**VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY**  
**UNIVERSITY OF INFORMATION TECHNOLOGY**  
**FACULTY OF INFORMATION SYSTEMS**

Logo

Description automatically generated

**ĐỒ ÁN CUỐI KỲ**

**MÔN THỊ GIÁC MÁY TÍNH NÂNG CAO**

**PHÁT HIỆN LÀN ĐƯỜNG TRONG BÀI TOÁN XE TỰ LÁI**

**GIÁO VIÊN HƯỚNG DẪN**

**TS. MAI TIẾN DŨNG**

**SINH VIÊN**

**NGUYỄN QUỐC KHÁNH – 20521452**

**ĐINH PHƯƠNG NAM – 20520641**

**HỒ CHÍNH MINH, 05/2023**

MỤC LỤC

[I. Tổng quan về bài toán phát hiện làn đường 3](#_Toc135863545)

[Khái niệm bài toán 3](#_Toc135863546)

[Ứng dụng của bài toán 4](#_Toc135863547)

[II. Các phương pháp phát hiện làn đường 4](#_Toc135863548)

[1. Kỹ thuật dựa trên xử lý hình ảnh 5](#_Toc135863549)

[1.1. Giai đoạn hiệu chuẩn camera và điều chỉnh sự méo hình ảnh. 5](#_Toc135863550)

[1.2. Tạo một hình ảnh nhị phân với ngưỡng bằng cách sử dụng biến đổi màu sắc và độ dốc. 11](#_Toc135863551)

[1.3. Áp dụng biến đổi phối cảnh để chỉnh lại hình ảnh nhị phân ("birds-eye view"). 12](#_Toc135863552)

[1.4. Phát hiện các pixel của làn đường và đánh giá một biểu thức đa thức để tìm ranh giới của làn đường. 12](#_Toc135863553)

[1.5. Xác định độ cong của làn đường và vị trí của phương tiện so với trung tâm. 12](#_Toc135863554)

[1.6. Chồng lên các đường biên của làn đường đã phát hiện trở lại hình ảnh gốc. 12](#_Toc135863555)

[Kỹ thuật dựa trên deep learning 12](#_Toc135863556)

[1.7. Mô hình SCNN-Tensorflow 12](#_Toc135863557)

[1.1. Mô hình SCNN-Tensorflow 12](#_Toc135863558)

[III. Xây dựng giao diện người dùng (GUI) ứng dụng kết quả 12](#_Toc135863559)

[IV. Kết quả và đánh giá 12](#_Toc135863560)

[1. Bộ dữ liệu sử dụng 12](#_Toc135863561)

[2. Kết quả 13](#_Toc135863562)

[3. Đánh giá hiệu suất hệ thống 13](#_Toc135863563)

[4. Hướng phát triển 13](#_Toc135863564)

[V. Tài liệu tham khảo 13](#_Toc135863565)

# Tổng quan về bài toán phát hiện làn đường

## Khái niệm bài toán

**Phát hiện làn đường** là một nhiệm vụ thị giác máy tính liên quan đến việc xác định ranh giới của các làn đường lái xe trong video hoặc hình ảnh về cảnh đường. Mục tiêu là định vị và theo dõi chính xác vạch kẻ đường trong thời gian thực, ngay cả trong những điều kiện khó khăn như ánh sáng kém, ánh sáng chói hoặc bố cục đường phức tạp.



Phát hiện làn đường là một thành phần quan trọng của hệ thống hỗ trợ người lái tiên tiến (ADAS) và xe tự lái, vì nó cung cấp thông tin về bố cục đường và vị trí của xe trong làn đường, điều này rất quan trọng đối với việc điều hướng và đảm bảo an toàn. Các thuật toán thường sử dụng kết hợp các kỹ thuật thị giác máy tính, chẳng hạn như phát hiện cạnh, lọc màu và biến đổi Hough, để xác định và theo dõi vạch kẻ đường trong cảnh đường.

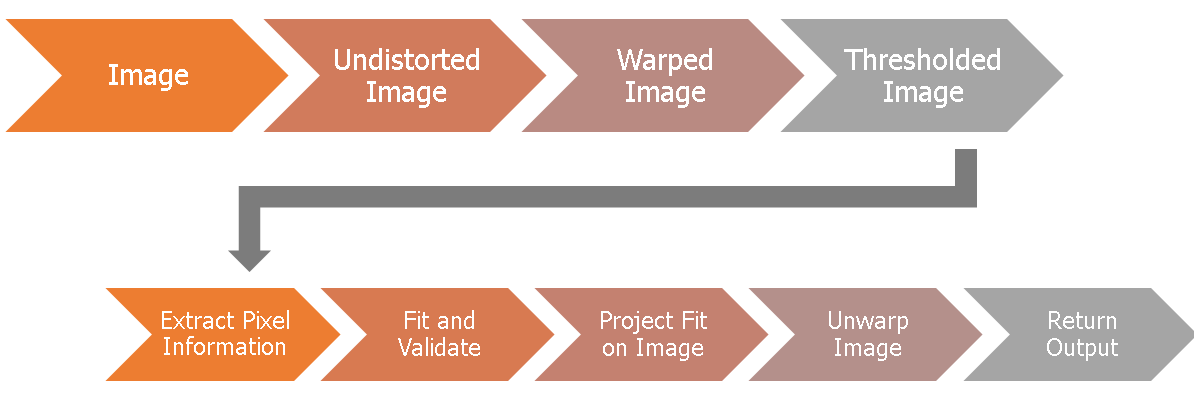
## Ứng dụng của bài toán

Bài toán phát hiện làn đường có nhiều ứng dụng trong lĩnh vực xe tự hành, hệ thống hỗ trợ lái và các ứng dụng thị giác máy tính khác. Một số ứng dụng phổ biến của bài toán phát hiện làn đường như:

* **Xe tự hành:** Trong hệ thống xe tự hành, phát hiện làn đường là một thành phần quan trọng để giúp xe xác định và duy trì vị trí trên đường. Thông qua việc nhận biết và theo dõi các làn đường, hệ thống xe tự hành có thể thực hiện di chuyển an toàn và tuân thủ luật lệ giao thông.
* **Hệ thống hỗ trợ lái**: Các công nghệ phát hiện làn đường được tích hợp trong hệ thống hỗ trợ lái, như hệ thống cảnh báo làn đường và hệ thống giữ làn đường. Các hệ thống này giúp người lái nhận biết và duy trì vị trí xe trên đường, giảm nguy cơ va chạm và cung cấp hướng dẫn lái an toàn.
* **Theo dõi đường:** Phát hiện làn đường cũng được sử dụng trong các ứng dụng theo dõi đường, như giám sát lưu lượng giao thông, phân tích dòng xe và dự đoán tình hình giao thông. Thông qua việc phát hiện và phân đoạn các làn đường, ta có thể thu thập thông tin về tình trạng đường, tạo ra bản đồ giao thông và cung cấp thông tin hữu ích cho quản lý giao thông.
* **Định vị và định hướng:** Phát hiện làn đường cũng có thể hỗ trợ trong việc định vị và định hướng cho các hệ thống định vị toàn cầu (GPS) và hệ thống định vị trong nhà (indoor positioning system). Thông qua việc nhận biết các làn đường và kết hợp với thông tin vị trí khác, ta có thể xác định vị trí chính xác và định hướng trong không gian.
* **An ninh giao thông:** Phát hiện làn đường cũng được áp dụng trong các hệ thống an ninh giao thông, như giám sát và phát hiện vi phạm luật giao thông. Bằng cách nhận dạng và theo dõi các làn đường, ta có thể phát hiện các hành vi vi phạm và giúp trong việc thực thi luật lệ giao thông.

# Các phương pháp phát hiện làn đường

## Kỹ thuật dựa trên xử lý hình ảnh



### Giai đoạn hiệu chuẩn camera và điều chỉnh sự méo hình ảnh.

#### Tại sao cần hiệu chỉnh máy ảnh

**Máy ảnh** là một thiết bị chuyển đổi thế giới 3D thành hình ảnh 2D. Một chiếc máy ảnh đóng vai trò rất quan trọng trong việc chụp ảnh ba chiều và lưu trữ chúng dưới dạng ảnh hai chiều.

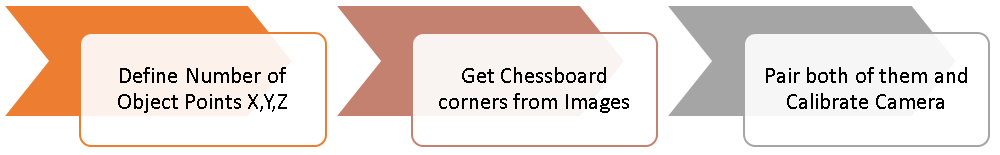
Khi ánh sáng đi qua kính máy ảnh hoặc ống kính máy ảnh. Ống kính không chỉ là những ống cắm mà ta gắn vào máy ảnh của mình; chúng thao túng ánh sáng theo những cách phức tạp. Một trong những tác động phụ của điều này là **biến dạng quang học.**

Ngày này, các máy ảnh thông thường, đặc biệt là các máy ảnh chuyên nghiệp hoặc cao cấp, thường được thiết kế và hiệu chỉnh để giảm thiểu biến dạng và tạo ra hình ảnh chính xác. Các hãng sản xuất máy ảnh thường tiến hành quá trình hiệu chỉnh và kiểm tra chất lượng để đảm bảo rằng hình ảnh thu được từ máy ảnh gần như không bị biến dạng đáng kể.

Nhưng đối với hầu hết các hệ thống máy ảnh đều có một sự biến dạng quang học nhất định, đặc biệt với máy ảnh lỗ kim, gây biến dạng đáng kể cho hình ảnh. Hai loại biến dạng chính là biến dạng xuyên tầm và biến dạng tiếp tuyến.

Bất cứ khi nào ta sử dụng camera trong bất kỳ hệ thống nào, chẳng hạn như trong hệ thống lái tự động trên ô tô, ta cần đảm bảo rằng camera được lắp trong ô tô đó là chính xác. Chính xác ở đây là máy ảnh phải có độ chính xác, độ phân giải cao và độ méo thấp hoặc không có. Độ chính xác cao có nghĩa là máy ảnh sẽ cung cấp cùng một đầu ra cho cùng một đầu vào trong mọi loại điều kiện môi trường. Độ phân giải cao có nghĩa là máy ảnh sẽ có thể giải mã những thay đổi nhỏ nhất có thể. Vì vậy, máy ảnh nên được hiệu chỉnh để đạt được độ chính xác cao hơn và độ méo thấp giúp đạt được sự thể hiện thế giới thực chính xác hơn trong các hình ảnh được chụp.

