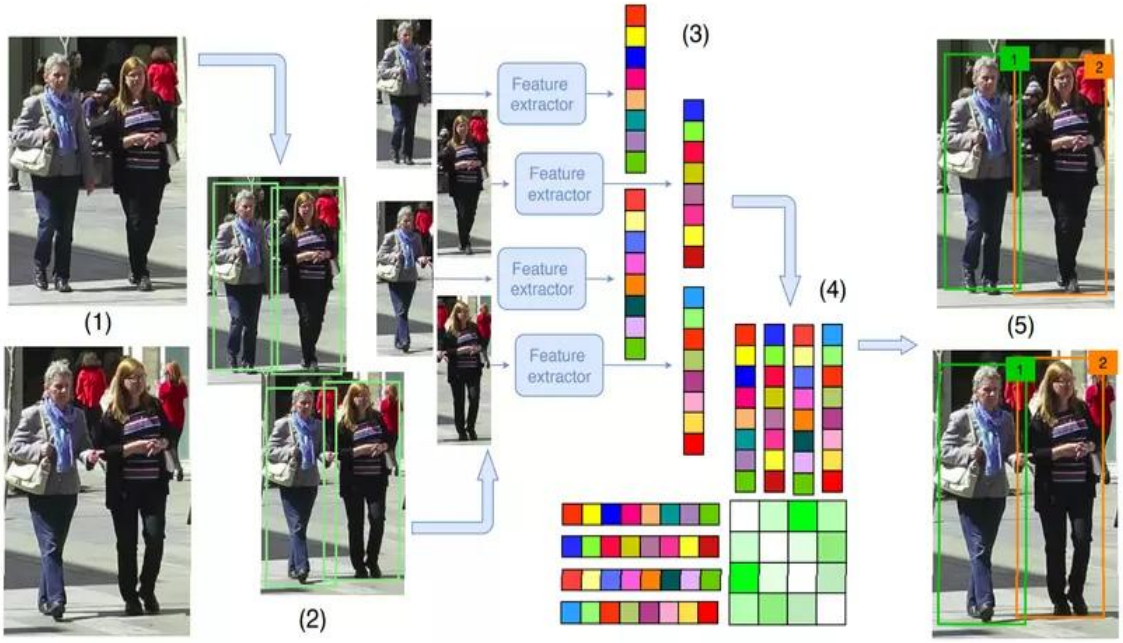


Hình 6. Luồng xử lý của Deep SORT

Re-Identification:

ReID là kỹ thuật học đặc trưng (feature embedding) của từng đối tượng, để giúp phân biệt các cá thể khác nhau, ngay cả khi chúng bị mất dấu rồi xuất hiện lại.

Trong Deep SORT, nhóm tác giả giải quyết vấn đề data association dựa trên thuật toán Hungary (tương tự như SORT), tuy nhiên, việc liên kết không chỉ dựa trên IOU mà còn quan tâm đến các yếu tố khác: khoảng cách của detection và track (xét tính tương quan trong không gian vector) và khoảng cách cosine giữa 2 vector đặc trưng được trích xuất từ detection và track - 2 vector đặc trưng của cùng 1 đối tượng sẽ giống nhau hơn là đặc trưng của 2 đối tượng khác nhau.



Hình 7. Kỹ thuật ReID

Các độ đo mới:

Mahalanobis Distance: Dùng để kiểm tra xem detection có nằm trong vùng dự đoán hợp lý không. Thường được so sánh với ngưỡng $t^{(1)} = 9.4877$.

$$d^{(1)}(i,j) = (d_j - t_i)^T S_i^{-1} (d_j - t_i)$$

Trong đó:

- d_j: là giá trị thực của detection thứ j.
- t_i: là giá trị dự đoán từ Kalman filter của track thứ i.
- S_i: là ma trận hiệp phương sai của biến ngẫu nhiên track thứ i.

Ngoài việc đo lường khoảng cách giữa track và detection, khoảng cách Mahalanobis còn được dùng để loại trừ các liên kết không chắc chắn bằng cách lập ngưỡng khoảng cách Mahalanobis ở khoảng tin cậy 95% được tính từ phân phối x^2.

$$b^{(1)}(i,j) = \left[d^{(1)}(i,j) \leq t^{(1)}\right]$$

Cosine Distance: Dùng để đánh giá độ giống nhau về appearance, nhằm đảm bảo việc liên kết chuẩn xác dù đối tượng đã biến mất và sau đó xuất hiện trở lại trong khung hình.

$$cosine\_distance(a,b) = 1 - \frac{a \cdot b}{\|a\|\|b\|}$$

Trong đó:

- a: là vector đặc trưng (embedding) của detection.
- b: là vector trung bình của track.

Với mỗi detection, đặc trưng r_j được trích xuất với $||r_j| = 1|$. Với mỗi track, một danh sách với độ dài khoảng 100 được sử dụng để lưu trữ đặc trưng của 100 track gần nhất: $R_k = \{r_k^{(i)}\}_{k=1}^{L_k}$, L_k=100.

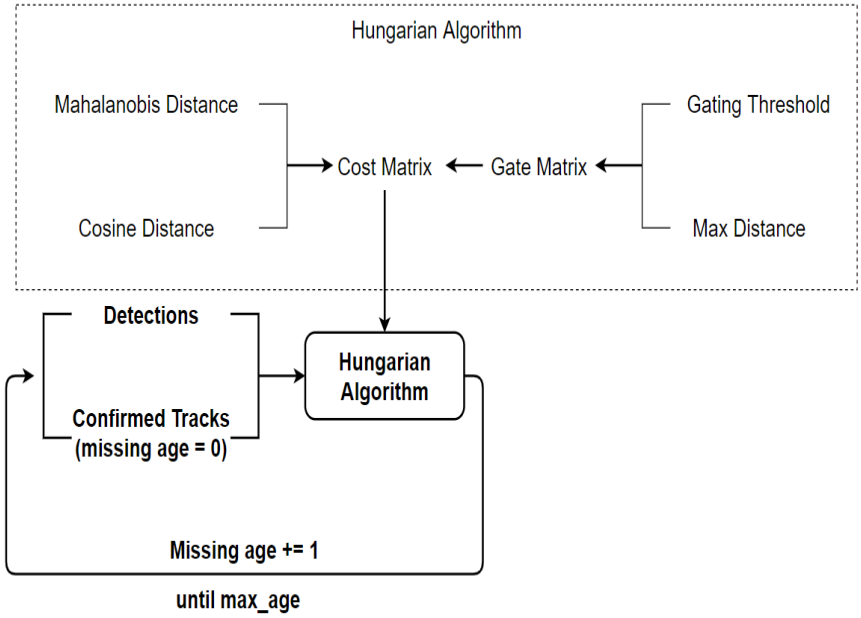
$$d^{(2)}(i,j) = min\left\{1 - r_j{}^T\, r_k^{(i)} \mid\ r_k^{(i)} \in R_i\right\}$$

Và $t^{(2)}$ được lựa chọn tùy vào tập dữ liệu sử dụng.

$$b^{(2)}(i,j) = \left[d^{(2)}(i,j) \leq t^{(2)}\right]$$

Khoảng cách Mahalanobis phản ánh thông tin vị trí dựa trên chuyển động tức thời, phù hợp cho dự đoán ngắn hạn. Trong khi đó, **khoảng cách cosine** đánh giá độ tương đồng đặc trưng của đối tượng, hiệu quả hơn trong các tình huống theo dõi dài hạn hoặc khi các đối tượng khó phân biệt.

