Tài Liệu Hướng Dẫn Lập Trình Xử Lý Ảnh

Chương III: PHÁT HIỆN LÀN ĐƯỜNG VÀ ĐIỀU KHIỂN GÓC LÁI XE TỰ HÀNH

1. Hiệu chỉnh camera:

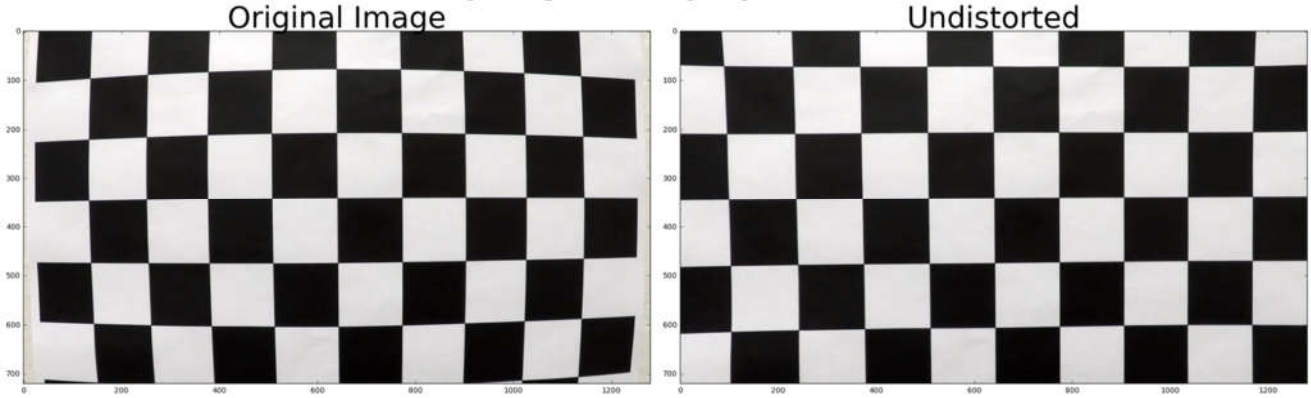
Camera thực tế sử dụng các ống kính cong để hình thành một hình ảnh, và các tia sáng thường uốn cong quá nhiều hoặc quá ít ở các cạnh của các ống kính này. Điều này tạo ra hiệu ứng bóp méo các cạnh của hình ảnh, do đó các đường kẻ hoặc các đối tượng xuất hiện nhiều hoặc ít cong hơn thực tế. Đây được gọi là biến dạng xuyên tâm, là loại biến dạng phổ biến nhất.
Có ba hệ số cần thiết để biến dạng xuyên tâm: k1, k2, và k3. Để sửa sự xuất hiện của các điểm bị bóp méo hình ảnh trong một hình ảnh, người ta có thể sử dụng một công thức chỉnh sửa được đề cập dưới đây.
Trong các phương trình sau, (x, y) là một điểm trong hình ảnh méo. Để undistort những điểm này, OpenCV tính r, đó là khoảng cách được biết giữa một điểm trong một hình ảnh đã sửa méo (undistorted: đã chỉnh lại) (xcorrected, ycorrected) và tâm của ảnh bị méo , mà thường là tâm của hình ảnh đó (xc, yc) . Điểm trung tâm này (xc, yc) đôi khi được gọi là điểm tâm bóp méo. Những điểm này được mô tả ở trên.
Chức năng **do_calibration ()** thực hiện các thao tác sau:

- Đọc ảnh chessboad và chuyển đổi sang ảnh xám gray scale.
- Tìm góc của chessboard .
- Chúng ta bắt đầu bằng cách chuẩn bị các điểm đối tượng (objpoints), nó sẽ là tọa độ (x, y, z) của các góc ảnh bàn cờ trên thế giới. Ở đây chúng ta giả định rằng ảnh bàn cờ đã được cố định trên mặt phẳng (x, y) tại z = 0 sao cho các điểm đối tượng giống nhau cho mỗi ảnh hiệu chuẩn. Do đó, objpoints chỉ là một mảng được lặp đi lặp lại của các tọa độ, và objpoints sẽ được nối với một bản sao của nó mỗi khi chúng ta phát hiện thành công tất cả các góc chessboard trong một hình ảnh thử nghiệm. imgpoints sẽ được nối với vị trí pixel (x, y) của mỗi góc của mặt phẳng hình ảnh với mỗi lần phát hiện ảnh bàn cờ thành công.
- Thực hiện **cv2.calibrateCamera ()** để tính hệ số co-efficients và ma trận camera mà chúng ta cần phải chuyển đổi các điểm đối tượng 3D thành điểm ảnh 2D.
- Lưu giá trị hiệu chuẩn trong tệp **camera_cal / camera_cal.p** để sử dụng sau.

Hàm **get_camera_calibration()** để đọc các giá trị calibration từ **camera_cal/camera_cal.p** file.

2. Sửa méo ảnh:

Sử dụng các hệ số biến dạng và ma trận camera thu được từ giai đoạn hiệu chỉnh camera, chúng ta loại méo ảnh sử dụng chức năng **cv2.undistort**.
Một ảnh cờ vua mẫu và hình ảnh không bị bóp méo tương ứng:



Bằng cách hiệu chỉnh méo , chúng ta thấy rằng các đường chessboard xuất hiện song song với hình ảnh thô ban đầu.
Kiểm tra sửa méo ảnh với các hình ảnh thu được từ camera:

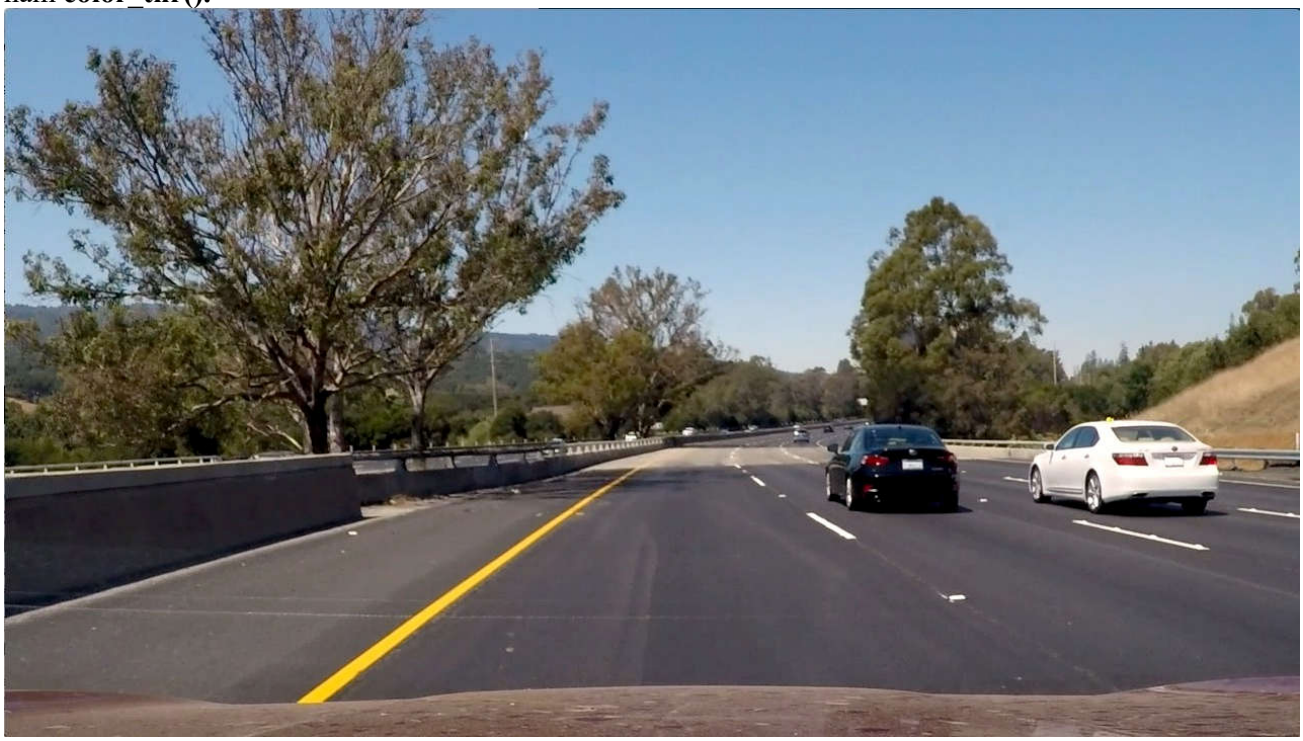
Chúng ta có thể thấy rằng chiếc xe bên trái xuất hiện sẽ được shift đi so với ảnh gốc ban đầu.

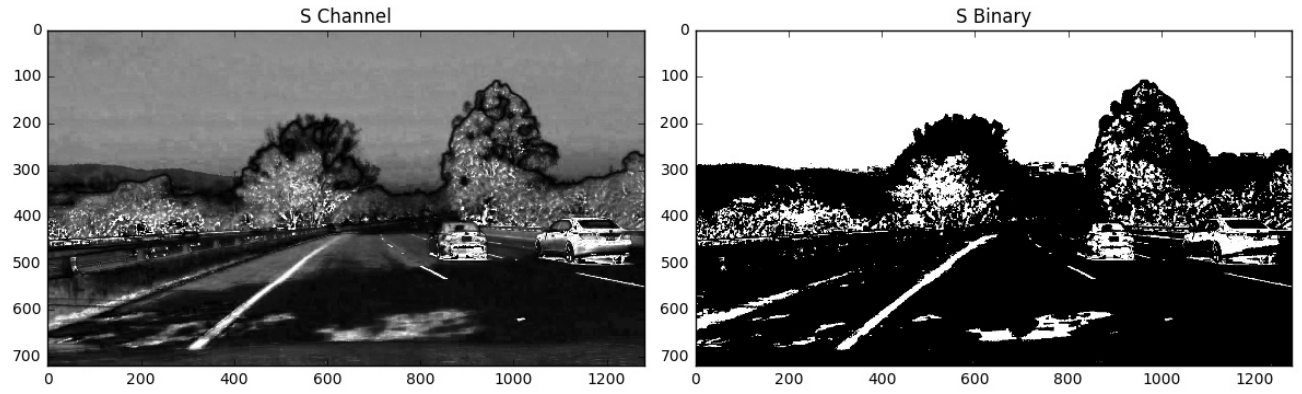
3. Tạo một ảnh nhịn phân sử dụng color transforms và gradients

Trong giai đoạn nhị phân ảnh, nhiều phép biến đổi được áp dụng và kết hợp lại để có được hình ảnh nhị phân tốt nhất để phát hiện làn đường. Chương trình sử dụng cho nhị phân ảnh là *thresholding_main.py*. Chi tiết các bước trong đoạn code này sẽ được giải thích dưới đây:

a. Saturation thresholding:

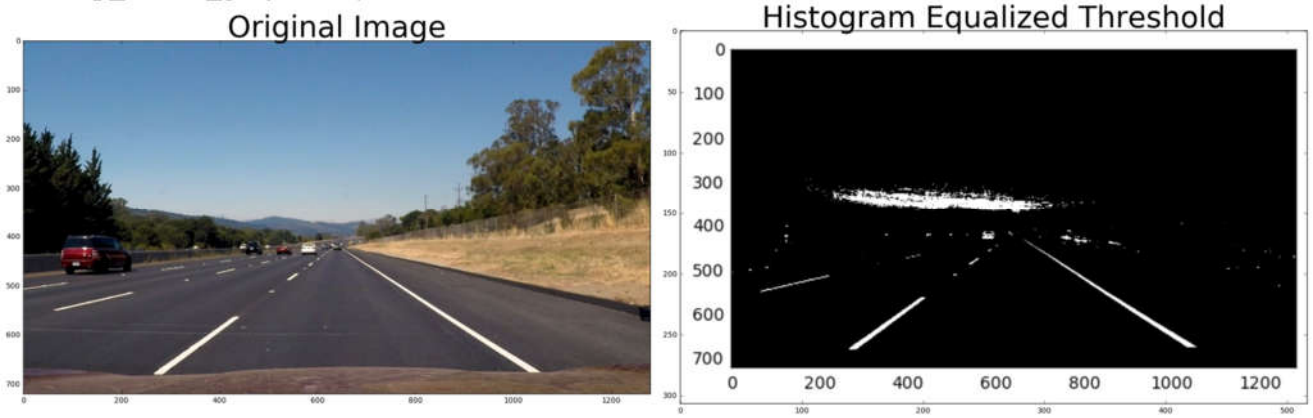
Các hình ảnh được chuyển đổi sang không gian màu HLS để có được các giá trị bão hòa, các làn đường màu vàng được phát hiện tốt nhất trong không gian màu bão hòa. Thao tác ngưỡng này được gọi trong hàm **color_thr().**