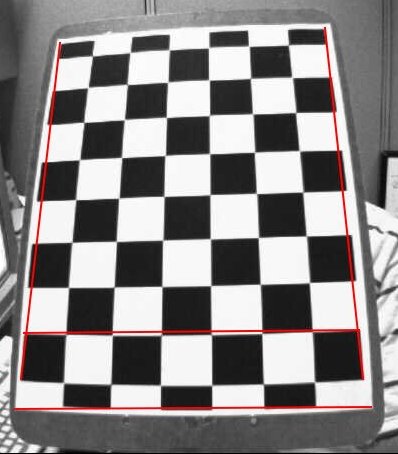
**Camera Calibration (hiệu chỉnh máy ảnh)** là quá trình để xác định các thông số nội tại và ngoại tại của một hệ thống máy ảnh. Quá trình này có mục đích tạo ra một mô hình toán học để chuyển đổi từ điểm ảnh trên mặt phẳng ảnh sang không gian thực tế.



#### Các tham số cần hiệu chỉnh

Như đã trình bày, có hai loại biến dạng chính là biến dạng xuyên tâm và biến dạng tiếp tuyến.

* Biến dạng xuyên tâm làm cho các đường thẳng xuất hiện cong. Biến dạng xuyên tâm trở nên lớn hơn khi các điểm xa hơn từ trung tâm của hình ảnh. Ví dụ: một hình ảnh được hiển thị bên dưới trong đó hai cạnh của bàn cờ được đánh dấu bằng các đường màu đỏ. Nhưng, bạn có thể thấy rằng đường viền của bàn cờ không phải là một đường thẳng và không khớp với đường màu đỏ. Tất cả các đường thẳng dự kiến ​​​​được phình ra.



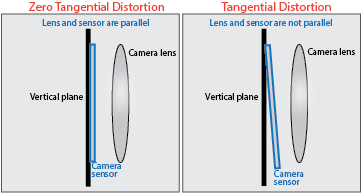
*Ảnh bàn cờ bị biến dạng quan học. Nguồn: OpenCV*

Biến dạng xuyên tâm có thể biểu diễn như sau

*Công thức do Fitzgibbon đề xuất. Nguồn: OpenCV*

Trong đó:

* + - và lần lượt là tọa độ tương ứng của điểm không bị biến dạng và bị biến dạng trong ảnh
    - là hệ số biến dạng xuyên tâm
    - là khoảng cách Eculidean từ điểm biến dạng tới điểm trung tâm ảnh bị biến dạng
* Tương tự, biến dạng tiếp tuyến xảy ra do thấu kính chụp ảnh không được căn chỉnh hoàn toàn song song với mặt phẳng tạo ảnh. Vì vậy, một số khu vực trong hình ảnh có thể trông gần hơn dự kiến



*Ảnh minh họa biến dạng tiếp tuyến. Nguồn: mathworks*

Biến dạng tiếp tuyến có thể biểu diễn như sau

Trong đó:

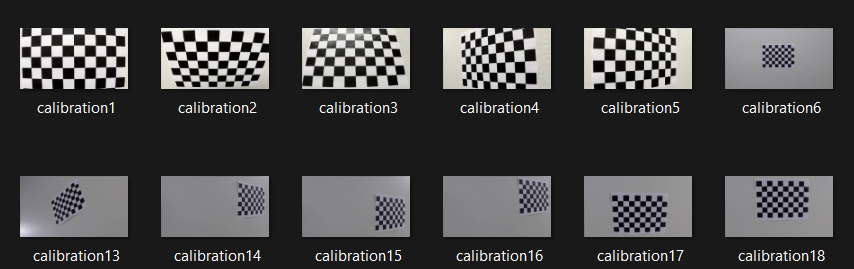
* + - và lần lượt là tọa độ tương ứng của điểm không bị biến dạng và bị biến dạng trong ảnh
    - là hệ số biến dạng tiếp tuyến
    - là khoảng cách Eculidean từ điểm biến dạng tới điểm trung tâm ảnh bị biến dạng

Tóm lại, ta cần tìm năm tham số, được gọi là hệ số biến dạng (distortion coefficients):

Ngoài ra, chúng ta cần một số thông tin khác là thông số bên trong và bên ngoài của máy ảnh. Các thông số bên trong dành riêng cho một máy ảnh. Chúng bao gồm thông tin như độ dài tiêu cự và tâm quang học . Độ dài tiêu cự và tâm quang học có thể được sử dụng để tạo ma trận máy ảnh, ma trận này có thể được sử dụng để loại bỏ biến dạng do ống kính của một máy ảnh cụ thể. Ma trận máy ảnh là duy nhất cho một máy ảnh cụ thể, do đó, sau khi được tính toán, nó có thể được sử dụng lại trên các hình ảnh khác được chụp bởi cùng một máy ảnh. Nó được thể hiện dưới dạng ma trận 3x3:

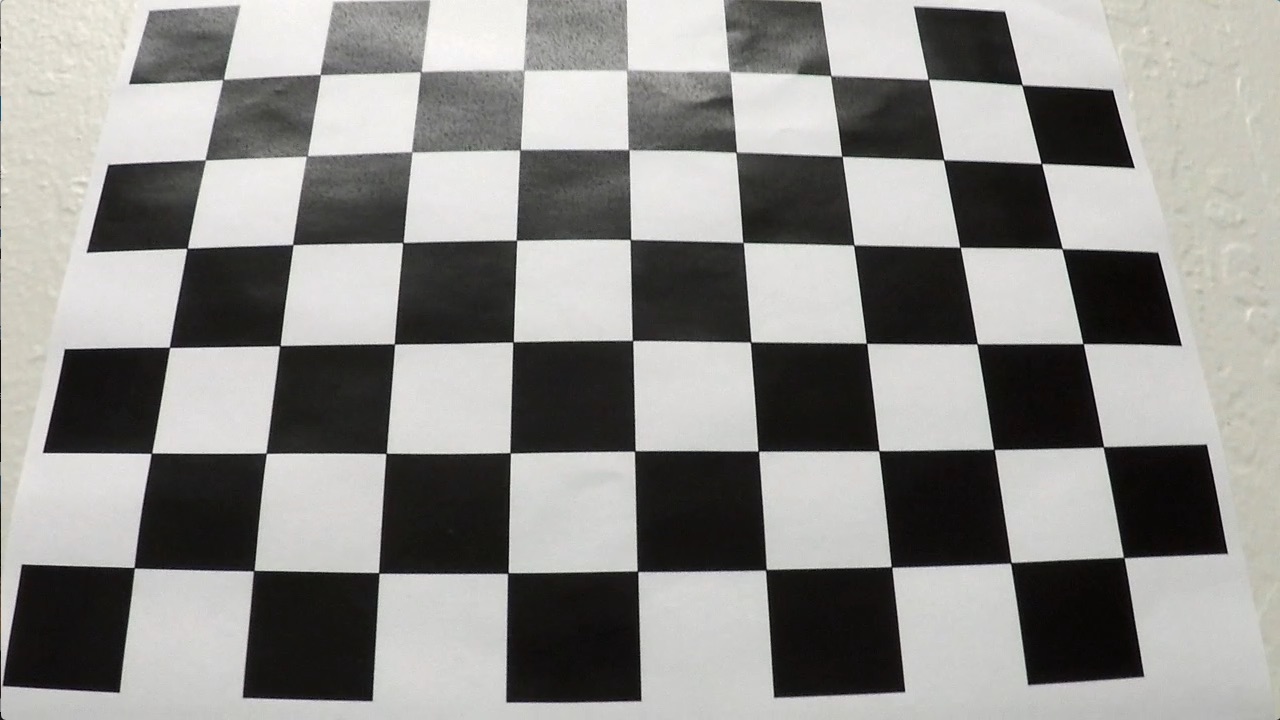
Và các tham số bên ngoài của máy ảnh tương ứng với các vectơ xoay và tịnh tiến giúp dịch tọa độ của điểm 3D sang hệ tọa độ.

Để tìm các tham số này, chúng ta cần cung cấp một số hình ảnh mẫu của một mẫu được xác định rõ ràng (ví dụ: bàn cờ). Chúng ta tìm thấy một số điểm cụ thể mà chúng ta đã biết vị trí tương đối của chúng (ví dụ: góc của ô vuông trên bàn cờ). Chúng ta biết tọa độ của những điểm này trong không gian thực và chúng ta biết tọa độ trên hình ảnh, vì vậy chúng ta có thể giải quyết cho các hệ số biến dạng. Để có kết quả tốt hơn, chúng ta cần ít nhất 10 mẫu thử nghiệm.



*Hình ảnh bàn cờ đã được chuẩn bị được chụp từ nhiều góc độ khác nhau*

Ta sẽ xem xét 1 hình ảnh.

