ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

MÔN HỌC: NHẬP MÔN THỊ GIÁC MÁY TÍNH

Đề tài:

Phát hiện làn đường với video giao thông

Lớp: CS231.M11..KHCL

Giảng viên hướng dẫn: TS. Nguyễn Vinh Tiệp

Sinh viên thực hiện:

Phan Nguyễn Thành Nhân	19521943
Phạm Minh Long	19521797
Tạ Huỳnh Đức Huy	19521634
Nguyễn Trần Phước Lộc	19521764

Thành phố Hồ Chí Minh, ngày 07 tháng 12 năm 2021

MỤC LỤC

I. Tổng quan bài toán

1. Giới thiệu

Cùng với sự phát triển tự động hóa, phương tiện đi lại trở nên tất yếu đặc biệt là ô tô. Chính vì vậy, việc hình thành và phát triển xe tự lái như một xu hướng tất yếu, đã được các công ty lớn như Tesla, Google và gần gũi nhất là VinAI. Một trong những yếu tố quan trọng và chủ chốt quyết định sự thành công là kỹ thuật **Phát hiện làn đường - Lane detection.**

Phát hiện làn đường với video giao thông - Là một phiên bản của bài toán Object Detection. Cải tiến ở mức độ cao hơn khi được xử lý trên một video hay chuỗi hình ảnh liên tiếp nhau dựa vào vị trí và chuyển động trong quá khứ giữa các khung hình chủ yếu dựa trên kỹ thuật xử lý ảnh Computer Vision và Machine Learning được vận dụng trong các hệ thống lùi xe, cảnh báo va chạm…

2. Thử thách và mục tiêu đồ án

2.1. Mục tiêu

Tìm hiểu về hướng tiếp cận và phương pháp xử lý bài toán "Phát hiện làn đường bằng video giao thông" từ đó xây dựng một ứng dụng để thử nghiệm kết quả theo hướng xử lý ảnh.

2.2. Thử thách

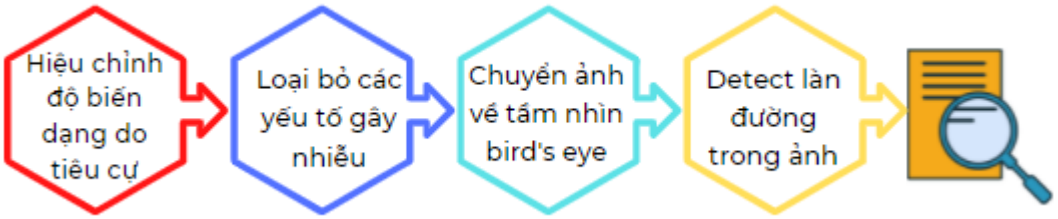
Một trong những thách thức khó khăn lớn nhất của bài toán này là mỗi camera lại cho mỗi thông số khác nhau như góc quay, tiêu cự,.. điều đó dẫn tới việc xử lý đầu vào gặp rất nhiều khó khăn. Đồng thời, việc xử lý ảnh để chọn được tham số thích hợp cũng phụ thuộc nhiều vào các điều kiện môi trường khác nhau.

Bên cạnh đó, do bị ảnh hưởng bởi dịch bệnh Covid-19 nên đã gây một chút khó khăn cho nhóm trong việc thu thập dữ liệu nên bộ dữ liệu ở đường phố Việt Nam hiện tại rất ít, do đó nhóm phải tận dụng thêm một số bộ dữ liệu thu thập trên mạng để so sánh kết quả.

II. Cách tiếp cận bài toán

1. Hướng tiếp cận

Đối với bài toán phát hiện làn đường có nhiều phương án tiếp cận khác nhau bao gồm từ phức tạp như sử dụng các phương pháp học sâu (deep learning) cho đến cơ bản như ứng dụng các kỹ thuật xử lý trên ảnh. Trong báo cáo này, nhóm xây dựng một mô hình nhận diện làn đường bằng những tính toán trên điểm ảnh thông qua thư viện hỗ trợ sẵn trên Python là OpenCV. Hướng tiếp cận các kỹ thuật sử dụng theo trình tự sau:



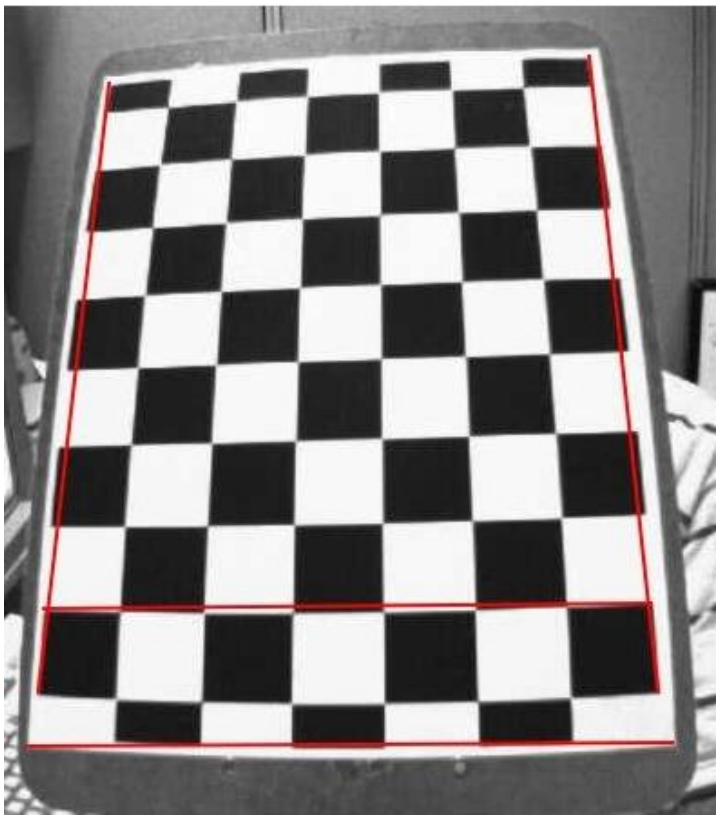
Hình 1: Trình tự các kỹ thuật xử lý

Đầu vào của bài toán là một đoạn video được xử lý lần lượt qua các bước chỉnh sửa biến dạng ảnh của camera, loại bỏ các yếu tố gây nhiễu bằng bộ lọc đưa ảnh về dạng nhị phân, chuyển ảnh về góc nhìn từ trên cao xuống, phát hiện các pixel trên vạch kẻ đường bằng đồ thị histogram và phương pháp sliding window. Sau cùng là tìm ra phương trình đường thẳng chứa làn đường và tô màu làn đường.

2. Xử lý chi tiết

2.1. Chỉnh sửa biến dạng của tấm ảnh (méo ảnh)

Trong điều kiện thực tế, với mỗi góc camera sẽ cho ra những những tấm ảnh khác nhau, có thể dẫn đến việc bị sai lệch và biến dạng hình ảnh, đặc biệt là với những loại camera góc rộng gây khó khăn trong việc xử lý. Nguyên nhân chủ yếu từ độ phóng đại của tiêu cự ảnh làm cho ánh sáng bị bẻ cong dẫn đến những đường kẻ không thẳng hàng như thực tế. Như *Hình 2* bên dưới, các đường cạnh của ô bàn cờ bị bóp méo, trở nên phình to ra, không trùng khớp với đường thẳng màu đỏ.



Hình 2: Hình ảnh bàn cờ bị bóp méo do độ sai lệch của camera

Những biến đổi do máy ảnh tạo nên có thể khó nhận ra bằng mắt thường, tuy nhiên điều này có thể ảnh hưởng đến các phép tính toán bị sai lệch trong thị giác máy tính. Mặc dù vậy, hiện nay đã có các phương pháp hỗ trợ tính toán giúp khôi phục những sai lệch này.

Để chỉnh sửa sự biến dạng của ảnh, nhóm sử dụng nhiều hình ảnh bàn cờ vua Chessboard trên mặt phẳng và được chụp bằng chính máy ảnh mà nhóm dùng để quay video làn đường. Mục đích của việc này là tạo ra ma trận hệ số của máy ảnh và hệ số biến dạng, cụ thể:

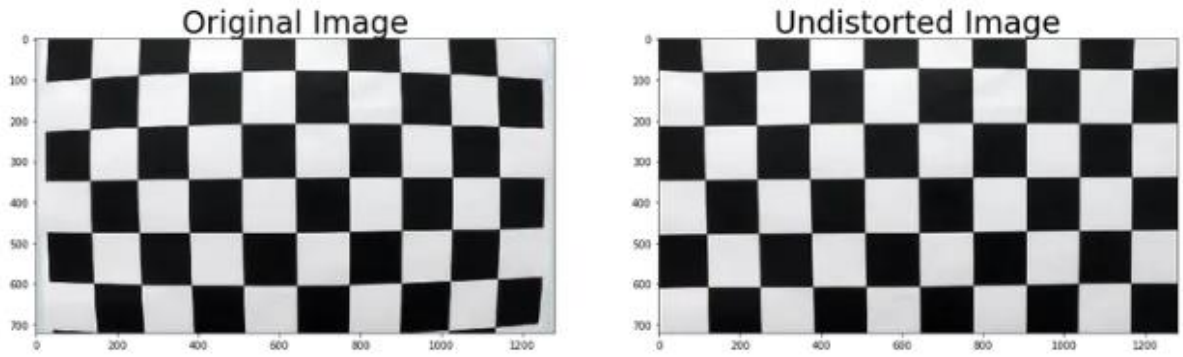
$$camera\ matrix = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$Distortion\ coefficients = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$$

Hình 3: Ma trận hệ số của máy ảnh và ma trận hệ số biến dạng

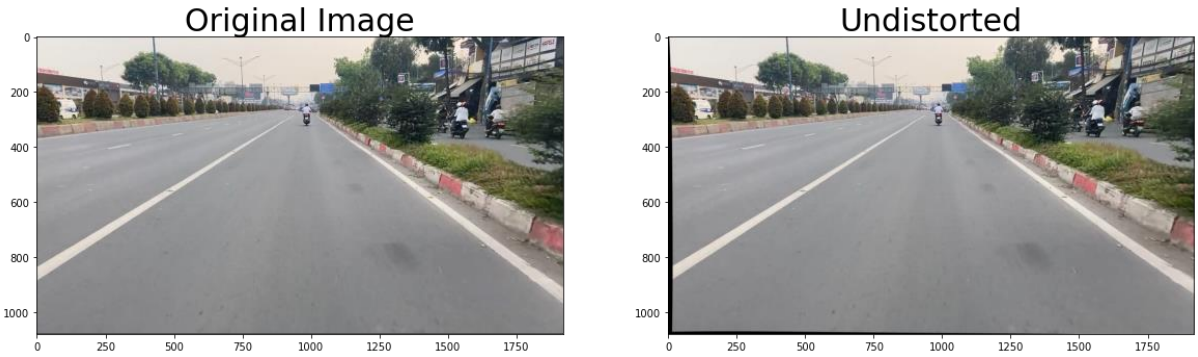
Trong đó, f là tiêu cự và c là tâm quang học. Còn lại là các hệ số biến dạng được dùng để biến đổi ảnh về bình thường. Các thông số trên sẽ được tự động tìm bằng phương pháp tính toán thông qua ảnh chụp các ô bàn cờ.

Các hình bàn cờ vua được chụp trước tiên sẽ được chuyển về dạng grayscale. Tiếp đến là xác định các góc trong bàn cờ bằng hàm cv2.findChessboardCorners. Vì bàn cờ là một vật thể hai chiều nên chúng ta có thể dựa trên các điểm thẳng hàng để phát hiện và căn chỉnh ảnh. Sau đó, các tập tọa độ các giao điểm trong bàn cờ sẽ được đưa vào hàm cv2.CalibrateCamera để tìm ra ma trận máy ảnh và hệ số biến dạng đã nêu trên. Khi đã có các hệ số trên, các ảnh đầu vào từ camera sẽ được biến đổi về bình thường thông qua hàm có sẵn cv2.undistort(). Sự thay đổi có thể nhận biết thông qua *Hình 4* bên dưới.



Hình 4: Hình ảnh mô phỏng sự thay đổi của bàn cờ vua sau khi hiệu chỉnh.

Việc áp dụng trong thực tế mắt thường khó nhận ra hơn so với bàn cờ vua tuy nhiên có ảnh hưởng đến các bước xử lý tiếp theo.



Hình 5: Hình ảnh mô phỏng sự thay đổi của đầu vào sau khi hiệu chỉnh.