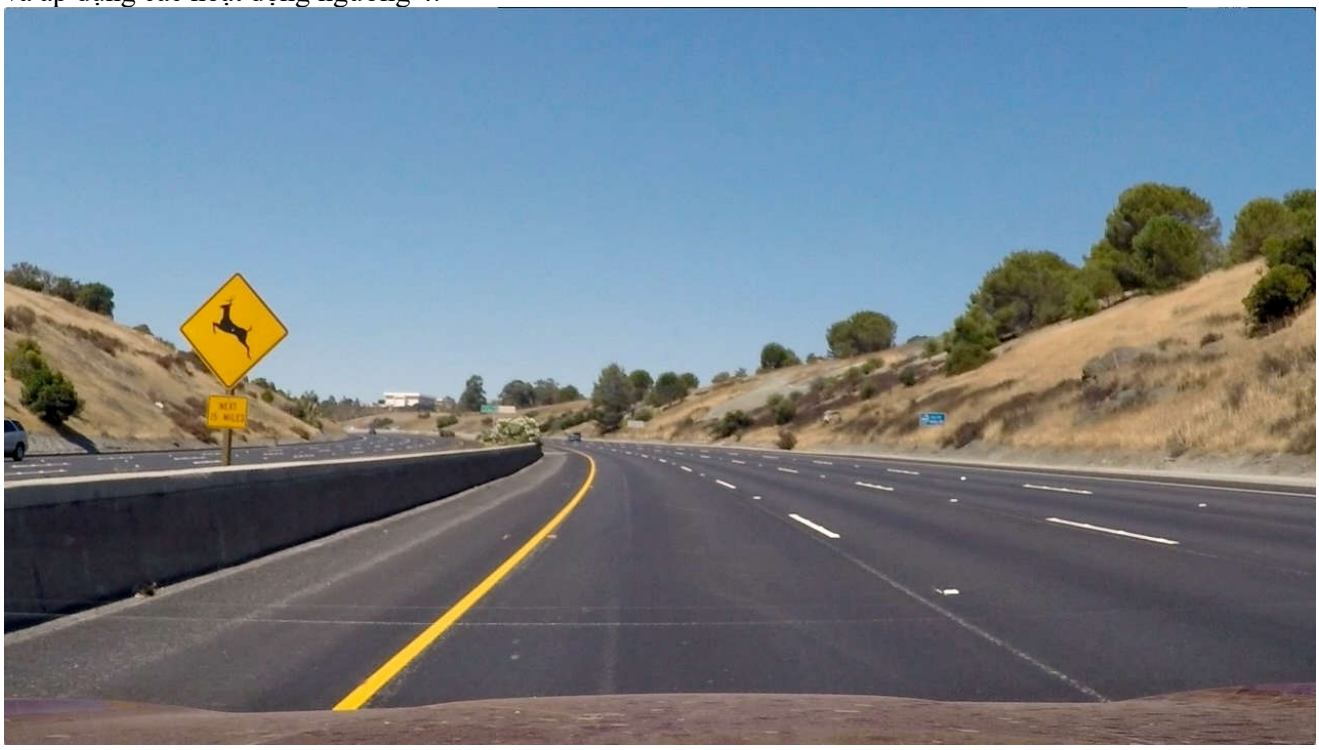
b. Histogram equalized thresholding:

Các hình ảnh được chuyển đổi sang dạnh ảnh xám và cân đối histogram bằng lệnh **cv2.equalizeHist()**, đường màu trắng được phát hiện tốt nhất bằng cách sử dụng phương pháp này. Thao tác ngưỡng này được gọi trong hàm **adp_thresh_grayscale ().**