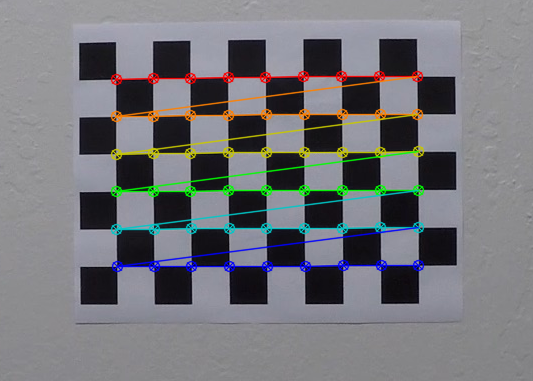


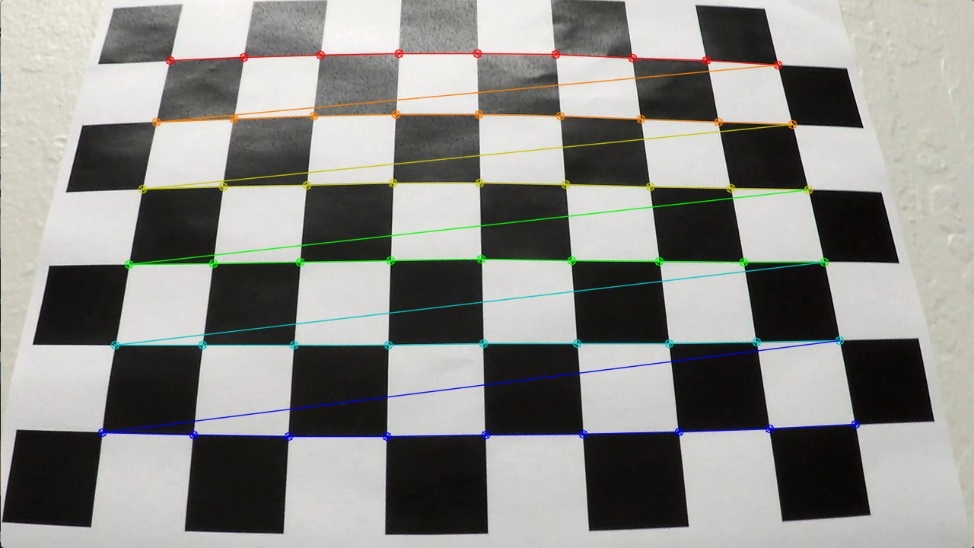
Dữ liệu đầu vào quan trọng cần thiết để hiệu chỉnh máy ảnh là tập hợp các điểm trong thế giới thực 3D và tọa độ 2D tương ứng của các điểm này trong ảnh.

Các điểm hình ảnh 2D chúng ta có thể dễ dàng tìm thấy từ hình ảnh. (Các điểm hình này là vị trí hai ô đen chạm nhau trên bàn cờ)

Còn các điểm 3D từ không gian trong thế giới thực. Những hình ảnh đó được chụp từ một camera tĩnh và các bàn cờ được đặt ở các vị trí và hướng khác nhau. Vì vậy chúng ta cần biết các giá trị . Nhưng để đơn giản, chúng ta có thể nói bàn cờ được giữ cố định trên mặt phẳng , (vì vậy Z luôn bằng -) và máy ảnh được di chuyển tương ứng. Việc xem xét này giúp chúng ta chỉ tìm các giá trị X, Y. Bây giờ đối với các giá trị X,Y, chúng ta chỉ cần chuyển các điểm như (0,0), (1,0), (2,0), ... biểu thị vị trí của các điểm. Trong trường hợp này, kết quả chúng ta nhận được sẽ là tỷ lệ kích thước của ô vuông bàn cờ. Nhưng nếu chúng ta biết kích thước hình vuông, (giả sử là 30 mm), chúng ta có thể chuyển các giá trị như (0,0), (30,0), (60,0), ... . Do đó, chúng tôi nhận được kết quả tính bằng mm. (Trong trường hợp này, chúng tôi không biết kích thước hình vuông vì chúng tôi không chụp những hình ảnh đó, vì vậy chúng tôi bỏ qua kích thước hình vuông). Điểm 3D được gọi là **điểm đối tượng** và điểm ảnh 2D được gọi là **điểm ảnh.**

Vì vậy, để tìm mẫu cẫu trúc trong bàn cờ, chúng ta có thể sử dụng hàm [**cv.findChessboardCorners()**](https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a) . Chúng ta cũng cần chuyển loại mẫu mà chúng ta đang tìm kiếm, chẳng hạn như lưới 8x8, lưới 5x5, v.v. Trong ví dụ này, chúng tôi sử dụng lưới 10x7. (Thông thường bàn cờ vua có ô vuông 8x8 và góc trong 10x9). Nó trả về các điểm góc và giá trị hồi quy sẽ là True nếu lấy được mẫu. Các góc này sẽ được đặt theo thứ tự (từ trái sang phải, từ trên xuống dưới).





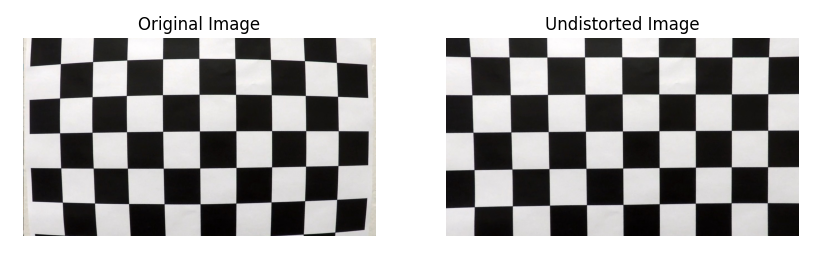
*Hình ảnh với vị trí các góc trong bàn cờ được vẽ*

Bây giờ chúng ta đã có các điểm đối tượng và điểm hình ảnh, chúng ta đã sẵn sàng để hiệu chuẩn. Chúng ta có thể sử dụng hàm, [**cv. CalibCamera()**](https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga3207604e4b1a1758aa66acb6ed5aa65d) để trả về ma trận máy ảnh, hệ số biến dạng, vectơ xoay và dịch, v.v.



#### Undistortion – Phục hồi ảnh biến dạng

Bây giờ, chúng ta có thể chụp ảnh và phục hồi nó bằng cách sử dụng [**cv.undistort()**](https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga69f2545a8b62a6b0fc2ee060dc30559d). Đây là cách dễ dàng nhất. Chỉ cần gọi hàm và sử dụng ROI thu được ở trên để cắt kết quả.



*Hình ảnh sau khi hiệu chỉnh biến dạng*

### Áp dụng biến đổi phối cảnh để chỉnh lại hình ảnh nhị phân ("birds-eye view").

Trong **Chuyển đổi phối cảnh,** chúng ta có thể thay đổi phối cảnh của một hình ảnh hoặc video nhất định để hiểu rõ hơn về thông tin được yêu cầu. Trong Perspective Transformation, chúng ta cần cung cấp các điểm trên ảnh mà từ đó muốn thu thập thông tin bằng cách thay đổi phối cảnh. Chúng tôi cũng cần cung cấp các điểm bên trong mà chúng tôi muốn hiển thị hình ảnh của mình. Sau đó, chúng ta lấy biến đổi phối cảnh từ hai tập hợp điểm đã cho và bọc nó bằng hình ảnh ban đầu.

Chúng tôi sử dụng **cv2.getPerspectiveTransform** và sau đó là **cv2.warpPerspective.**

#### PerspectiveTransform trong bài toán phát hiện làn đường

Trong bài toán lane line detection, Perspective Transform được sử dụng để chuyển đổi góc nhìn của hình ảnh từ góc nhìn camera sang góc nhìn từ trên cao (bird's-eye view) của đường đi.

Vì camera được đặt ở một vị trí cố định trên xe, hình ảnh thu được từ camera thường có góc nhìn nghiêng và biến dạng. Điều này làm khó khăn trong việc phân tích và nhận dạng các đường line trên mặt đường. Bằng cách sử dụng Perspective Transform, chúng ta có thể chuyển đổi hình ảnh từ góc nhìn camera sang góc nhìn từ trên cao, tạo ra một hình ảnh mới mà các đường line sẽ trở nên song song và không bị biến dạng.

Quá trình Perspective Transform thường được thực hiện bằng cách xác định các điểm tham chiếu trên mặt phẳng camera và xác định các điểm mục tiêu tương ứng trên mặt phẳng bird's-eye view. Sau đó, chúng ta áp dụng một phép biến đổi hình học để ánh xạ các điểm ảnh từ góc nhìn camera sang góc nhìn bird's-eye view.

Kết quả là chúng ta thu được một hình ảnh mới với góc nhìn từ trên cao, trong đó các đường line trở nên song song và dễ dàng để xác định và phân tích. Điều này rất hữu ích trong việc xử lý lane line detection, giúp chúng ta xác định vị trí và hướng di chuyển của xe trên đường.

Chúng ta sẽ cung cấp bảng Transform Points mà chúng tôi muốn tập trung vào làn đường chính của xe đang di chuyển.

|  |  |
| --- | --- |
| **Source** | **Destination** |
| 550, 460 | 100,0 |
| 150, 720 | 100, 720 |
| 1200, 720 | 1100, 720 |
| 770, 460 | 1100, 0 |

*Transform points table*

Trong Perspective Transform, chỉ cần có 4 điểm trong hệ tọa độ nguồn và hệ tọa độ đích để xác định một phép biến đổi perspective tương ứng theo thứ tự từ trên xuống là các điểm **top-left, bottom-left, bottom-right, top-right**.

#### Visualization of Transform

Kết quả một số hình ảnh trong khi biến đổi phối cảnh



### Tạo một hình ảnh nhị phân với ngưỡng bằng cách sử dụng biến đổi màu sắc và độ dốc.

**Thresholding** là một phương pháp xử lý ảnh trong lĩnh vực thị giác máy tính, được sử dụng để chuyển đổi ảnh xám thành ảnh nhị phân. Phương pháp này dựa trên việc chọn một giá trị ngưỡng (threshold) và gán giá trị nhị phân (thường là trắng hoặc đen) cho mỗi điểm ảnh dựa trên giá trị của nó so với ngưỡng.

Trong quá trình xử lý ảnh để phát hiện làn đường, các phép biến đổi màu sắc và gradient được áp dụng để tách riêng các đặc trưng quan trọng của làn đường khỏi nền ảnh. Các phép biến đổi như chuyển đổi màu sắc sang không gian màu khác, phân tích gradient theo hướng và độ lớn, phát hiện cạnh, và các phép biến đổi khác có thể được sử dụng để tạo ra ảnh nhị phân trong đó các pixel của làn đường được nhận dạng và đánh dấu là "một" (hoặc màu trắng), trong khi các pixel không phải là làn đường được đánh dấu là "không" (hoặc màu đen).