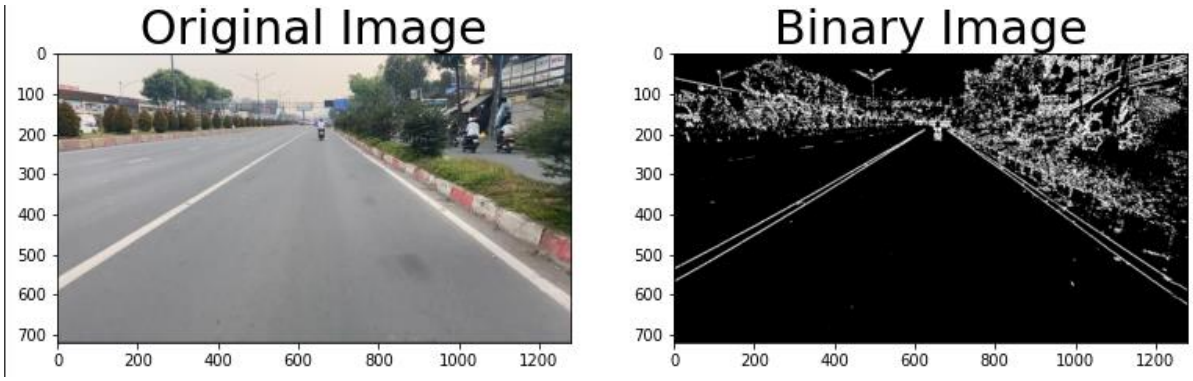
2.2. Áp dụng bộ lọc để tách biệt các làn đường với các vật thể khác trong tấm ảnh

Ý tưởng ban đầu được áp dụng là lọc ra làn đường bằng các màu sắc, cụ thể là màu trắng. Thuật toán có thể hoạt động tốt trên môi trường cố định hoàn hảo. Tuy nhiên, đây không phải là ý tưởng tốt. Vì đôi khi làn đường được tô màu khác như màu vàng thay vì màu trắng, hay đôi lúc trên làn đường sẽ xuất hiện bóng cây hoặc bóng của các phương tiện khác làm ảnh hưởng đến kết quả.

Thay vào đó, phương pháp phát hiện ra làn đường dựa vào sự tương phản màu giữa làn đường và các vật thể khác cho ra kết quả tối ưu hơn. Vì làn đường có xu hướng màu nổi hơn ví dụ như màu trắng hoặc vàng so với màu đen hoặc

xám của bê tông đường. Lợi dụng đặc điểm đó làm cho việc phát hiện làn đường được tốt hơn ngay cả khi làn đường bị đổ bóng.

Về chi tiết cách xử lý, đầu tiên bức ảnh sau khi được chỉnh sửa biến dạng sẽ được chuyển sang chiều không gian HLS, trong không gian màu HLS chúng ta sẽ sử dụng 2 kênh màu chính là *Saturation (S)* và *Lightness (L)* để tính toán. Nhận thấy rằng làn đường có xu hướng tương phản theo chiều dọc, nên nhóm đã sử dụng kỹ thuật đạo hàm Sobel bằng hàm cv2.Sobel để tính đạo hàm ảnh theo chiều ngang, mục đích ở đây là để đo sự tương của làn đường với mặt đường theo chiều ngang. Các điểm ảnh nổi bật hơn ngưỡng cho trước sẽ được gán bằng 1 và được thêm vào ảnh nhị phân.



Hình 6: Hình ảnh mô phỏng kết quả của đầu vào khi áp dụng bộ lọc.

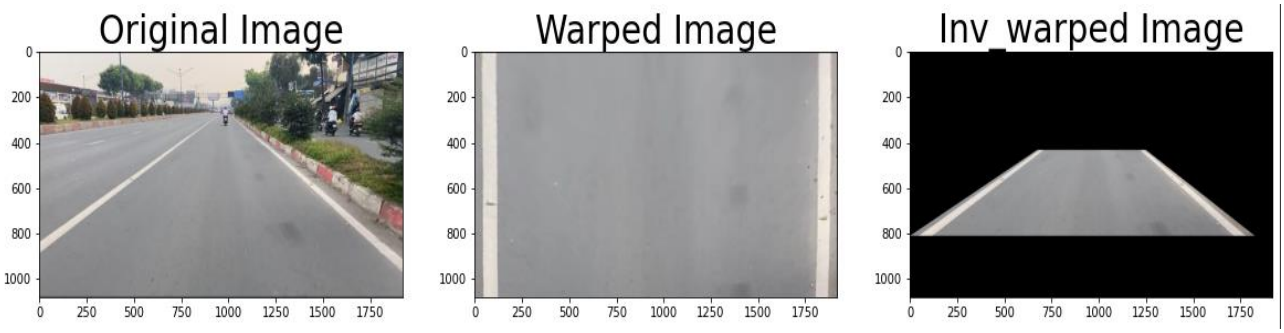
2.3. Chuyển tấm ảnh về tầm nhìn bird's eye