Deep SORT kết hợp hai độ đo này thành một hàm đánh giá tổng hợp, với hệ số **λ** dùng để cân bằng mức độ ảnh hưởng giữa chúng. Trong thực nghiệm gốc, tác giả đặt **λ = 0**, tức chỉ sử dụng khoảng cách cosine, tuy nhiên **khoảng cách Mahalanobis vẫn hữu ích, do có thể lọc bỏ những detection có độ liên kết không đảm bảo ngưỡng.** Ma trận chi phí được tính từ khoảng cách Mahalanobis và khoảng cách cosine, sau đó được lọc qua ngưỡng để chọn ghép tốt nhất bằng thuật toán Hungarian. Các track không được cập nhật sẽ tăng tuổi và bị xóa nếu vượt quá max_age.



Hình 8. Quy trình kết hợp phát hiện và theo dõi đối tượng trong Deep SORT

Matching cascade:

Matching cascade là chiến lược ưu tiên ghép nối các track đã được xác nhận trong các frame gần nhất, giúp giảm sai số khi object bị mất dấu tạm thời. Deep SORT sử dụng cascade để tăng độ ổn định trong các tình huống che khuất.

Chiến lược đối sánh theo tầng tiến hành lấy lần lượt từng track ở các frame trước đó, để tiến hành xây dựng ma trận chi phí và giải bài toán phân công theo từng tầng. Chi tiết thuật toán được trình bày cụ thể hơn ở mã giả:

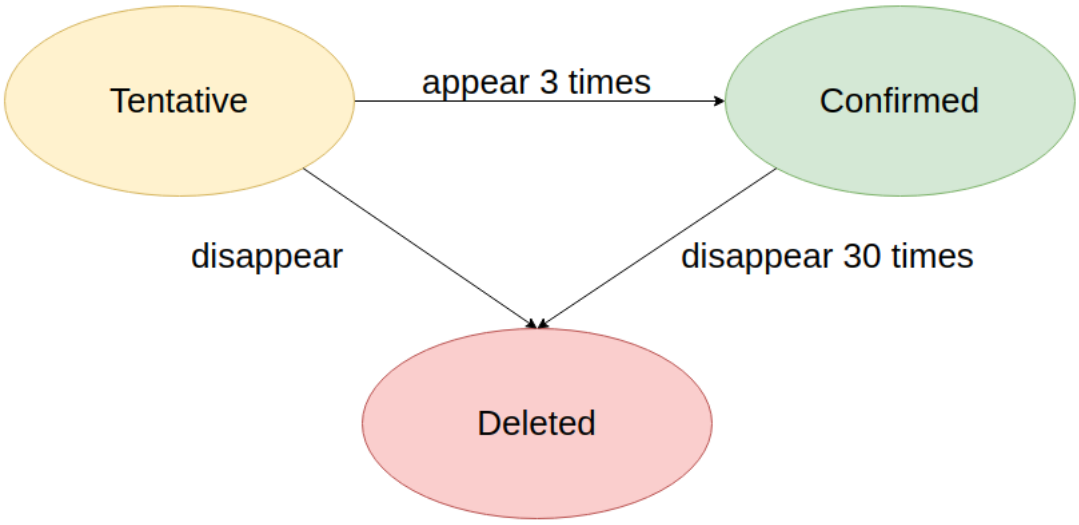
Listing 1 Matching Cascade

Input: Track indices $\mathcal{T} = \{1, \ldots, N\}$, Detection indices $\mathcal{D} = \{1, \ldots, M\}$, Maximum age $A_{\max}$

Compute cost matrix $C = [c_{i,j}]$
Compute gate matrix $B = [b_{i,j}]$
Initialize set of matches $\mathcal{M} \leftarrow \emptyset$
Initialize set of unmatched detections $\mathcal{U} \leftarrow \mathcal{D}$
for $n \in \{1, \ldots, A_{\max}\}$ do
  Select tracks by age $\mathcal{T}_n \leftarrow \{i \in \mathcal{T} \mid a_i = n\}$
  $[x_{i,j}] \leftarrow$ min cost matching $(C, \mathcal{T}_n, \mathcal{U})$
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{(i,j) \mid b_{i,j} \cdot x_{i,j} > 0\}$
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \{j \mid \sum_i b_{i,j} \cdot x_{i,j} > 0\}$
end for
return $\mathcal{M}, \mathcal{U}$

Hình 9. Thuật toán Matching Cascade.

Track lifecycle management:



Hình 10. Sơ đồ quản lý vòng đời của một track.

Deep SORT quản lý vòng đời của 1 track dựa trên 1 biến trạng thái với 3 giá trị (tentative, confirmed, deleted)

- Các trạng thái này lúc mới khởi tạo sẽ được gán 1 giá trị mang tính thăm dò (tentative).

- Giá trị này nếu vẫn đảm bảo duy trì được trong 3 frame tiếp theo, trạng thái sẽ chuyển từ thăm dò sang xác nhận (confirmed)

- Các track có trạng thái confirmed sẽ cố gắng được duy trì theo dõi, dù bị biến mất thì Deep SORT vẫn sẽ duy trì theo dõi trong 30 frame tiếp theo.

- Ngược lại, nếu mất dấu khi chưa đủ 3 frame, trạng thái sẽ bị xóa khỏi trình theo dõi (deleted).

Cơ chế hoạt động của Deep SORT:

- Sau khi phát hiện (detected bounding boxes), trích xuất đặc trưng hình ảnh sử dụng mạng deep learning.

- Sử dụng các đặc trưng này để tính khoảng cách giữa các đối tượng, từ đó tăng độ chính xác trong việc ghép nối các track và detection.