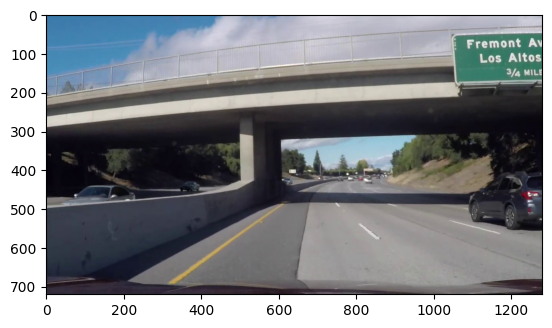
Việc tạo ra ảnh nhị phân giúp làm nổi bật các đặc trưng quan trọng của làn đường và loại bỏ thông tin không cần thiết trong ảnh, từ đó tạo điều kiện thuận lợi cho việc phát hiện và theo dõi làn đường trong quá trình xử lý tiếp theo của thuật toán lane detection.

Có nhiều cách cho quy trình này như dùng Sobel Gradient Thresolding, Color Thresholding, kết hợp,.. Nhóm chúng em đã thử nghiệm các kỹ thuật này trên nhiều ảnh và đối với xử lý trên Sobel Gradient đòi hỏi điều chỉnh tham số (Parameter Tuning) bao gồm như ngưỡng và độ lớn gradient theo trục x và trục y, kích thước ma trận lọc Sobel rất phức tạp, đặc biệt là đối với các video điều kiện khắc nghiệp (thời tiết, mật độ giao thông, chất lượng camera) nên nhóm quyết định kết hợp với phạm vi nghiên cứu đồ án lựa chọn Color Thresholding để trích xuất làn đường.

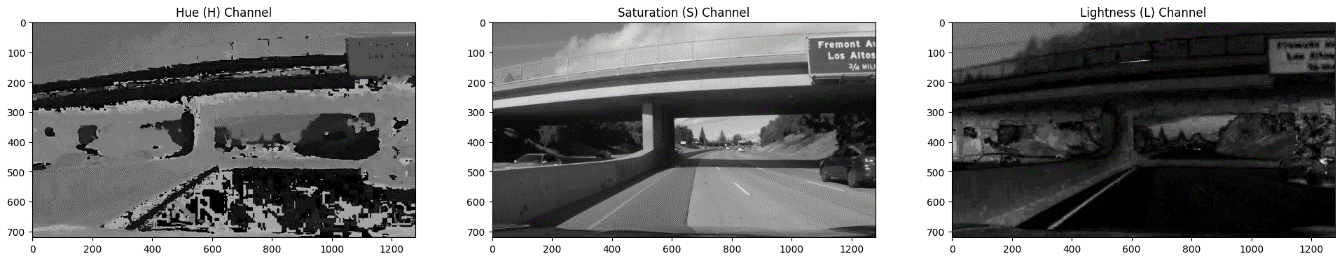
#### Không gian màu

Không gian màu HLS (Hue, Lightness, Saturation) và không gian màu HSV (Hue, Saturation, Value) là hai không gian màu thường được sử dụng trong xử lý ảnh và phân tích màu sắc. Dưới đây là mô tả về cấu trúc và ý nghĩa của từng thành phần trong hai không gian màu này:

1. **Không gian màu HLS:**
   * Hue (Màu sắc): Biểu thị các tông màu cơ bản như đỏ, xanh lá cây, xanh dương, vv. Theo chu kỳ 360 độ, trong đó màu đỏ tương ứng với giá trị gần 0 hoặc 360, màu xanh lá cây tương ứng với giá trị khoảng 120, và màu xanh dương tương ứng với giá trị khoảng 240.
   * Lightness (Độ sáng): Đại diện cho mức độ sáng tối của màu. Giá trị 0 tương ứng với màu đen hoàn toàn, và giá trị 1 tương ứng với màu trắng hoàn toàn.
   * Saturation (Độ bão hòa): Xác định mức độ bão hòa màu. Giá trị 0 đại diện cho màu xám hoàn toàn, trong khi giá trị 1 đại diện cho màu sắc tối đa.



*Ảnh minh họa gốc*



*Ảnh minh họa trong 3 kênh màu của không gian màu HSL*

1. **Không gian màu HSV:**
   * Hue (Màu sắc): Tương tự như không gian màu HLS, đại diện cho các tông màu cơ bản. Giá trị của hue cũng được biểu thị theo chu kỳ 360 độ.
   * Saturation (Độ bão hòa): Xác định mức độ bão hòa của màu. Giá trị 0 đại diện cho màu xám hoàn toàn, và giá trị 1 đại diện cho màu sắc tối đa.
   * Value (Giá trị): Biểu thị độ sáng của màu. Giá trị 0 tương ứng với màu đen hoàn toàn, và giá trị 1 tương ứng với màu sáng nhất.

Cả hai không gian màu HLS và HSV đều cung cấp các thành phần để biểu thị màu sắc và thông tin về độ sáng của một pixel trong hình ảnh. Sự khác biệt chính giữa chúng là cách chúng xử lý độ sáng và bão hòa màu, vì vậy mỗi không gian màu có ứng dụng riêng trong việc phân tích và xử lý ảnh.

#### Ngưỡng (Threshold)

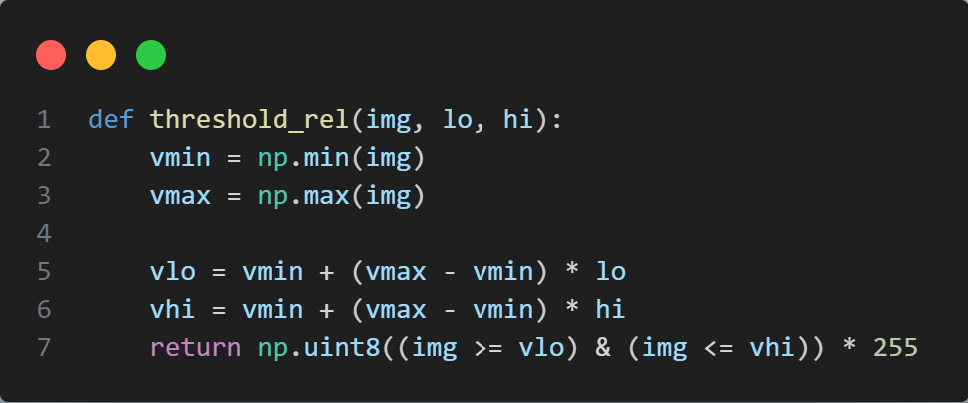
Vì phạm vi của đồ án và giới hạn kỹ thuật trên xử lý hình ảnh, nên ta sẽ mặc định làn đường của ta là đường có vạch kẻ đường phân biệt làn ngược chiều và nếu bến trái là vỉa hè hoặc giải phân cách đường bộ thì cũng sẽ có màu vàng. **Ảnh minh họa:**



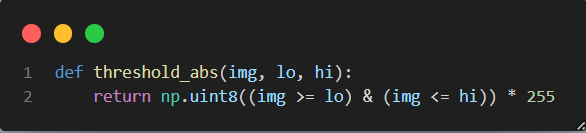


Vậy nên để có 2 loại màu dải đường để xác định làn đường nên ta sẽ sử dụng 2 ngưỡng (threshold) trên 2 không gian màu.

* **Threshold relative**
  + Thực hiện việc áp dụng ngưỡng tương đối (relative thresholding) cho một hình ảnh.
  + Ngưỡng tương đối được xác định bằng hai giá trị **low** (ngưỡng dưới) và **high** (ngưỡng trên) được tính dựa trên giá trị tối thiểu và tối đa của hình ảnh.
  + Các pixel trong hình ảnh mà có giá trị nằm trong khoảng từ **vlo** đến **vhi** (tính toán dựa trên giá trị tối thiểu và tối đa) sẽ được đánh dấu là 255 (màu trắng), còn lại là 0 (màu đen). Do giá trị Lightness của không gian màu HLS có giá trị từ 0 tới 1 xác định phần trăm tương đối của khoảng giá trị cường độ sáng nên ta phải dùng ngưỡng tương đối.



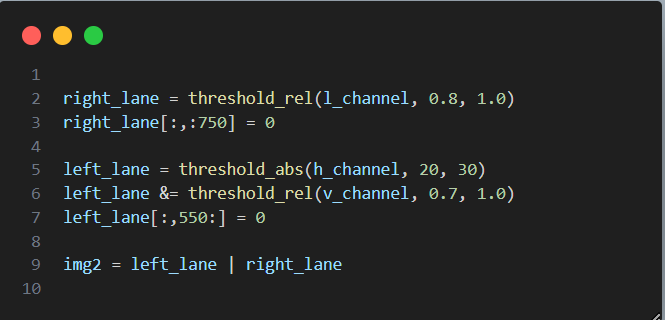
* **Threshold absolute**
  + Thực hiện việc áp dụng ngưỡng tuyệt đối (absolute thresholding) cho một hình ảnh.
  + Ngưỡng tuyệt đối được xác định bằng hai giá trị low (ngưỡng dưới) và high (ngưỡng trên).
  + Các pixel trong hình ảnh mà có giá trị nằm trong khoảng từ lo đến hi sẽ được đánh dấu là 255 (màu trắng), còn lại là 0 (màu đen).