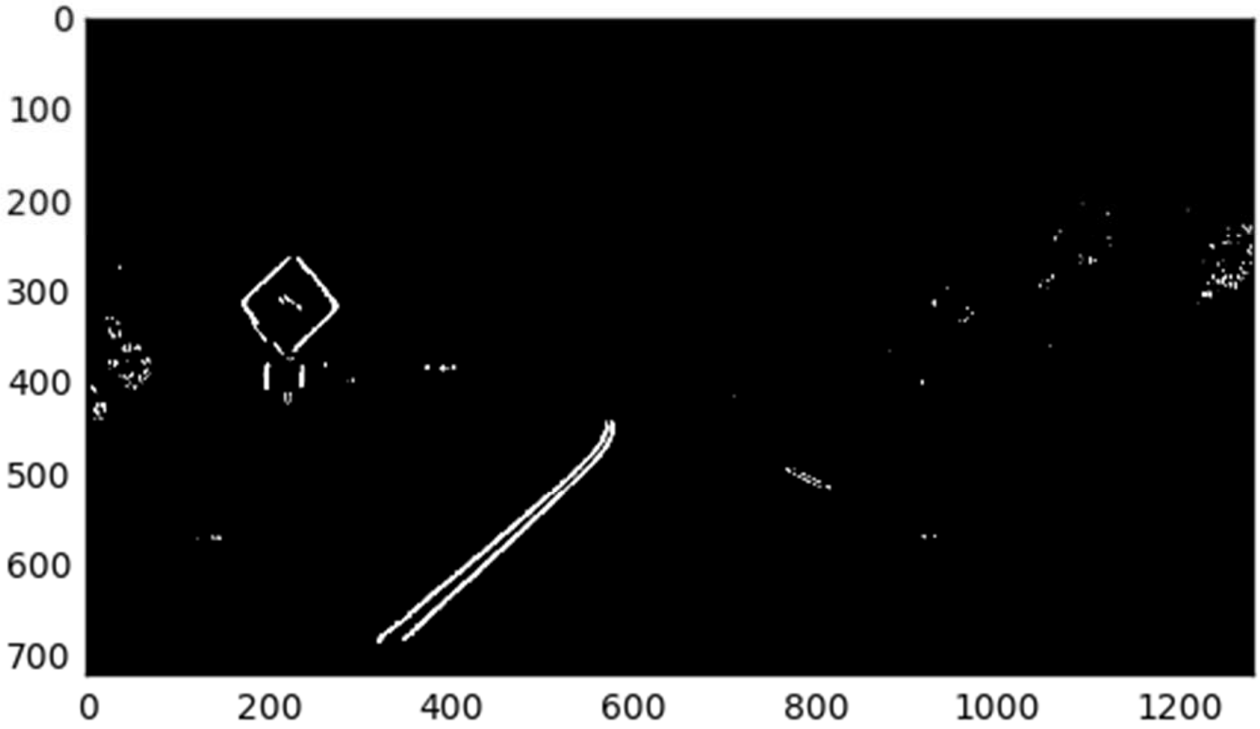


c. Gradient Thresholding:

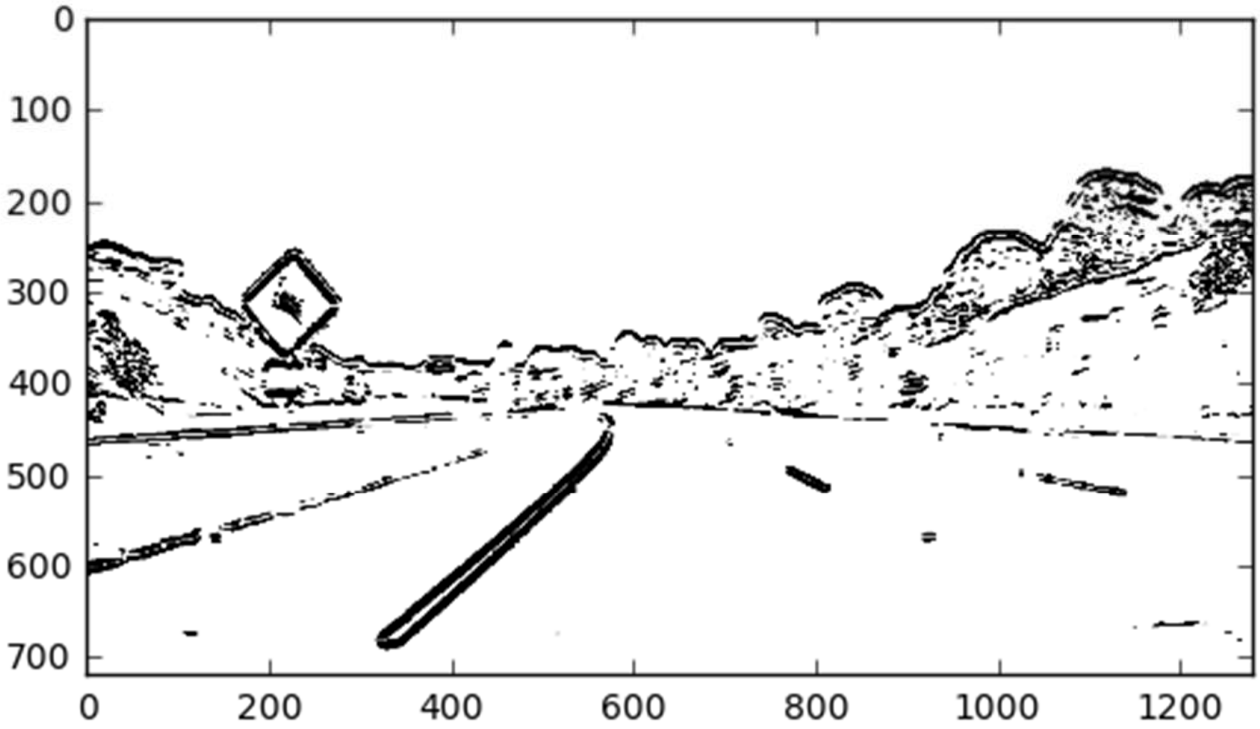
Phương thức **Sobel** được áp dụng để lấy **gradient** theo hướng x và y cũng được sử dụng để có được cường độ và hướng của các ngưỡng ảnh . Để giải thích các ngưỡng này tôi sử dụng hình ảnh thử nghiệm dưới đây và áp dụng các hoạt động ngưỡng 4.



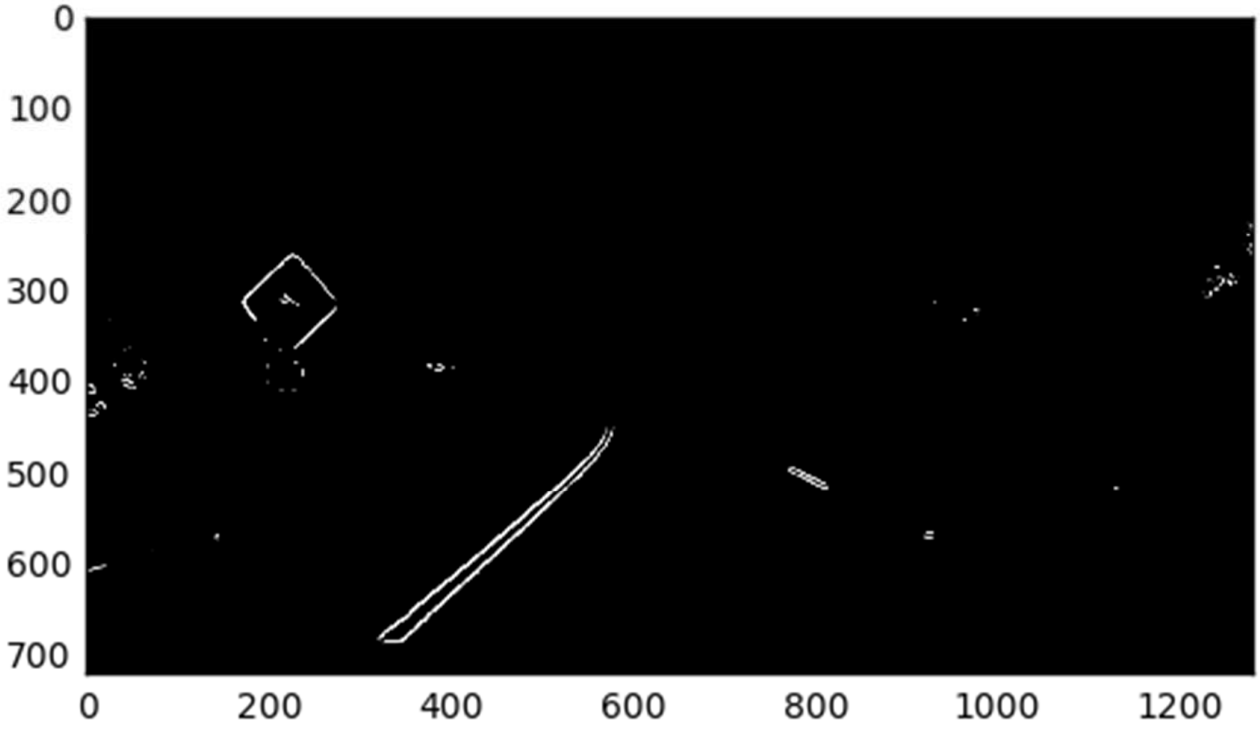
- **Gradient thresholded tại hướng x** sử dụng Sobel Thresholding được gọi trong hàm **abs_sobel_thresh()** .