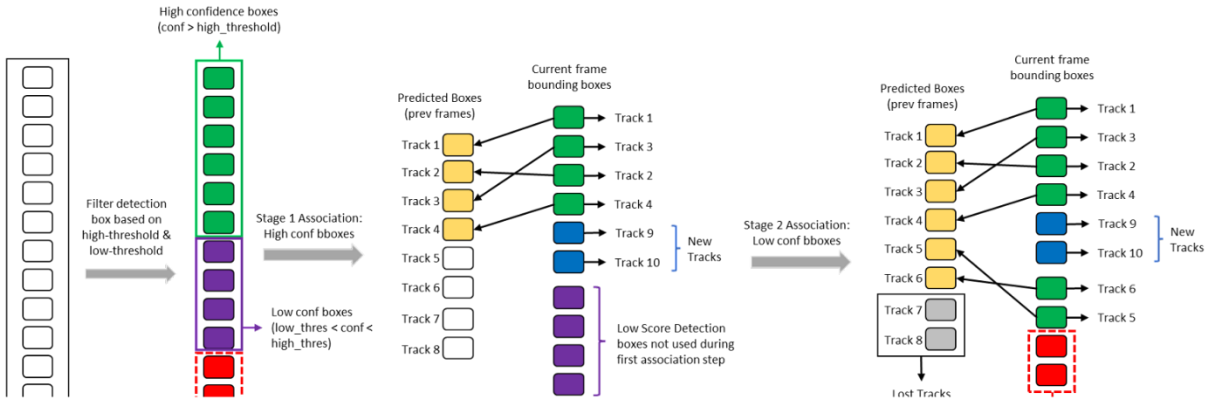
- Quản lý vòng đời của track dựa trên biến trạng thái, với các trạng thái như tentative, confirmed, deleted (như trong Deep SORT).

Hạn chế: Phụ thuộc nhiều vào khả năng trích xuất đặc trưng tốt, đòi hỏi mô hình mạnh và nhiều tài nguyên tính toán. Khả năng xử lý kém khi đối tượng có đặc trưng không rõ nét hoặc trong môi trường nhiều nhiễu, gây ảnh hưởng đến độ chính xác.

Giải pháp nâng cao: Các kỹ thuật dựa trên deep learning khác, như ByteTrack, đã cải thiện hiệu suất bằng cách giảm bớt phụ thuộc vào đặc trưng sâu này hoặc tối ưu mô hình để đáp ứng tốt hơn.

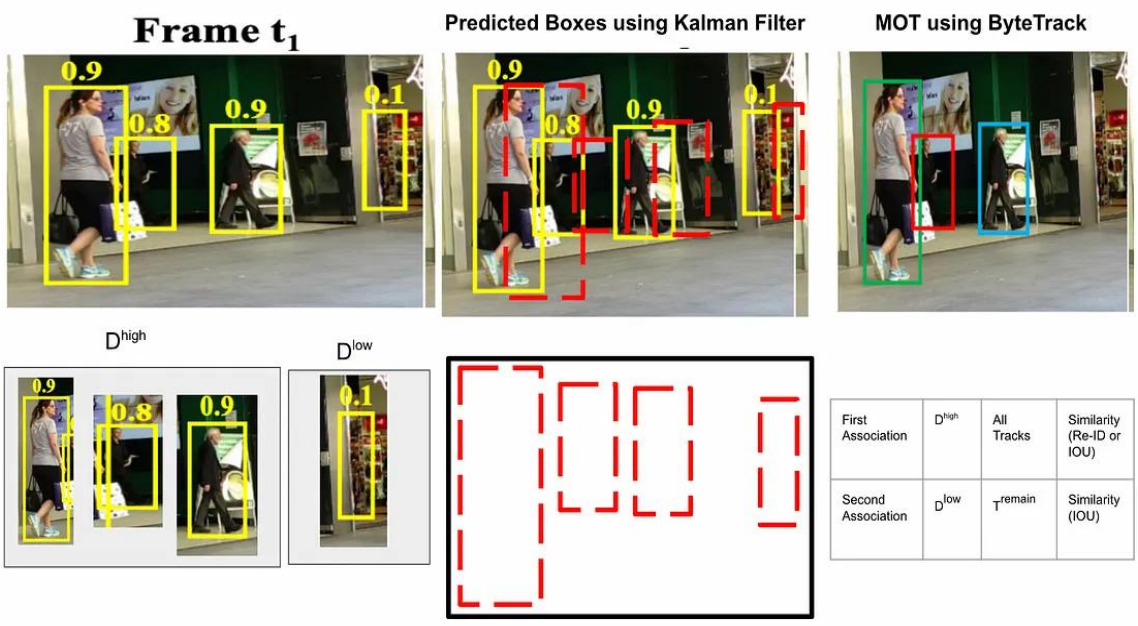
2.3.3. ByteTrack

Giới thiệu chung: ByteTrack là phương pháp tracking mới hơn, tối ưu cho môi trường nhiều đối tượng di chuyển nhanh và chen chúc, bằng cách tập trung vào hệ thống ghép nối detection và track một cách tối ưu hơn, hạn chế nhược điểm của các phương pháp trước.



Hình 11. Luồng xử lý của Byte Track.

Chiến lược theo dõi tất cả các đối tượng:



Hình 12. So sánh chiến lược liên kết tracklet giữa việc chỉ sử dụng các hộp detection có điểm số cao và việc sử dụng tất cả các hộp detection. Phương pháp sử dụng toàn bộ giúp theo dõi đối tượng chính xác hơn trong các khung hình khi confidence tạm thời bị giảm

Sau giai đoạn phát hiện, các hộp giới hạn được phát hiện được lọc thành các hộp có độ tin cậy cao, hộp độ tin cậy thấp và hộp nền với ngưỡng trên và dưới cố định. Các hộp nền sẽ bị loại bỏ sau quá trình này, nhưng cả hộp phát hiện có độ tin cậy thấp và cao đều được giữ lại cho các giai đoạn liên kết trong tương lai.

Tương tự như các bước liên kết điển hình từ các thuật toán khác, các hộp phát hiện điểm cao của các khung hình hiện tại được khớp với các hộp dự đoán từ các tracklet khung hình trước đó (sử dụng bộ lọc Kalman), bao gồm cả các tracklet đang hoạt động và các tracklet bị mất từ các khung hình gần đây. Việc khớp được thực hiện bằng cách sử dụng điểm IoU đơn giản hoặc điểm tương tự cosin giữa các nhúng tính năng (sử dụng các trình trích xuất tính năng như DeepSORT, QDTrack, v.v.) bằng thuật toán tiếng Hungary hoặc xếp tầng phù hợp trên Khoảng cách hàng xóm gần nhất. Việc gán tuyến tính giữa các cặp hộp giới hạn chỉ được xác nhận nếu điểm phù hợp cao hơn ngưỡng khớp cố định. Các hộp phát hiện điểm cao chưa từng có trong quá trình triển khai thực tế trước tiên được khớp với các tracklet với các bản cập nhật từ một khung hình trước khi được gán cho một tracklet mới.