#### Tạo ảnh nhị phân dựa trên các ngưỡng

##### Gradient Thresholding

##### HLS thresholding

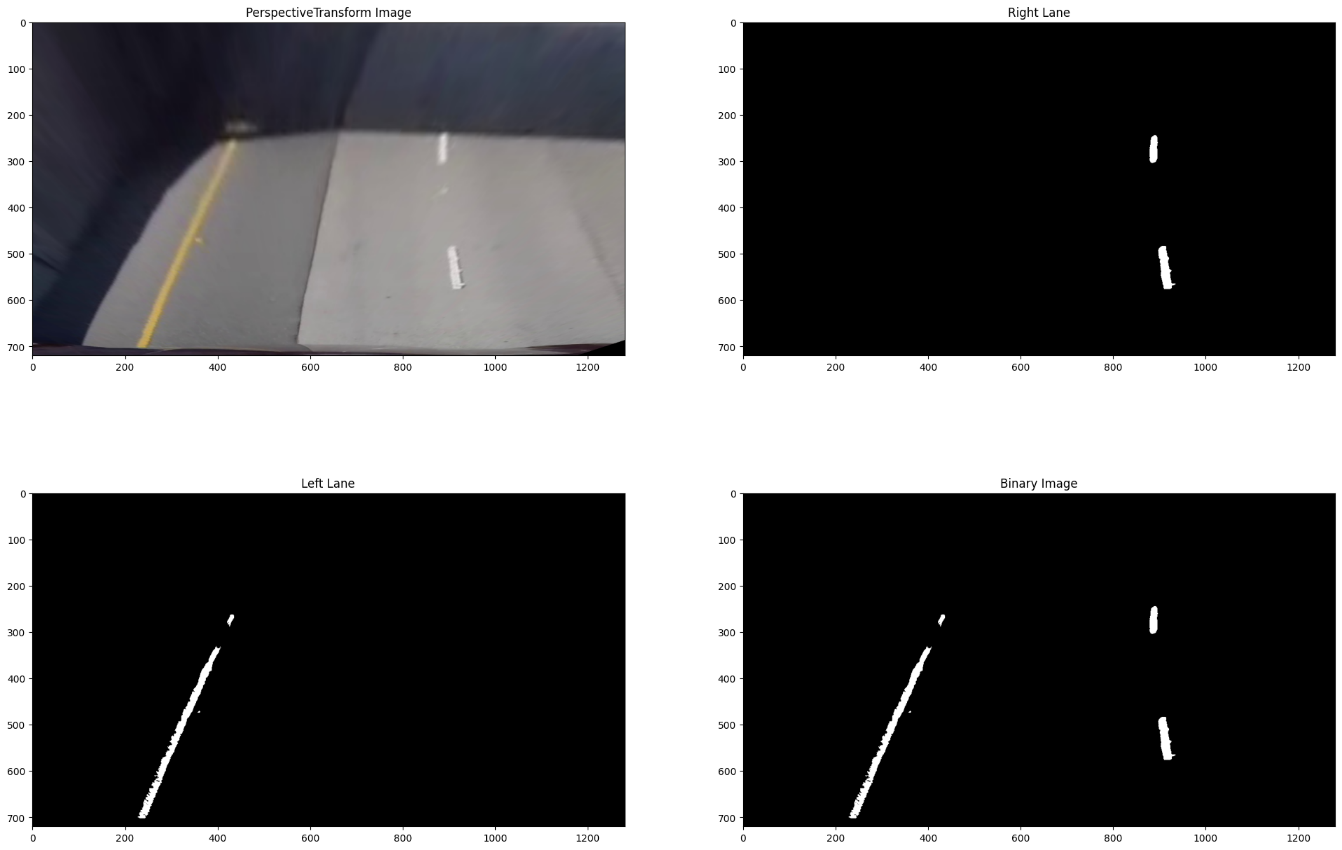


* Áp dụng hàm **threshold\_rel** trên kênh Lightness (l\_channel) của không gian màu HLS để tạo ra một ảnh nhị phân cho làn đường phía bên phải. Giá trị ngưỡng thấp và ngưỡng cao được đặt là 0.8 và 1.0, tức là chúng ta chỉ quan tâm đến phần trăm 20% trên cùng của khoảng giá trị ánh sáng trong kênh màu L (độ sáng) của không gian màu HLS. Kết quả là một ảnh nhị phân với các pixel thuộc làn đường phía bên phải được đặt thành 255 và các pixel khác được đặt thành 0.
* Đặt giá trị 0 cho các pixel nằm trong vùng từ cột đầu tiên đến cột thứ 750 của **right\_lane** và từ hàng đầu tiên đến hàng cuối cùng giúp loại bỏ các vùng không cần thiết hoặc nhiễu từ phía bên trái của đường xe bên phải.
* Áp dụng hàm **threshold\_abs** lên kênh Hue (h\_channel) của không gian màu HLS để tạo ra ảnh nhị phân left\_lane. Các pixel có giá trị trong khoảng từ 20 đến 30 (đại diện cho màu vàng) được coi là relavant và được gán giá trị 255 (trắng), trong khi các pixel khác được gán giá trị 0 (đen).
* Tiếp tục áp dụng hàm **threshold\_rel** lên kênh Value (v\_channel) của không gian màu HSV và thực hiện phép toán bitwise AND với left\_lane. Điều này có ý nghĩa là chỉ giữ lại các vùng trong left\_lane mà cả hai điều kiện threshold là True (giá trị pixel trong khoảng từ 0.7 đến 1.0 và giá trị pixel từ **h\_channel** là relavant).
* Đặt giá trị 0 cho các pixel nằm trong vùng từ cột thứ 550 đến cột cuối cùng của **left\_lane** và từ hàng đầu tiên đến hàng cuối cùng giúp loại bỏ các vùng không cần thiết hoặc nhiễu từ phía bên phải của đường xe bên trái.
* Cuối cùng, thực hiện phép toán bitwise OR giữa **left\_lane** và **right\_lane** để kết hợp các vùng relavant từ cả hai làn đường. Kết quả được gán cho biến img2, là ảnh nhị phân kết hợp của cả hai làn đường.

#### Kết quả minh họa



*Ảnh minh họa*



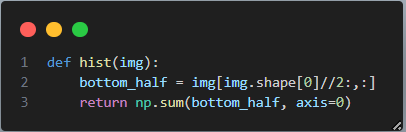
*Lần lượt ảnh đã được biến đổi phối cảnh, ảnh nhị phân bên phải làn đường, bên trái làn đường và kết hợp cuối cùng*

### Phát hiện và vẽ làn đường

#### Tìm tọa độ pixel của làn đường

Ta sẽ nhận đầu vào là một ảnh nhị phân (binary warped image) và và trả về các pixel của làn đường trái, làn đường phải và một hình ảnh RGB (out\_img) được sử dụng để hiển thị kết quả.

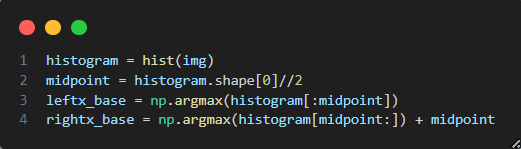
* Tính toán histogram của ảnh nhị phân – đã xác định ở phần trước (để xác định vị trí ban đầu của làn đường trái và phải dựa trên phân bố pixel trong histogram). Histogram của một ảnh là biểu đồ thống kê hiển thị phân bố của các giá trị pixel trong ảnh. Nó biểu thị số lượng pixel có cùng giá trị nằm trong các khoảng giá trị khác nhau.



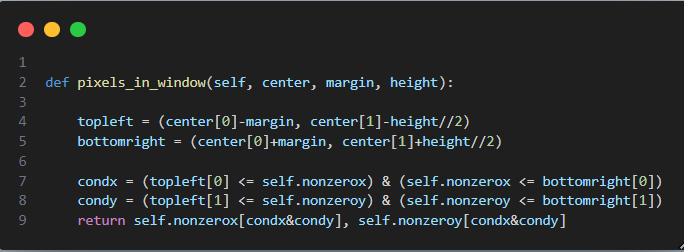
*Hàm tính histogram của một ảnh nhị phân theo chiều ngang. Hàm này có tác dụng tính tổng số lượng pixel có giá trị 1 (hoặc mức độ sáng khác 0) trong nửa dưới của ảnh.*

Lý do ta chỉ lấy nửa dưới của ảnh trong nhận dạng làn đường. Làn đường xuất hiện chủ yếu trong phạm vi dưới của ảnh. Do đó, việc tập trung vào nửa dưới của ảnh giúp giảm thiểu ảnh hưởng của các yếu tố không quan trọng ở phần trên. Ngoài ra, việc chỉ quan tâm đến nửa dưới của ảnh cũng có thể giảm thiểu ảnh hưởng của các yếu tố khác như nhiễu, ánh sáng không đồng đều, hoặc các vật thể phía trên không gian quan sát như cầu vượt, phương tiện từ xa, ...

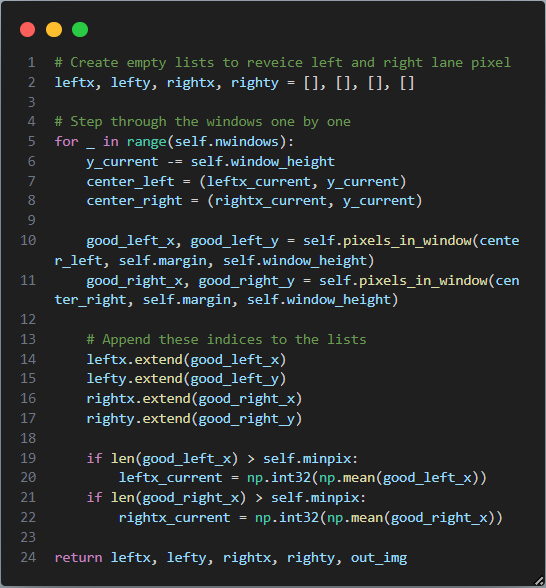
* Sau khi tính toán được histogram của ảnh nhị phân, ta có thể xác định vị trí ban đầu của làn đường trái (**leftx\_base**) và làn đường phải (**rightx\_base**) dựa trên các cột histogram trái và phải.



* Vì các làn đường trái phải là dạng đường kéo dài ở dạng binary image nên ta phải thực hiện tìm kiếm tất cả điểm ảnh.
  + Thiết lập vị trí hiện tại của cửa sổ tìm kiếm, bắt đầu từ vị trí ban đầu của làn đường trái và phải.
  + Tạo các danh sách rỗng (**leftx**, **lefty**, **rightx**, **righty**) để lưu các pixel của làn đường trái và phải.
  + Lặp qua các cửa sổ (windows) một cách tuần tự từ trên xuống dưới.
  + Trong mỗi cửa sổ, xác định tọa độ trung tâm của cửa sổ hiện tại và tìm các pixel rơi vào cửa sổ đó (dựa trên tọa độ trung tâm, margin và chiều cao cửa sổ).