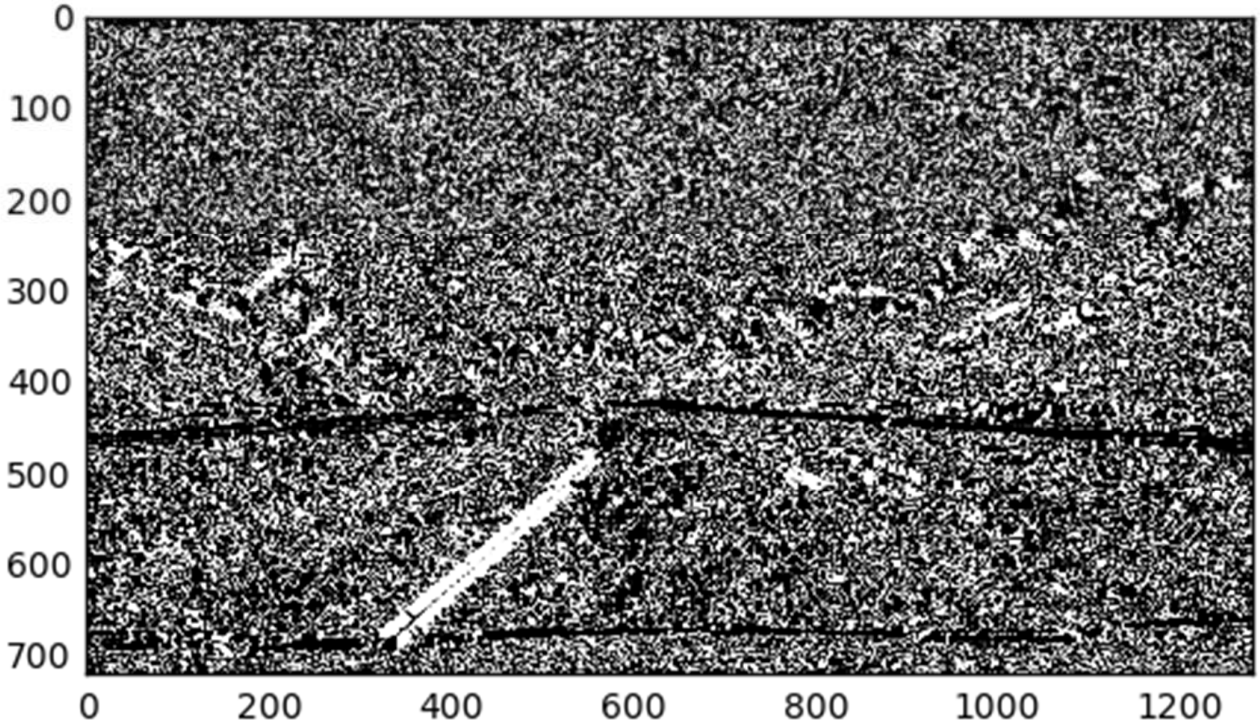
- **Gradient thresholded tại hướng y** sử dụng Sobel thresholding được gọi trong hàm **abs_sobel_thresh()**.



- **Ngưỡng biên độ của Gradient thresholding** được gọi trong hàm **mag_thresh().**



- **Ngưỡng hướng của Gradient thresholding** được gọi trong hàm **dir_threshold().**



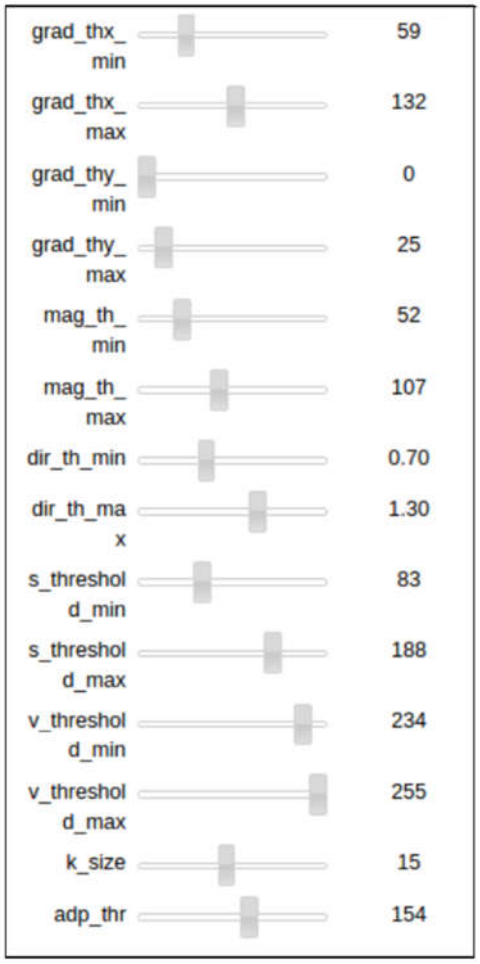
d. Tìm vùng quang trọng/cần thiết:

Để tìm vùng cần thiết cho việc tìm làn nhận dạng làn đường chúng ta sử dụng hàm **region_of_interest()** được thực hiên trong file **perspective_regionofint_main.py**.