Hình 7: Hình ảnh mô phỏng sự hội tụ về phía chân trời của làn đường

Với góc nhìn chính diện, ở phía cuối bức ảnh các đường thẳng sát nhau như dần hội tụ về một điểm như *Hình 7*. Điều này dẫn đến việc phát hiện làn đường trở nên khó khăn hơn. Giải pháp là chuyển góc nhìn trực diện thành góc nhìn từ trên cao xuống (Bird's eye) bằng thuật toán biến đổi phối cảnh (Perspective Transform) đồng thời loại bỏ các thành phần khác không cần thiết trong bức hình. Điều này giúp dễ dàng trong việc xác định phương trình đối với làn đường và tính được độ cong. Các bước xử lý bao gồm 2 bước: Xác định vùng không gian cần quan tâm và Sử dụng Thuật toán biến đổi phối cảnh.

Xác định vùng không gian cần quan tâm

Để tăng hiệu quả của thuật toán, nhóm quyết định xử lý trên một khu vực cụ thể bằng cách chọn bốn điểm trên hình ảnh ban đầu không bị sai lệch và chuyển đổi phối cảnh như hình bên dưới.



Hình 8: Hình ảnh mô phỏng kết quả của ảnh đầu vào khi áp dụng phương pháp Biến đổi phối cảnh.

Thuật toán Biến đổi phối cảnh (Perspective Transform)

Ma trận phép biến đổi:

$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = M . \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

Hình 9: Ma trận biến đổi

Với M là ma trận 3x3, sử dụng để biến đổi ảnh.

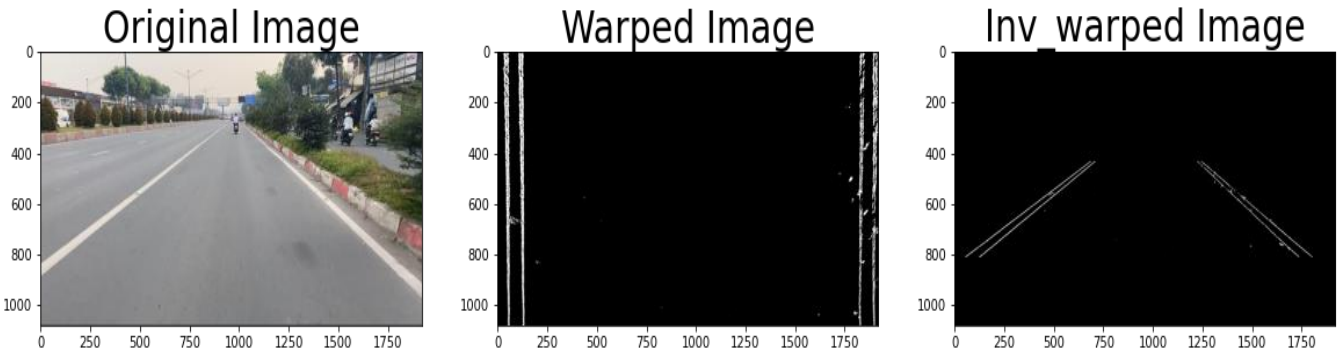
$$dst(x,y) = src\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}\right)$$

Hình 10: Công thức tính Ma trận biến đổi

Trong đó:

- dst (x, y) là tọa độ của các pixel trong hình ảnh được biến đổi
- src (x, y) là tọa độ pixel trong hình ảnh gốc