Trong giai đoạn liên kết thứ hai, các hộp phát hiện điểm thấp được khớp với các hộp dự đoán không khớp còn lại từ các khung hình trước. Thuật toán khớp giống với giai đoạn liên kết đầu tiên, nhưng ngưỡng khớp được đặt thấp hơn giai đoạn liên kết đầu tiên vì trực giác rằng các hộp bị che khuất sẽ khớp kém hơn với các hộp từ các khung trước đó. Các hộp dự đoán không khớp được gán làm tracklet bị mất, trong khi các hộp phát hiện không khớp sẽ bị loại bỏ. Các tracklet bị mất được giữ trong một khoảng thời lượng khung hình nhất định và được thêm trở lại các tracklet đang hoạt động trước khi dự đoán bộ lọc Kalman. Điều này cho phép các trình theo dõi khôi phục một số tracklet đã bị mất do các vật thể biến mất hoàn toàn trong một số khung hình ngắn.

Điểm nổi bật:

- Sử dụng kỹ thuật ghép nối dựa trên các khả năng xử lý các detections có độ tin cậy thấp, giúp không bỏ sót các đối tượng nhỏ hoặc bị che khuất.
- Tối ưu trong việc xử lý số lượng lớn đối tượng, giảm hiện tượng mất track do các detections bị phân tán hoặc nhiễu.

Cơ chế hoạt động của Byte Track:

- Sử dụng bộ phát hiện như YOLO để phát hiện các bounding box trong từng frame.
- Sử dụng bộ lọc Kalman để dự đoán vị trí các đối tượng từ frame trước sang frame hiện tại.
- Chia detection thành 2 nhóm dựa trên confidence: cao và thấp.Ưu tiên ghép các detection confidence cao với tracks dự đoán bằng thuật toán Hungarian dựa trên IoU hoặc Mahalanobis distance.
- Các tracks được cập nhật theo detection ghép. Tracks không ghép được coi là mất tạm thời và đếm số frame không cập nhật.
- Loại bỏ tracks mất quá nhiều frame liên tiếp.

Hạn chế: Có thể gặp khó khăn trong các trường hợp có chuyển động quá nhanh gây mờ hoặc mất thông tin trong detection.Vẫn còn phụ thuộc vào độ chính xác của detection ban đầu và khả năng liên kết các detections không chính xác, nhưng đã giảm so với các phương pháp cũ.

Giải pháp: ByteTrack tối ưu quá trình ghép nối để nâng cao hiệu suất trong các môi trường phức tạp nhất, đồng thời giảm bớt khả năng sai lệch do đối tượng chen chúc hoặc mất detection.

2.4. Speed estimate

Vấn đề khi tính toán vận tốc bằng phương pháp thông thường

- Biến dạng phối cảnh (Perspective distortion): Trong hình ảnh từ camera, các đối tượng xa hoặc gần camera sẽ bị biến dạng khác nhau do góc nhìn và phối cảnh. Ví dụ, xe ở xa sẽ bị nhỏ hơn so với xe gần, dù chúng cùng tốc độ.

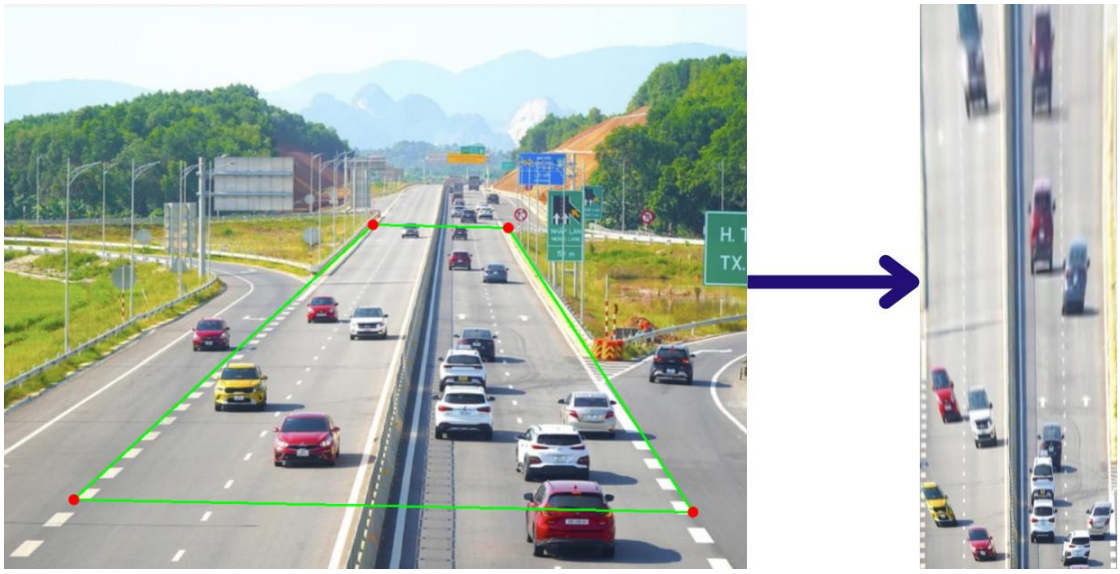
- Pixel không tuyến tính với khoảng cách thực: Khoảng cách pixels đại diện không tuyến tính với khoảng cách thực tế, dẫn đến sai số trong tính vận tốc nếu chỉ dựa trên chuyển động trong pixel.

- Hiệu ứng parallax: Khi camera hoặc vật thể di chuyển, các đối tượng ở các khoảng cách khác nhau sẽ có vận tốc apparent khác nhau, gây khó khăn trong ước lượng chính xác.

Phương pháp đề xuất: Perspective Transformation (biến đổi phối cảnh)

Nguyên lý hoạt động:

- Chọn 4 điểm đặc trưng trên hình ảnh đại diện cho mặt phẳng cố định, thường là mặt đất.
  - Gán tọa độ thực của 4 điểm này theo không gian thế giới (dưới đất).
  - Tính ma trận biến đổi hình sai lệch (homography H) dùng cv2.findHomography.
  - Dùng ma trận này để chuyển đổi tất cả các điểm ảnh sang hệ tọa độ trên mặt đất, tạo ra hình ảnh "bird's-eye view".