Hình ảnh phía dưới là ảnh sau khi đã loại bỏ những vùng ảnh không cần thiết.

e. Kết hợp bước thresholding ở trên để có được hình ảnh nhị phân tốt nhất để phát hiện làn xe.

Để có được làn đường rõ ràng trong hình ảnh nhị phân, các tham số ngưỡng cho các công đoạn trên phải được tinh chỉnh. Đây là phần quan trọng nhất vì những làn đường nhìn thấy rõ ràng dễ phát hiện và phù hợp với đa thức trong các bước tiếp theo. Quy trình tinh chỉnh được thực hiện bằng cách tương tác thay đổi các giá trị ngưỡng và kiểm tra kết quả như được hiển thị bên dưới. Ở đây khu vực cần thiết/có ích cũng được thực hiện để có được hình ảnh nhị phân cuối cùng.

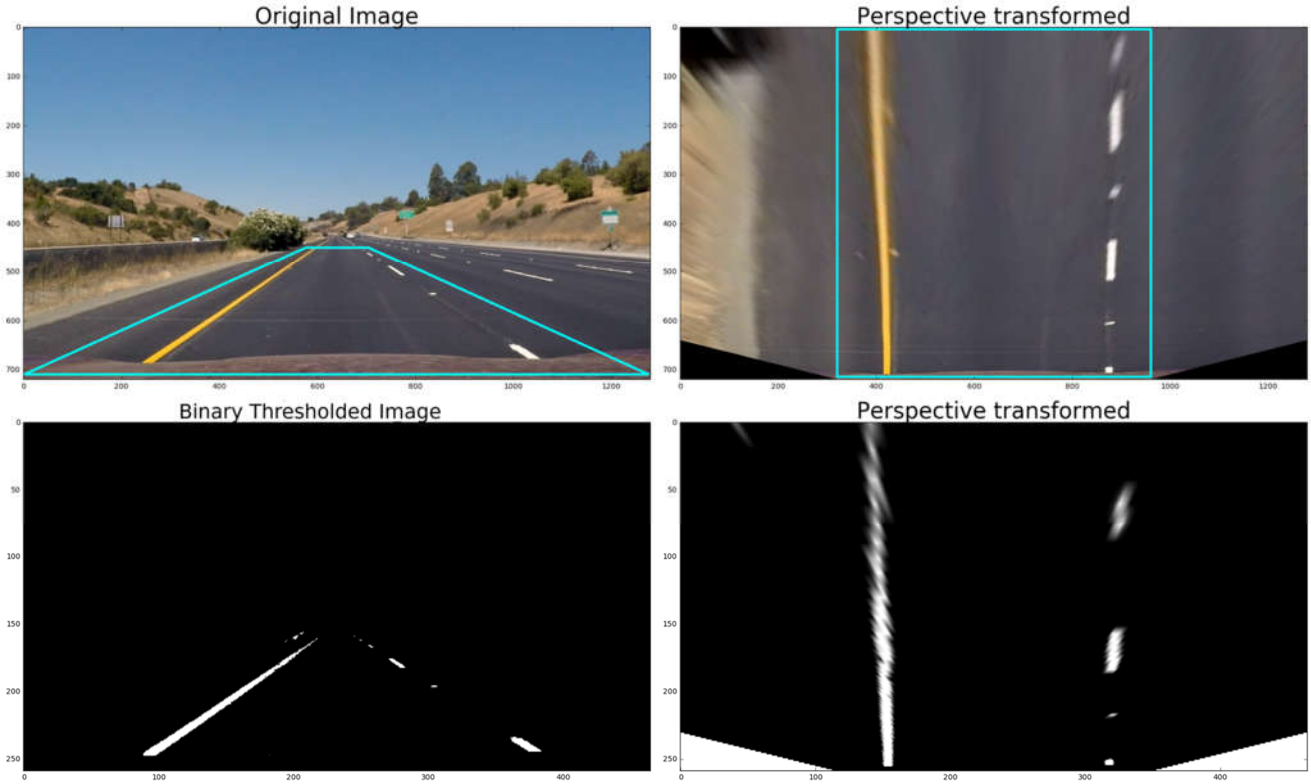