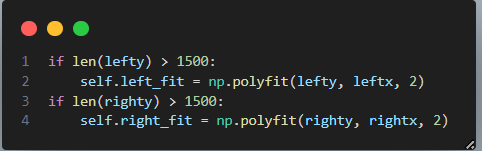


* **topleft**: Tọa độ của góc trái trên của cửa sổ được tính bằng cách lấy tọa độ x của trung tâm cửa sổ trừ đi nửa chiều rộng, và tọa độ y của trung tâm cửa sổ trừ đi nửa chiều cao.
* **bottomright**: Tọa độ của góc phải dưới của cửa sổ được tính bằng cách lấy tọa độ x của trung tâm cửa sổ cộng thêm nửa chiều rộng, và tọa độ y của trung tâm cửa sổ cộng thêm nửa chiều cao.
* **condx**: Tạo một mảng điều kiện kiểm tra xem các pixel có tọa độ x nằm trong khoảng từ topleft[0] đến bottomright[0] không.
* **condy**: Tạo một mảng điều kiện kiểm tra xem các pixel có tọa độ y nằm trong khoảng từ topleft[1] đến bottomright[1] không.
* self.nonzerox[condx&condy]: Lấy các tọa độ x của các pixel mà cả hai điều kiện condx và condy đều là True.
* self.nonzeroy[condx&condy]: Lấy các tọa độ y của các pixel mà cả hai điều kiện condx và condy đều là True.
  + Gắn các pixel tìm được vào danh sách tương ứng (**leftx**, **lefty**, **rightx**, **righty**).
  + Nếu số lượng pixel tìm được trong cửa sổ đó lớn hơn một ngưỡng (**minpix**), cập nhật lại vị trí hiện tại của làn đường trái và phải bằng cách lấy giá trị trung bình của các pixel tìm được.
  + Trả về các pixel của làn đường trái, làn đường phải và hình ảnh gốc (**leftx**, **lefty**, **rightx**, **righty**, **out\_img**).

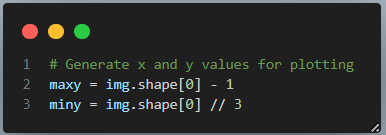


#### Vẽ đường cong

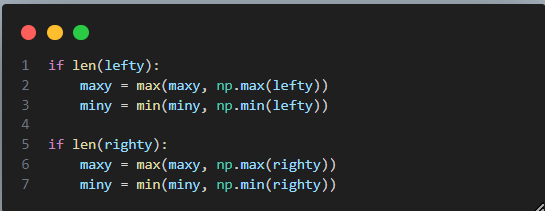
Nếu số lượng pixel ở 2 đường lớn hơn 1500 thì sử dụng phương pháp polyfit là một phương pháp trong đại số tuyến tính để tìm một đường curve (đường cong) tốt nhất để khớp với một tập hợp các điểm dữ liệu. Trong trường hợp này, nó được sử dụng để tìm một đường curve để khớp với các điểm pixel thuộc lane bên trái và phải.



* Tiếp tục khởi tạo **maxy** là chỉ số hàng tối đa của ảnh trừ 1 (do chỉ số hàng bắt đầu từ 0) và **miny** là chỉ số hàng tối thiểu của ảnh chia cho 3. Điều này giúp xác định khoảng giới hạn y mà đường curve sẽ được vẽ, giới hạn này giúp hạn chế vùng không mong muốn.



* Kiểm tra xem có tồn tại các giá trị trong danh sách **lefty** và **righty** hay không. Nếu tồn tại, nghĩa là đã tìm được các điểm pixel thuộc lane tương ứng với **leftx** và **rightx**, và chúng có đủ số lượng lớn hơn 1500.
  + Nếu **len(lefty)** lớn hơn 0 (tức danh sách **lefty** không rỗng), ta sẽ cập nhật **maxy** bằng giá trị lớn nhất trong danh sách **lefty** và **miny** bằng giá trị nhỏ nhất trong danh sách **lefty.**
  + Tương tự, nếu **len(righty)** lớn hơn 0 (tức danh sách **righty** không rỗng), ta cập nhật **maxy** bằng giá trị lớn nhất trong danh sách **righty** và **miny** bằng giá trị nhỏ nhất trong danh sách **righty**.



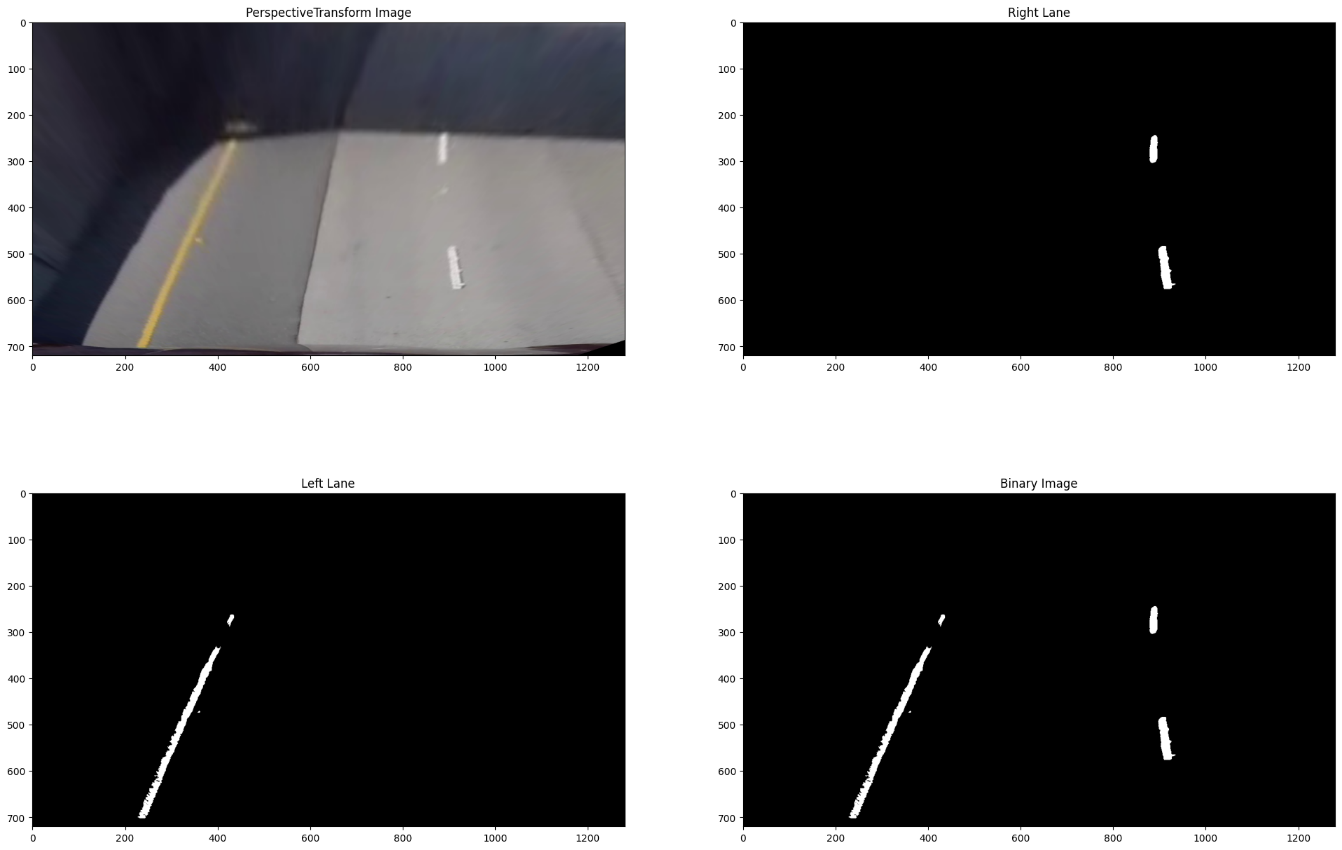
* Sử dụng np.linspace để tạo một mảng ploty chứa các giá trị y tương ứng với mỗi hàng của ảnh. Mảng ploty này sẽ được sử dụng để tính toán giá trị x của đường curve.
* Sử dụng các hệ số ước tính (self.left\_fit và self.right\_fit) và mảng ploty để tính toán các giá trị x tương ứng của đường curve cho lane trái (left\_fitx) và lane phải (right\_fitx). Các công thức tính toán đường curve dựa trên phương trình đường curve bậc hai.
* Tiến hành vẽ đường curve lên ảnh out\_img bằng cách sử dụng vòng lặp. Với mỗi giá trị y trong ploty, các giá trị x tương ứng (left\_fitx[i] và right\_fitx[i]) được chuyển đổi thành kiểu số nguyên và sử dụng để vẽ một đường thẳng ngang tại vị trí đó trên ảnh out\_img sử dụng hàm cv2.line. Điều này tạo ra đường curve liên kết các điểm trên ảnh.



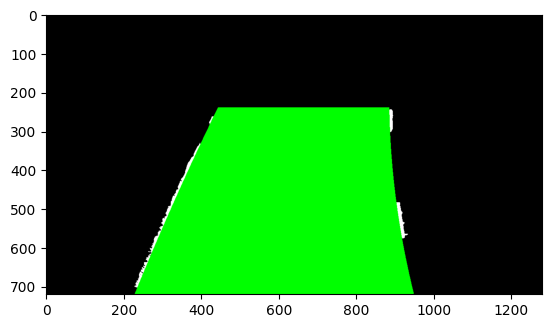
#### Kết quả minh họa

Ta cùng xem lại kết quả từ đầu tới pipline hiện tại:





*Ảnh nhị phân áp dụng các ngưỡng*



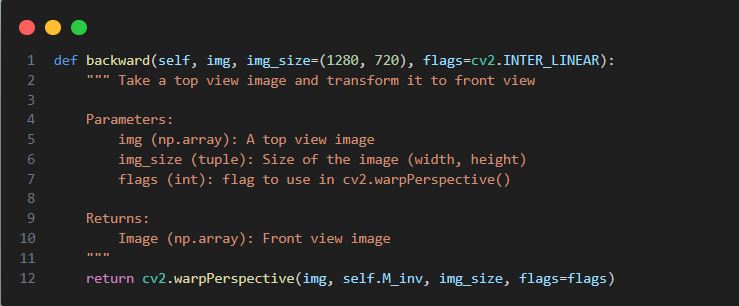
*Ảnh sau khi phát hiện và vã làn đường*

Từ lúc áp dụng biến đổi phối cảnh từ góc nhìn camera sang góc nhìn từ trên cao (bird's-eye view) của đường đi, giờ ta sẽ biến đổi phối cảnh backward ngược lại (chuyển từ top view sang front view). Để làm điều này, hàm sử dụng phép biến đổi perspective, được thực hiện bằng hàm **cv2.warpPerspective()**.