Hình 13 Mô phỏng chuyển đổi phối cảnh

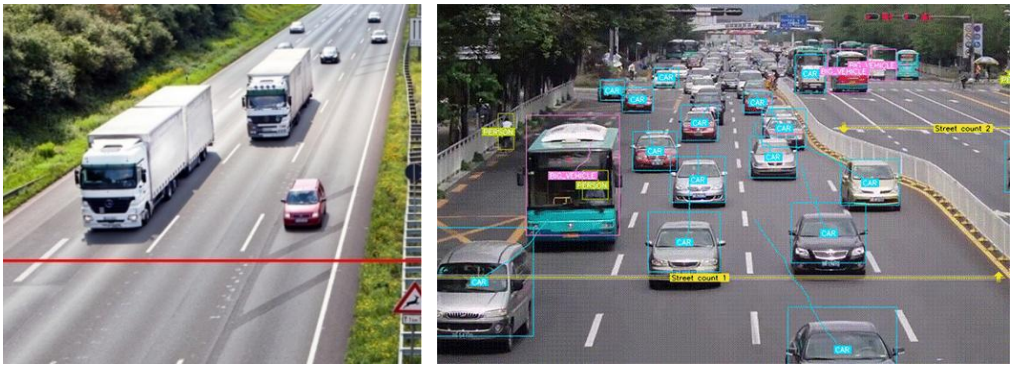
Công dụng: Giúp "dịch chuyển" hình ảnh thành bản đồ nhìn từ trên xuống, giảm ảnh hưởng của phối cảnh, làm cho các chuyển động của phương tiện trở nên tuyến tính và dễ dàng đo lường hơn.

Ưu điểm: Perspective transformation giảm thiểu ảnh hưởng của biến dạng phối cảnh, giúp ước lượng vận tốc chính xác hơn. Có thể chuyển đổi dữ liệu hình ảnh thành hệ thống tọa độ thực. Tăng tính ổn định trong theo dõi và đo vận tốc của phương tiện.

Hạn chế: Yêu cầu chọn chính xác 4 điểm đặc trưng, dễ bị sai lệch nếu điểm không ổn định hoặc bị mồi thuốc tốt / ánh sáng kém. Không phù hợp khi có nhiều vùng bị che khuất hoặc nền phức tạp. Thường cần tốn công chuẩn bị và hiệu chỉnh ban đầu.

2.5. Counting

Vạch đếm (line): Đây là phương pháp đếm bằng cách thiết lập một đường thẳng ảo trên ảnh hoặc video, thường là đường thẳng thẳng nằm ngang hoặc dọc. Phương tiện vượt qua đường này sẽ được ghi nhận đếm. Ưu điểm của đếm line là đơn giản, dễ triển khai, phù hợp để đếm số phương tiện theo hướng đi vào hoặc ra xác định.



Hình 14. Mô phỏng vạch đếm

Vùng đếm (zone): Là một khu vực hình chữ nhật hoặc đa giác trong ảnh, có diện tích xác định. Phương tiện xuất hiện hoặc rời khỏi vùng này sẽ được đếm. Phương pháp này linh hoạt hơn đếm line vì có thể xác định các vùng phức tạp, phù hợp để đếm số lượng xe tại các khu vực phức tạp như ở ngã tư, các khúc cua,...

Hình 15. Mô phỏng vùng đếm

Các trường hợp có thể gây đếm sai gồm:

- Đối tượng cùng lúc xuất hiện hoặc rời khỏi vùng đếm gây trùng lặp hoặc bỏ sót số lượng.
- Đối tượng vượt qua biên giới đếm nhiều lần do không có logic kiểm soát retries hoặc bỏ qua liên kết.

- Đối tượng có xuất hiện hoặc di chuyển sai hướng hoặc bị che khuất, gây nhầm lẫn trong việc xác định đúng thời điểm đếm

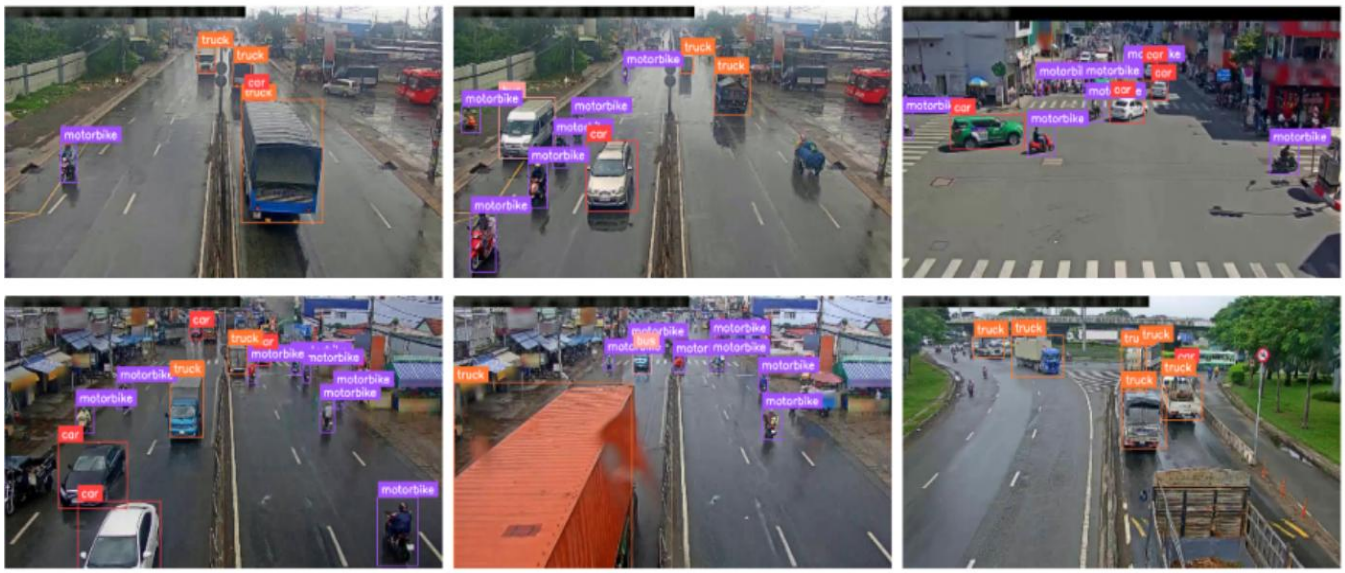
3. Dữ liệu (Data)

Do mục tiêu đồ án là đánh giá hiệu quả từng giai đoạn trong pipeline xử lý video giao thông, nhóm quyết định sử dụng hai tập dữ liệu riêng biệt tương ứng với hai nhiệm vụ chính: **phát hiện đối tượng (Object Detection)** và **theo dõi đối tượng (Object Tracking)**.