4. Perspective transformation(Phối cảnh chuyển đổi):

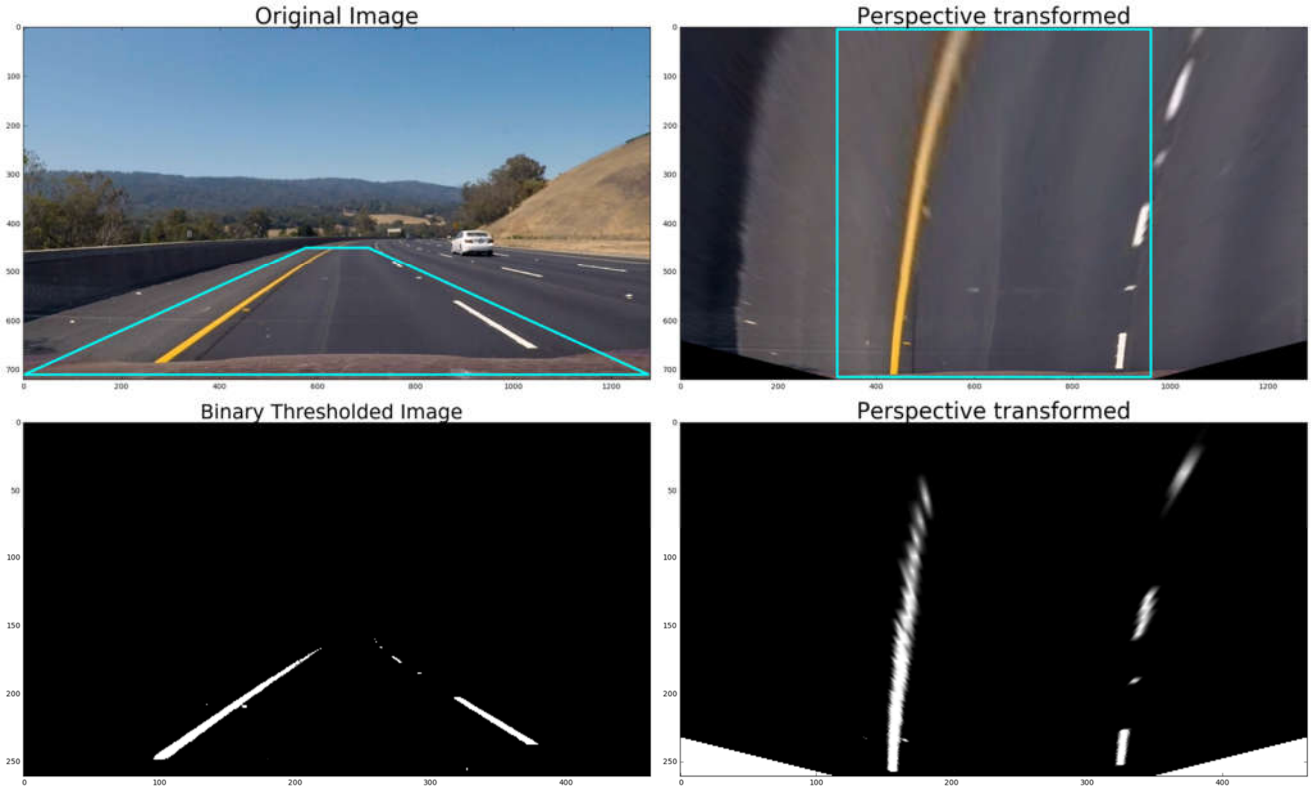
Sau khi hoàn tất các tham số ngưỡng(thresholding), chúng ta tới giai đoạn tiếp theo – Perspective transformation. Bước này có tác dụng biến đổi khung ảnh chụp được với camera và kéo giãn ra cho các bước sau này dễ dàng nhận dạng làn đường.

Ở đây chúng ta sử dụng hàm **perspective_transform()** nằm trong file **perspective_regionofint_main.py**.

- *Ảnh 1 - Làn đường song song*

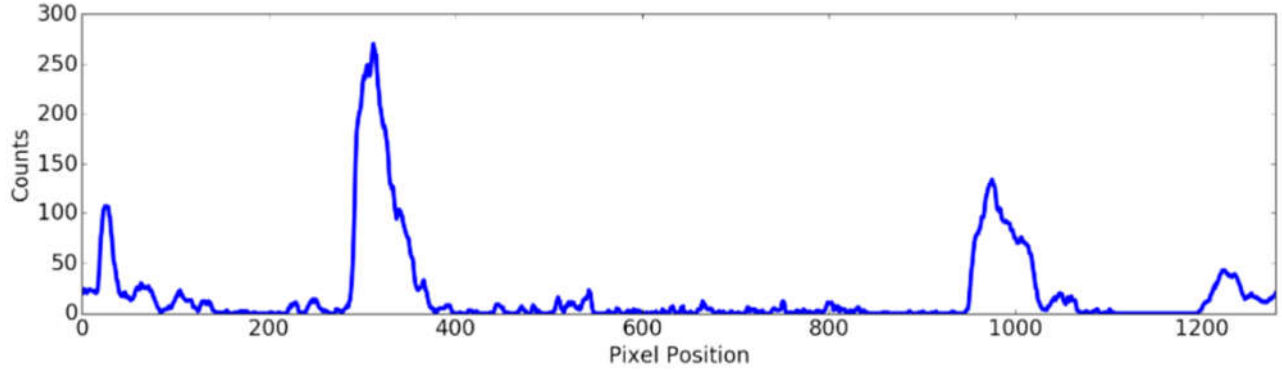


- *Ảnh 2 - Làn đường cong, ở đây làn xe xuất hiện song song với góc quan sát bình thường, nhưng khi chuyển đổi chúng ta có thể thấy rõ là các làn đường cong.*