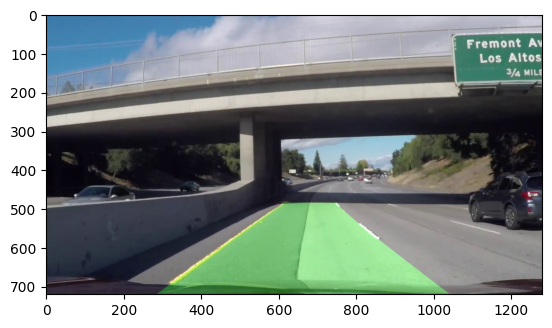
Các tham số của hàm **backward** bao gồm:

* **img** là ảnh góc nhìn trên (top view image) mà chúng ta muốn chuyển đổi.
* **img\_size** là kích thước của ảnh đầu ra sau khi chuyển đổi (width, height). Giá trị mặc định là (1280, 720).
* **flags** là một cờ (flag) sử dụng trong hàm **cv2.warpPerspective()** để xác định phương pháp biến đổi ảnh. Giá trị mặc định là **cv2.INTER\_LINEAR**, đại diện cho phương pháp nội suy tuyến tính.

Hàm **cv2.warpPerspective()** nhận đầu vào là ảnh **img** và ma trận biến đổi nghịch đảo **self.M\_inv**, và thực hiện việc biến đổi ảnh từ góc nhìn trên xuống góc nhìn phía trước. Kết quả là ảnh front view.



Cuối cùng, hàm trả về ảnh front view đã chuyển đổi.

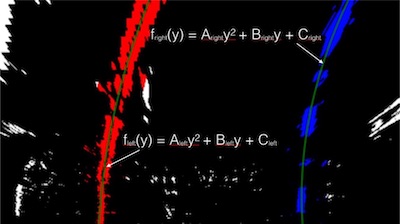


*Kết quả sau khi chuyển ảnh về frontview*

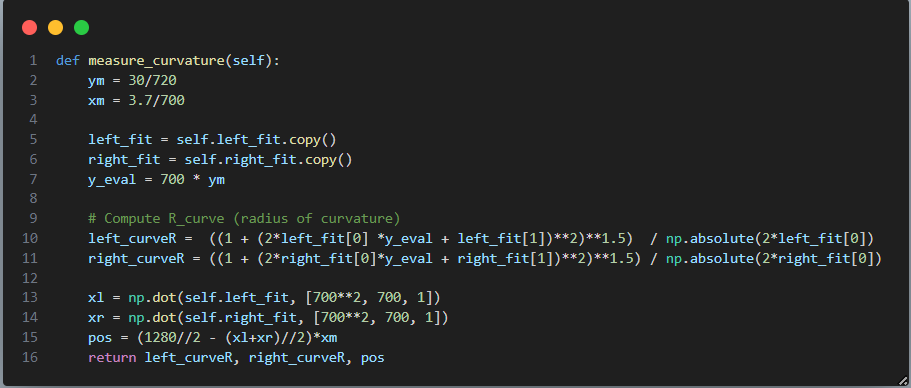
### Xác định độ cong của làn đường và vị trí của phương tiện so với trung tâm.

#### Độ cong của làn đường

Giờ đây khi các pixel làn đường đã được định vị, ta có thể tìm kiếm các vị trí pixel x và y để khớp với đường cong đa thức bậc 2 có dạng:



Ở đây ta chỉ điều chình cho hàm f(y) vì các vạch làn đường bị biến dạng thay đổi gần như thẳng đứng và có thể có nhiều giá trị X cho một giá trị Y.



*Hàm thực hiện tính toán độ cong của đường lái (lane curvature) và vị trí của xe so với trung tâm của làn đường.*

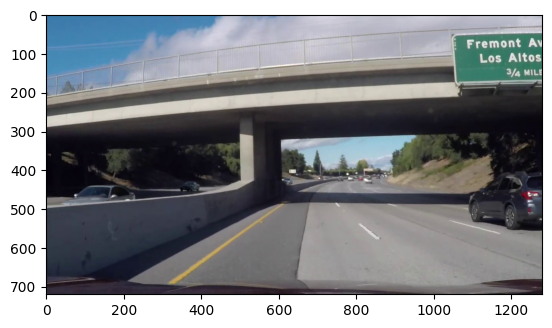
* Định nghĩa **ym** và **xm** là hệ số chuyển đổi từ pixel sang mét theo chiều dọc và chiều ngang tương ứng. Ví dụ: ym = 30/720 nghĩa là mỗi pixel trong chiều dọc tương ứng với 30/720 mét.
* Sao chép các hệ số đường cong của đường lái từ self.left\_fit và self.right\_fit vào các biến left\_fit và right\_fit để tiến hành tính toán. Hai hệ số đường cong thường được biểu diễn dưới dạng một đa thức bậc hai (quadratic polynomial) trong hình thức:
  + **left\_fit: [a, b, c]**
  + **right\_fit: [a, b, c]**

Trong đó:

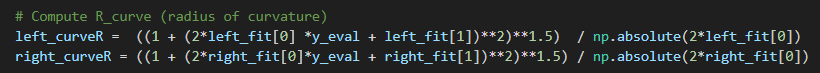
* + a là hệ số của thành phần bậc hai (x^2)
  + b là hệ số của thành phần bậc nhất (x)
  + c là hệ số của thành phần tự do (hằng số)

Các hệ số này được sử dụng để tính toán vị trí của các điểm trên đường lái bên trái và bên phải dựa trên giá trị y\_eval (vị trí đánh giá) và công thức đường cong.

* Tính giá trị y tương ứng với vị trí đánh giá độ cong (**y\_eval**). Trong trường hợp này, giá trị **y\_eval** được chọn là 700 (trong tọa độ pixel) và được chuyển đổi sang đơn vị mét bằng cách nhân với ym. Trong trường hợp này, giá trị y\_eval là 700 được chọn để tính toán đường cong tại một điểm trên trục y có giá trị tương đương với chiều cao của ảnh đầu vào (720 pixels). Bằng cách chọn giá trị này, chúng ta có thể đánh giá đường cong ở một vị trí gần cuối của hình ảnh, tương ứng với vị trí gần cuối của đường lái trên ảnh.



* Tính độ cong của đường lái bên trái (left\_curveR) và bên phải (right\_curveR) dựa trên hệ số đường cong và vị trí đánh giá. Công thức tính toán độ cong dựa trên công thức tích phân của đường cong:



#### Khoảng cách của xe với trung tâm

* Tính vị trí của xe so với trung tâm của làn đường. Đầu tiên, tính toán vị trí x của đường lái bên trái (xl) và bên phải (xr) tại giá trị y\_eval bằng cách nhân hệ số đường cong với [700^2, 700, 1] (tương ứng với công thức ax^2 + bx + c).
* Trả về độ cong của đường lái bên trái (left\_curveR), độ cong của đường lái bên phải (right\_curveR) và vị trí của xe so với trung tâm của làn đường (pos) để sử dụng trong quá trình xử lý tiếp theo.

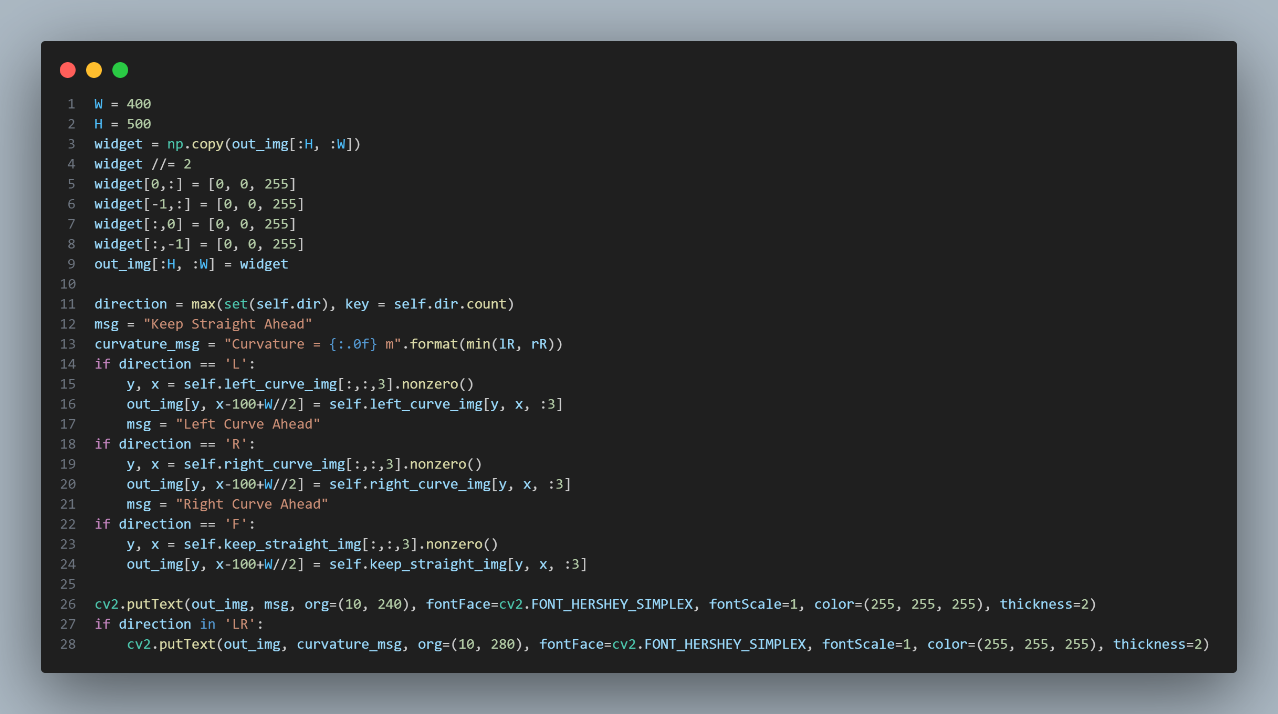
### Hiển thị dữ liệu và cảnh báo lên ảnh

Sau khi đã có thông số của làn đường, ta có thể hiển thị các dữ liệu hoặc các cảnh bảo suy luận từ dữ liệu lên ảnh đầu ra.