3.1. Dữ liệu cho Object Detection

Tập dữ liệu cho bài toán phát hiện đối tượng bao gồm:

- **Số lượng ảnh**: 9454 ảnh trong tập huấn luyện (train) và 2067 ảnh trong tập đánh giá (validation), với vai trò như tập kiểm thử (test).
- **Đặc điểm dữ liệu**: Ảnh được thu thập vào cả ban ngày và ban đêm, phản ánh được sự đa dạng về điều kiện ánh sáng.
- **Annotations**: Các ảnh được gán nhãn theo chuẩn định dạng của YOLO (dạng .txt), với tập nhãn bao gồm 4 loại phương tiện:
  - 0: Motorbike
  - 1: Car
  - 2: Bus
  - 3: Truck
- **Kích thước ảnh đầu vào**: Mỗi ảnh được resize về chuẩn đầu vào 640 × 640 pixel.
- **Nguồn dữ liệu**: Tập dữ liệu được thu thập và công khai trên Kaggle.



Hình 16. Một số ảnh minh họa cho tập dữ liệu

3.2 Dữ liệu cho Object Tracking

Do hạn chế về sự sẵn có của các bộ dữ liệu tracking chuyên biệt cho giao thông tại Việt Nam, nhóm quyết định tự tạo một bộ dữ liệu nhỏ bằng cách thủ công:

- **Tổng số video**: 5 video ngắn ghi lại cảnh phương tiện lưu thông thực tế trên đường.
- **Tổng số khung hình (frames)**: 3.252 frame sau khi trích xuất từ các video.
- **Gán nhãn thủ công**: Các bounding box và ID đối tượng được gán bằng tay thông qua công cụ CVAT (https://www.cvat.ai/) nhằm phục vụ cho việc kiểm tra và đánh giá mô hình theo dõi.
- **Mục tiêu sử dụng**: Tập dữ liệu này được dùng để so sánh hiệu quả giữa các thuật toán tracking sau khi kết hợp với các đầu ra từ mô hình phát hiện.

4. Độ đo đánh giá (Metrics)

Trong đồ án, hiệu quả của mô hình được đánh giá riêng biệt theo hai nhiệm vụ chính: **phát hiện đối tượng (Object Detection)** và **theo dõi đối tượng (Object Tracking)**. Mỗi nhiệm vụ sử dụng một tập chỉ số (metric) chuyên biệt nhằm phản ánh toàn diện độ chính xác, tính ổn định và khả năng ứng dụng thực tế của mô hình.

4.1. Object Detection Metrics

1. Mean Average Precision (mAP):

Là chỉ số trung bình của Average Precision (AP) trên tất cả các lớp đối tượng. mAP đo lường sự cân bằng giữa precision và recall trong suốt dải giá trị ngưỡng IoU (Intersection over Union).

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

- **mAP@0.5**: Tính precision tại ngưỡng IoU = 0.5.
- **mAP@0.5:0.95**: Trung bình mAP trên nhiều ngưỡng IoU từ 0.5 đến 0.95, là tiêu chuẩn đánh giá toàn diện hơn.

2. Precision, Recall và F1 score:

- **Precision** đo lường tỷ lệ dự đoán đúng trên tổng số dự đoán.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall** đo lường tỷ lệ dự đoán đúng trên tổng số thực thể đúng.

$$Recall = \frac{TP}{TP + FN}$$

- **F1-Score** là trung bình điều hòa giữa precision và recall, dùng để đánh giá tổng quan chất lượng phát hiện.

$$F_1\ score = 2 \times \frac{Precison \times Recall}{Precison + Recall}$$

3. Inference Time (ms):

Là thời gian suy luận (dự đoán) cho một ảnh đầu vào. Metric này phản ánh khả năng xử lý gần thời gian thực của mô hình, đặc biệt quan trọng trong các ứng dụng giao thông.

4.2. Object Tracking Metrics

1. MOTA – Multiple Object Tracking Accuracy:

Đo độ chính xác trong theo dõi đa đối tượng, phản ánh tổng thể mức độ sai lệch do các lỗi phổ biến như không phát hiện (False Negative), phát hiện sai (False Positive), và chuyển ID (ID Switch).

$$MOTA = 1 - \frac{FN + FP + IDSW}{GT}$$

- **IDSW (ID Switch)** là lỗi xảy ra khi hệ thống thay đổi ID đã gán cho một object trong quá trình theo dõi, mặc dù object đó vẫn đang hiện diện và di chuyển liên tục.

2. MOTP – Multiple Object Tracking Precision:

Đo độ chính xác về vị trí của các bounding box được gán cho các đối tượng được theo dõi, bằng cách tính trung bình IoU giữa các đối tượng thực và được dự đoán:

$$MOTP = \frac{\Sigma_{i,t} IoU_{i,t}}{\Sigma_t c_t}$$

Trong đó là c_t số lượng object tại thời điểm t.

3. IDF1 – ID F1 Score:

Đánh giá khả năng bảo toàn danh tính của đối tượng trong suốt quá trình theo dõi. Đây là trung bình điều hòa giữa ID Precision (IDTP / (IDTP + IDFP)) và ID Recall (IDTP / (IDTP + IDFN)):

$$IDF_{1=} \frac{2 \times IDTP}{2 \times IDTP + IDFP + IDFN}$$

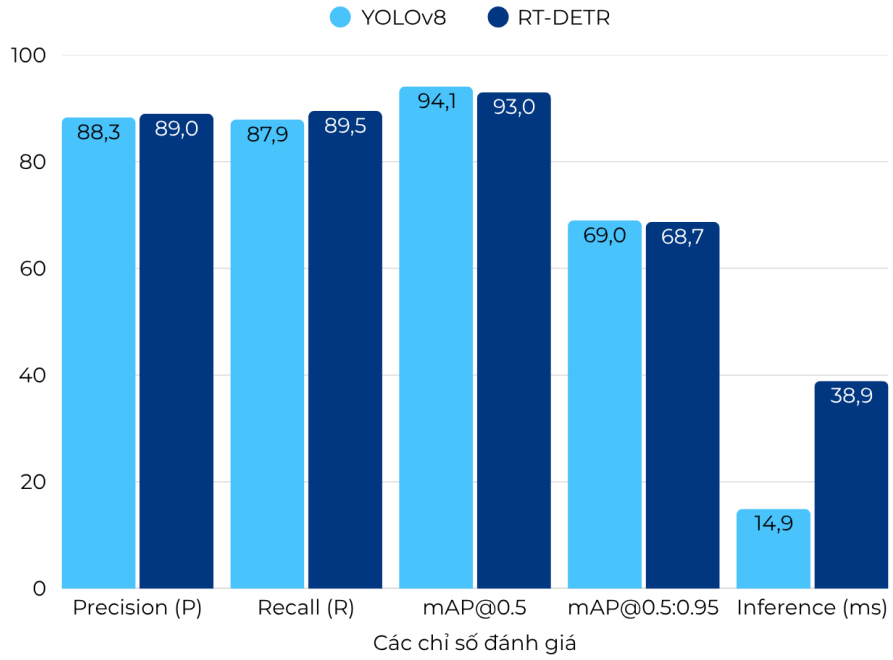
5. Thực nghiệm (Experiments)