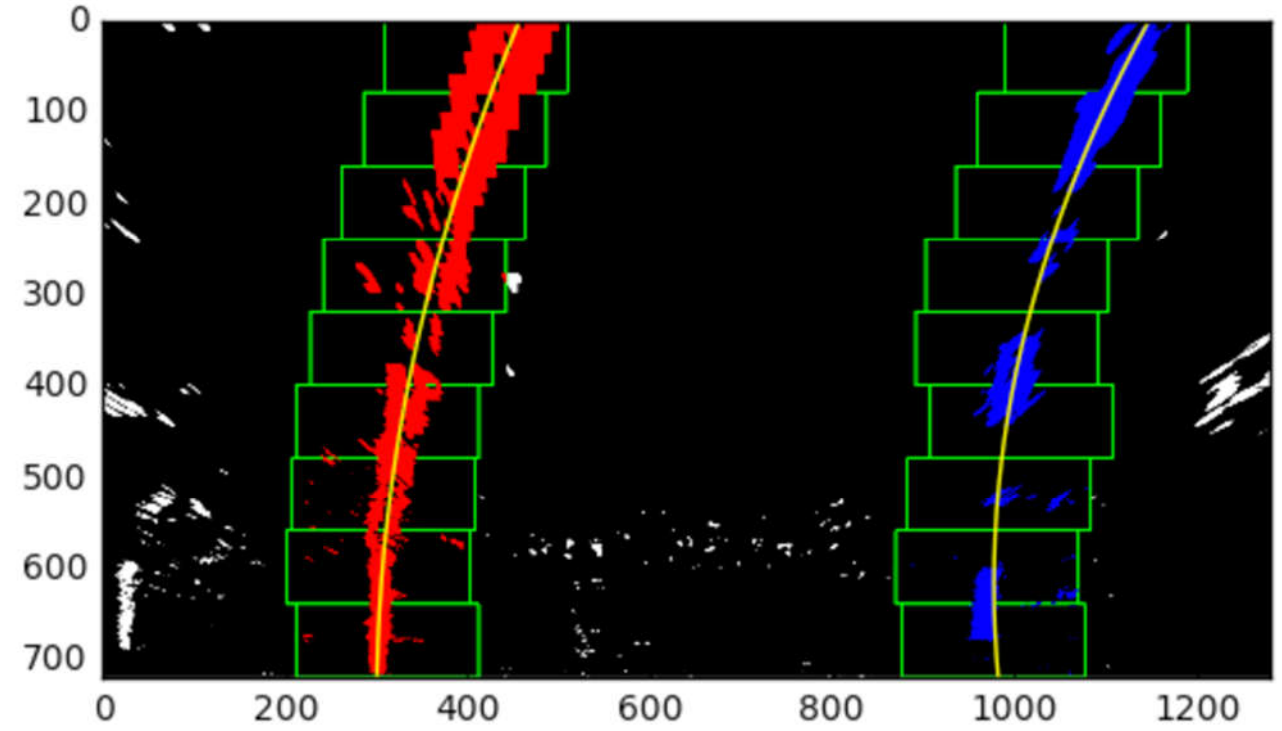


5. Phát hiện các pixels của làn đường và làm khớp để tìm biên làn xe.

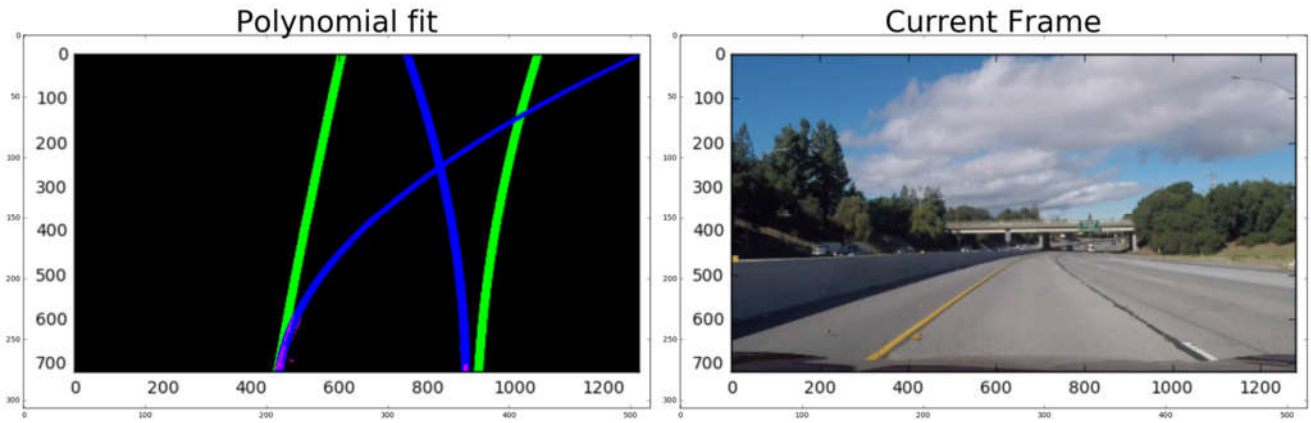
Sau khi áp dụng các kỹ thuật tiền xử lý ảnh, ngưỡng và phối cảnh chuyển đổi cho ảnh làn đường, chúng ta có một hình ảnh nhị phân với các đường nét nổi bật như đã trình bày ở trên. Tiếp theo, chúng ta tìm một đường cong đa giác khớp với các làn xe. Điều này được định nghĩa trong hàm **for_sliding_window()** trong tệp tin **sliding_main.py**.
Đầu tiên, chúng ta có một histogram cùng tất cả các cột ở nửa dưới của hình ảnh. Biểu đồ histogram được hiển thị bên dưới.



Với histogram này, chúng ta thêm các giá trị pixel dọc theo mỗi cột trong ảnh. Trong ảnh nhị phân ngưỡng (thresholded binary image) của chúng ta, giá trị mỗi điểm ảnh là 0 hoặc 1, vì vậy hai đỉnh nổi bật nhất trong biểu đồ này sẽ là các chỉ số rõ rang về vị trí x của đường cơ sở của các làn đường. Chúng ta sử dụng nó như là một điểm khởi đầu để tìm kiếm các đường giới hạn làn đường. Từ đó, chúng ta sử dụng một cửa sổ trượt, đặt xung quanh các trung tâm đường giới hạn làn đường, để tìm và theo các đường lên trên cùng của khung. Kỹ thuật trượt cửa sổ có thể được hiển thị như trong hình dưới đây:



Trong hình trên, các cửa sổ trượt được hiển thị màu xanh lá cây, làn đường bên trái có màu đỏ, đường phải là màu xanh và phù hợp với đa thức là các đường màu vàng.
Đường liên hợp này khi áp dụng cho các khung hình video cho rất nhiều biến động giữa các khung. Chúng ta phải thực hiện làm mịn / trung bình trên 10 khung hình trước đó để có được làn đường được nhận dạng mà không có biến động.



Ở đây các đường màu xanh lá cây là sự phù hợp đa thức trong 10 khung hình trước và màu xanh dương biểu thị sự phù hợp đa thức cho khung hiện tại. Các điểm ảnh làn đường có màu hồng. Có thể nhận thấy rằng các làn đường bên trái và bên phải vượt qua nhau không phù hợp với một môi trường thực tế. Do đó, một cách tốt hơn ở đây là xem xét đa thức trung bình của các khung ảnh trong quá khứ trong những trường hợp này.

6. Xác định độ cong của làn đường và vị trí của xe đối với trung tâm.

Độ cong của làn đường f (y) được tính bằng cách sử dụng công thức R (đường cong)

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

$$f'(y) = \frac{dx}{dy} = 2Ay + B$$

$$f''(y) = \frac{d^2x}{dy^2} = 2A$$

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

Vị trí của xe được tính như sự khác biệt giữa trung tâm hình ảnh và trung tâm làn xe. Nó được thể hiện trong hình dưới đây như là độ lệch từ trung tâm.

Code tham khảo: https://github.com/sujaybabruwad/Advanced-Lane-Detection

7. Xác định góc lái

Dựa vào độ lệch giữa tâm hình và tâm làn đường, chúng ta khảo sát để tìm ra góc lệch của ô tô so với làn đường. Ta lấy góc lệch này nhân với hệ số đánh lái (phụ thuộc vào tốc độ của ô tô) để điều chỉnh góc lái giúp cho xe đi theo đường trung tâm của làn đường.