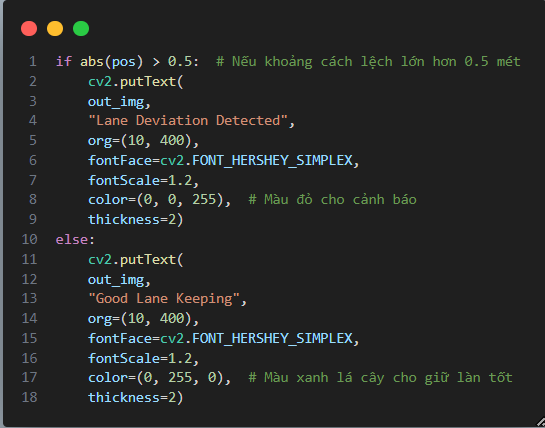
* Hướng lái dựa trên hệ số đường cong của đường lái bên trái (**self.left\_fit[0]**) và đường lái bên phải (**self.right\_fit[0]**). Nếu giá trị tuyệt đối của hệ số đường cong lớn hơn 0.00015, thì được xác định là lái thẳng (**'F'**). Nếu hệ số đường cong âm, thì được xác định là rẽ trái (**'L'**). Các trường hợp còn lại được xác định là rẽ phải (**'R'**). Hướng lái được lưu vào danh sách **self.dir** và giới hạn độ dài của danh sách là 10.

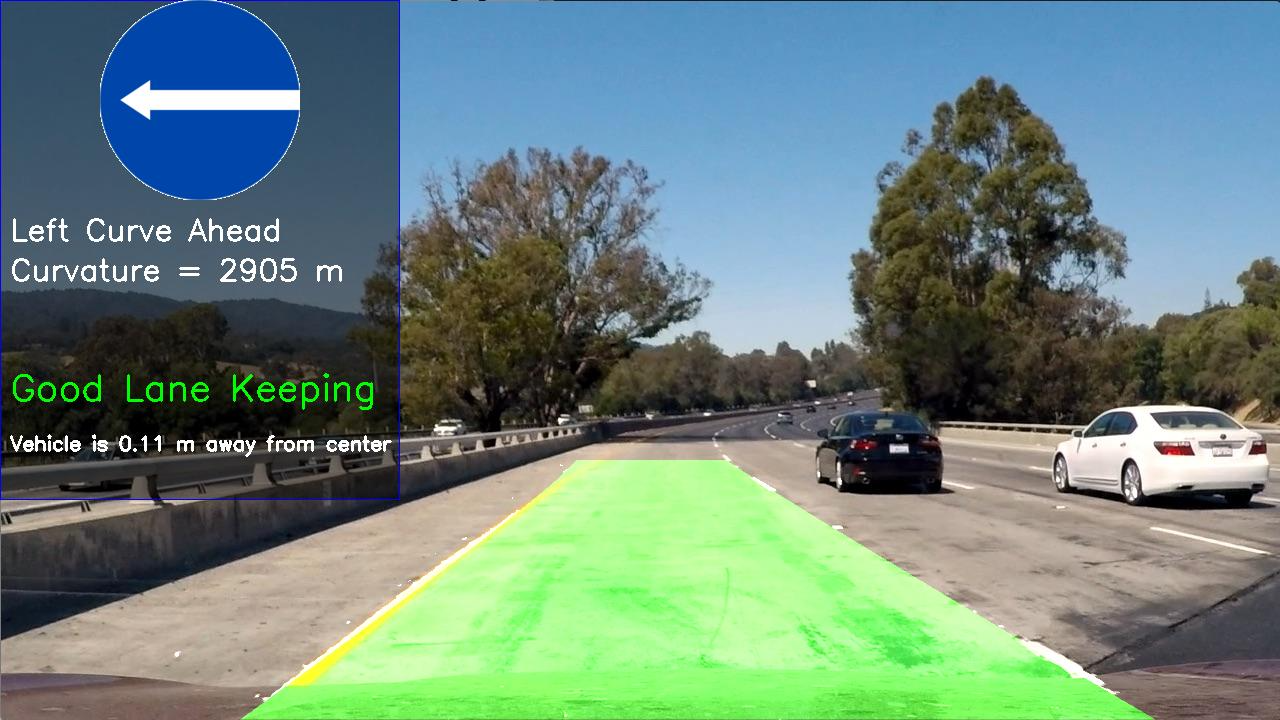


* Tiếp theo, hàm tạo một widget (khu vực nhỏ) trên ảnh đầu ra và hiển thị các chỉ báo về hướng lái và độ cong của đường lái lên widget này.
* Dựa vào hướng lái, hàm xác định thông điệp **msg** và vị trí các điểm trên ảnh liên quan đến đường cong của đường lái (**self.left\_curve\_img**, **self.right\_curve\_img**, **self.keep\_straight\_img**). Thông điệp **msg** là một thông điệp về hướng lái, trong đó "Left Curve Ahead" (Rẽ trái), "Right Curve Ahead" (Rẽ phải) và "Keep Straight Ahead" (Lái thẳng) có thể được hiển thị. Thông điệp **curvature\_msg** chứa giá trị độ cong của đường lái (**lR** hoặc **rR**) và được hiển thị nếu hướng lái là "L" hoặc "R".



* Nếu khoảng cách lệch với trung tâm lớn hơn 0.5 mét, thì sẽ hiển thị thông báo "Lane Deviation Detected" với màu đỏ. Ngược lại, nếu giữ được làn đường tốt (khoảng cách lệch nhỏ hơn hoặc bằng 0.5 mét), sẽ hiển thị thông báo "Good Lane Keeping" với màu xanh lá cây.





*Ảnh minh họa widget*

## Kỹ thuật dựa trên deep learning

### Mô hình SCNN-Tensorflow

### Mô hình SCNN-Tensorflow

# Xây dựng giao diện người dùng (GUI) ứng dụng kết quả

# Kết quả và đánh giá

## Bộ dữ liệu sử dụng

Bộ dữu liệu được nhóm sử dụng trong huấn luyện và đánh giá là [**CuLane**](https://paperswithcode.com/dataset/culane). Bộ dữ liệu CuLane là một bộ dữ liệu phổ biến được sử dụng cho nhiệm vụ phát hiện làn đường trong lĩnh vực xe tự hành và thị giác máy tính. Bộ dữ liệu này được tạo ra bởi nhóm nghiên cứu tại trường Đại học Trung Quốc ([Xiaohang Zhan](https://xiaohangzhan.github.io/) , Jun Li và Xudong Cao) và chứa các hình ảnh đường phố từ nhiều góc nhìn và điều kiện ánh sáng khác nhau.



**CULane** là một bộ dữ liệu đầy thách thức quy mô lớn dành cho nghiên cứu học thuật về phát hiện làn đường giao thông. Nó được thu thập bởi các camera gắn trên sáu phương tiện khác nhau do các tài xế khác nhau điều khiển ở Bắc Kinh. Hơn 55 giờ video đã được thu thập và 133.235 khung hình đã được trích xuất. Tập dữ liệu được chia thành 88880 hình ảnh cho tập huấn luyện, 9675 cho tập xác thực và 34680 cho tập kiểm tra. Bộ bài kiểm tra được chia thành loại bình thường và 8 loại thử thách.

## Kết quả

### Phương pháp xử lý ảnh

#### Kết quả đạt được và nhược điểm

* Giải pháp này hoạt động tốt với điều kiện ánh sáng bình thường. Tuy nhiên, nó cần được cải thiện để giải quyết các trường hợp sử dụng khác nhau. Một ví dụ mà nó có thể được cải thiện là đối với các làn đường có một phần của làn đường là đường mới trải nhựa và có màu khác với phần còn lại của làn đường là đường trải nhựa cũ hơn. Thuật toán này cũng cần được cải thiện trong trường hợp máy ảnh bị lóa do ánh sáng mặt trời chiếu trực tiếp vào camera và trong các trường hợp có độ tương phản cao khác khi các vạch phân làn dường như bị mờ khiến chúng khó bị phát hiện hơn. Những loại tình huống này có thể được giải quyết bằng cách tự động điều chỉnh độ tương phản của hình ảnh để đảm bảo các đường phân làn trong ảnh không bị mờ và để đảm bảo thuật toán có dải động tốt trong mọi điều kiện ánh sáng. Trong trường hợp đường quanh co và có độ dốc, nó sẽ gây khó khăn cho việc làm cong hình ảnh đúng cách và có thể gây ra sự cố cho thuật toán. Vấn đề này cũng có thể được giải quyết bằng cách tạo một vùng quan tâm động cho từng khung hình.
* thuật toán giả định tính song song của các đường làn đường trên các hình ảnh bị biến dạng (ví dụ: khi nó sử dụng khoảng cách đều để vẽ một đường vắng mặt), điều này không đúng trong trường hợp địa hình đồi núi và trong các tình huống như vậy, thuật toán có thể thất bại.
* Ánh sáng chói trên đường, chẳng hạn như xuất hiện dưới tán cây, có thể dẫn đến kết quả nhiễu. Các vấn đề khác có thể xảy ra do tình trạng kém của vạch kẻ đường, giao lộ của các đường khác nhau. Nó có thể được giải quyết một phần bằng cách thêm filtering  giữa các khung hình video.
* Tuy nhiên, vấn đề chính là không có vạch kẻ đường hoặc khả năng tàng hình của chúng. Các vạch kẻ trên đường có thể vô hình do bụi hoặc, thường xảy ra hơn, do tuyết phủ hoặc lá mùa thu trên đường. Tuyết phủ một phần cũng có thể gây nhầm lẫn cho thuật toán vì gió có thể tạo thành các mô hình tuyết phức tạp trên đường. Các thuật toán tinh vi hơn (chẳng hạn như mạng lưới thần kinh sâu) nên được áp dụng trong trường hợp không có vạch để dự đoán và xác định vị trí của làn.

#### Hướng phát triển

* Thuật toán lọc video có thể được cải thiện để làm cho nó mạnh mẽ trong thế giới thực khác.
* Cần lưu ý rằng trong trường hợp đường đứt nét, thuật toán tìm điểm xác định từng nét đứt riêng lẻ là một nhóm điểm. Vì vậy, người ta có thể theo dõi chúng để có một số ước tính tốc độ.
* Có thể là một ý kiến ​​hay khi thay đổi thuật toán tìm điểm của đường để sử dụng các tham số tương tự cho các đường màu vàng và trắng.

## Đánh giá hiệu suất hệ thống

## Hướng phát triển

# Tài liệu tham khảo