5.1. Object detection

Để lựa chọn mô hình phù hợp cho bài toán nhận diện phương tiện giao thông, nhóm đã tiến hành huấn luyện và so sánh hai mô hình: **YOLOv8m** và **RT-DETRl**. Cả hai mô hình đều được huấn luyện lại (fine-tune) trên tập dữ liệu tự thu thập gồm bốn lớp: motorbike, car, bus, và truck. Tập dữ liệu bao gồm ảnh chụp ban ngày và ban đêm nhằm đảm bảo tính đa dạng.

Thông số huấn luyện chung:

- Số epoch: 5
- Image size: 640×640
- Optimizer: SGD (YOLOv8) và AdamW (RT-DETR)
- Batch size: 16
- Learning rate: mặc định theo từng mô hình



Hình 17. Đồ thị so sánh hiệu quả giữa YOLOv8 và RT-DETR

Nhận xét:

- RT-DETR cho kết quả tốt hơn về **độ chính xác tổng thể,** cho kết quả cao hơn YOLOv8 ở các chỉ số Precision và recall và mức mAP xấp xỉ cho thấy độ khả năng phát hiện vật thể chính xác và ổn định trên nhiều IoU threshold, nhất là trong trường hợp nhận diện vật thể nhỏ.

- **YOLOv8m** vượt trội về **tốc độ xử lý**, với thời gian suy luận chỉ ~11.6ms/ảnh, nhanh hơn **gấp 3 lần so với RT-DETR** (~38.9ms/ảnh). Điều này khiến YOLO trở thành lựa chọn lý tưởng cho các hệ thống yêu cầu **xử lý thời gian thực** như giám sát giao thông trực tiếp.

5.2. Xây dựng object tracking

Thiết lập tham số cho các kỹ thuật tracking:

SORT sử dụng các tham số đơn giản, phù hợp với yêu cầu thời gian thực:

- max_age = 1: số lượng frame tối đa mà track có thể mất dấu trước khi bị loại bỏ.
- min_hits = 3: số lần xuất hiện liên tiếp để một track được xác nhận chính thức.
- iou_threshold = 0.3: ngưỡng IOU tối thiểu để kết nối một detection mới với track hiện tại.

Deep SORT nâng cấp so với SORT bằng cách tích hợp đặc trưng ngoại hình (appearance feature), giúp theo dõi ổn định hơn trong môi trường phức tạp:

- max_dist = 0.1: ngưỡng khoảng cách cosine giữa vector đặc trưng để đánh giá độ giống giữa đối tượng mới và track hiện có.
- min_confidence = 0.3: loại bỏ các detection có độ tin cậy thấp.
- max_iou_distance = 0.7: kiểm soát việc ghép detection với track dựa trên IOU.

- max_age = 70: track được giữ lại tối đa 70 frame nếu không có detection mới.
- n_init = 3: track mới cần được xác nhận qua 3 lần liên tiếp để được xem là hợp lệ.
- nn_budget = 100: giới hạn số vector đặc trưng được lưu trữ để tránh quá tải bộ nhớ.

ByteTrack là một cải tiến hiện đại, cho phép sử dụng cả detection có confidence thấp nhằm tăng độ ổn định của track:

- track_thresh = 0.5: ngưỡng confidence để phân loại detection là đáng tin cậy.
- track_buffer = 30: số frame tối đa mà một track có thể bị mất mà vẫn được giữ lại.
- match_thresh = 0.85: ngưỡng IOU để thực hiện matching giữa detection và track.
- frame_rate = 30: tốc độ khung hình, ảnh hưởng đến chiến lược cập nhật track.
- fuse_score = False: không sử dụng confidence để tính điểm matching, giúp đơn giản hóa logic.

Kết quả :

Kết hợp mô hình	MOTA	MOTP	IDF1
YOLO + ByteTrack	**63.88**	23.67	**77.88**
RT-DETR + ByteTrack	52.78	**30.53**	76.23
YOLO + Deep SORT	58.38	27.30	53.75
RT-DETR + Deep SORT	57.48	26.88	76.38
YOLO + SORT	56.38	28.60	69.10
RT-DETR + SORT	53.75	29.98	71.63

Bảng 1. Kết quả kết hợp các mô hình

Nhận xét:

- **Hiệu suất mô hình:**

RT-DETR + ByteTrack đạt độ chính xác cao và ổn định, đặc biệt phù hợp cho các tác vụ cần phân biệt chi tiết giữa các loại phương tiện. Trong khi đó, YOLOv8 tuy có độ chính xác thấp hơn nhưng lại cho tốc độ xử lý cao, là lựa chọn phù hợp với ứng dụng thời gian thực trên thiết bị hạn chế tài nguyên.

- **Hiệu quả theo dõi:**

Kết hợp YOLO + ByteTrack cho kết quả IDF1 cao nhất, thể hiện khả năng duy trì định danh đối tượng hiệu quả qua nhiều khung hình. RT-DETR + ByteTrack đạt MOTP cao nhất, cho thấy khả năng định vị chính xác vị trí đối tượng, đặc biệt hữu ích trong môi trường nhiều vật thể chồng lấp.

- **Tác động từ điều kiện môi trường:**

Hệ thống hoạt động tốt dưới điều kiện ánh sáng ban ngày. Tuy nhiên, ban đêm hoặc khi video có độ phân giải thấp, mô hình dễ nhầm lẫn giữa các loại xe, đặc biệt là giữa car và truck khi đối tượng vừa xuất hiện từ xa.

- **Một số lỗi ghi nhận (tần suất thấp):**
  - ***Mất track*:** khi phương tiện bị khuất hoặc di chuyển nhanh ra khỏi khung hình.
  - ***Sai ID*:** xảy ra khi vật thể chồng lấp mạnh hoặc giao nhau.
  - ***Đếm sai*:** có thể xuất hiện nếu phương tiện quay đầu hoặc chạy ngược chiều.

5.3. Xây dựng hệ thống

Hệ thống được phát triển hoàn toàn bằng ngôn ngữ Python. Quá trình huấn luyện và đánh giá các mô hình nhận diện vật thể được thực hiện trên nền tảng Kaggle, sử dụng GPU để tăng hiệu quả xử lý. Giao diện người dùng được xây dựng bằng thư viện Streamlit, giúp triển khai thành ứng dụng web một cách nhanh chóng và trực quan.

- Các chức năng chính:

- Nhận diện và theo dõi phương tiện trong video
- Ước lượng tốc độ di chuyển của phương tiện
- Đếm số lượng xe theo từng loại
- Thống kê số lượng phương tiện theo hướng/ trong vùng chỉ định
- Xuất kết quả thành file excel để lưu trữ và xử lý nâng cao

Trong quá trình sử dụng, người dùng thực hiện thủ công việc khoanh vùng làn đường và cung cấp thông số chiều dài thực để hệ thống có thể tính toán vận tốc chính xác.

Mã nguồn và tài liệu liên quan có thể được truy cập tại:

https://github.com/HuynhNghiaKHMT/Vehicle_Tracking_Counting_Speeding

6. Kết luận (Conclusion)

Đồ án đã xây dựng một hệ thống phân tích phương tiện giao thông toàn diện dựa trên thị giác máy tính, với khả năng phát hiện, theo dõi, ước lượng vận tốc và đếm phương tiện hiệu quả trong nhiều điều kiện thực tế. Các mô hình hiện đại như **YOLOv8**, **YOLO-NAS** và **RT-DETR** được lựa chọn, trong đó YOLOv8 và RT-DETR được huấn luyện và thực nghiệm trực tiếp trên bộ dữ liệu thực tế, cho kết quả khả quan với độ chính xác cao và tốc độ suy luận phù hợp với triển khai thực tế theo thời gian thực.

Hệ thống đã cho thấy khả năng hoạt động hiệu quả với:

- Tốc độ suy luận cao (~30–50 FPS với GPU tầm trung).
- Độ chính xác phát hiện mAP@0.5 đạt từ 52–58% tùy mô hình.
- Tính ổn định trong theo dõi qua các chỉ số MOTA, MOTP, IDF1.
- Áp dụng thực tế vào việc đếm và tính vận tốc phương tiện với sai số nhỏ khi giả định đầu vào phù hợp.

Hạn chế

Tuy pipeline đã đáp ứng được yêu cầu kỹ thuật cơ bản, song hệ thống vẫn tồn tại một số **hạn chế**:

- **Phụ thuộc vào đầu vào thủ công**: Quá trình hiệu chỉnh tọa độ làn đường, điểm tham chiếu để tính tốc độ hiện vẫn được nhập bằng tay. Việc này khiến hệ thống thiếu đồng

bộ, phụ thuộc vào kiến thức và thao tác của người dùng cuối, đồng thời tiềm ẩn sai số do lỗi hiệu chỉnh.

- **Giảm hiệu quả trong điều kiện phức tạp**: Với những đoạn giao thông có mật độ cao, hiện tượng che khuất giữa các phương tiện xuất hiện thường xuyên, dẫn đến suy giảm độ chính xác cả trong phát hiện lẫn theo dõi. Các mô hình anchor-free hoặc transformer-based tuy có cải thiện nhất định, nhưng vẫn gặp khó khăn với cảnh chồng lấp mạnh hoặc các vật thể nhỏ nằm xa camera.

Hướng phát triển tương lai

Để nâng cao khả năng ứng dụng thực tế và tăng tính tự động hóa, nhóm đề xuất một số hướng phát triển như sau:

- **Tự động ước lượng biến đổi phối cảnh (Perspective Transformation)**: Thay vì người dùng phải tự hiệu chỉnh vùng đo tốc độ, hệ thống có thể áp dụng các kỹ thuật homography hoặc camera calibration để tự động hóa bước này. Điều này sẽ giúp đồng bộ hóa pipeline, nâng cao độ chính xác và dễ triển khai trên nhiều loại camera khác nhau.
- **Tích hợp phân tích hành vi phương tiện**: Hệ thống có thể mở rộng để phân loại và phát hiện các hành vi vi phạm giao thông như **vượt quá tốc độ**, **đi sai làn**, **dừng đỗ sai quy định**… bằng cách kết hợp thêm mô-đun phân loại hành vi dựa trên rule hoặc mô hình học sâu.
- **Cải thiện tracking trong môi trường đông đúc**: Kết hợp thêm mô hình tái nhận diện (re-identification) hoặc tracking học sâu (như OC-SORT) có thể giúp cải thiện tính ổn định khi có che khuất kéo dài. Ngoài ra, có thể tích hợp thêm dữ liệu từ radar hoặc camera từ nhiều góc độ (multi-camera) để bổ sung thông tin khi quan sát bị hạn chế.

7. Tài liệu tham khảo (References)

[1] A. H. Redmon et al., *A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS*, Journal of Artificial Intelligence Research, 2023.

[2] Ultralytics, *YOLOv8 Architecture Explained*. [Online]. Available: https://docs.ultralytics.com [Accessed: Jun. 2025].

[3] Deci AI, *A Comprehensive Guide to YOLO NAS: Object Detection with Neural Architecture Search*. [Online]. Available: https://www.deci.ai/yolo-nas/ [Accessed: Jun. 2025].

[4] X. Chu, Z. Yu, and B. Li, *RT-DETR: DETRs Beat YOLOs on Real-time Object Detection*, arXiv preprint arXiv:2304.08069, 2023.

[5] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, *Simple Online and Realtime Tracking*, in Proc. IEEE Int. Conf. Image Processing (ICIP), 2016, pp. 3464–3468.

[6] H. W. Kuhn, *The Hungarian Method for the Assignment Problem*, Naval Research Logistics Quarterly, vol. 2, no. 1–2, pp. 83–97, 1955.

[7] R. E. Kalman, *A New Approach to Linear Filtering and Prediction Problems*, Transactions of the ASME – Journal of Basic Engineering, vol. 82, no. 1, pp. 35–45, 1960.
[8] N. Wojke, A. Bewley, and D. Paulus, *Simple Online and Realtime Tracking with a Deep Association Metric*, in Proc. IEEE Int. Conf. Image Processing (ICIP), 2017, pp. 3645–3649.
[9] Y. Zhang et al., *ByteTrack: Multi-Object Tracking by Associating Every Detection Box*, in Proc. European Conf. Computer Vision (ECCV), 2022.