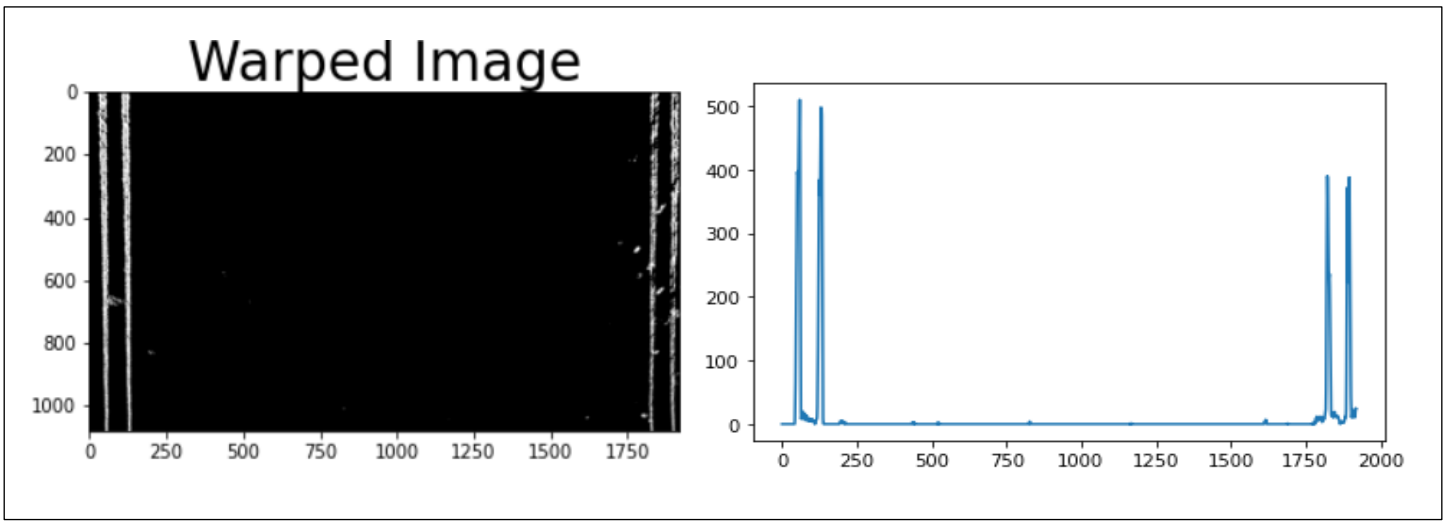
Dựa vào dst(x,y) và src(x,y) chúng ta có thể tính được ma trận biến đổi hàm cv2.getPerspectiveTransform. Sau đó chúng ta áp dụng hàm cv2.warpPerspective với ma trận biến đổi trên ảnh nhị phân thì chúng ta sẽ được tấm ảnh *Warped Image* bên dưới.



Hình 11: Hình ảnh mô phỏng kết quả của ảnh đầu vào khi áp dụng phương pháp Biến đổi phối cảnh và bộ lọc.

2.4. Phát hiện các làn đường trong tấm ảnh

Sử dụng đồ thị Histogram với ảnh đầu vào đã qua các bước tiền xử lý ở trên, ta tính toán biểu đồ của các điểm nhị phân theo trục thẳng đứng, từ đó xác định được những vị trí x tương ứng có cường độ pixel cao nhất.



Hình 12: Hình ảnh mô phỏng đồ thị Histogram của ảnh đầu vào đã được tiền xử lý

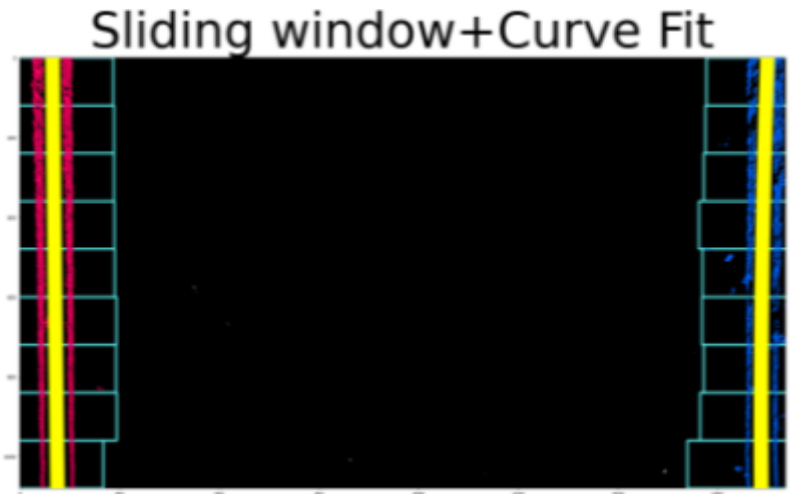
Từ biểu đồ cường độ pixel trên, ta chọn tọa độ điểm x tương ứng tại cái đỉnh để thực hiện thuật toán *Sliding Window Search* để tìm kiếm những trường hợp có khả năng cao là làn đường, sau đó lấy tọa độ x,y tương ứng để tìm ra một đa thức bậc hai, phù hợp nhất với hai vạch kẻ đường.

Kỹ thuật *Sliding Window Search* dùng để phát hiện các điểm ảnh bắt đầu của hai làn đường khác nhau. Khi có được điểm bắt đầu từ đồ thị Histogram, các khung trượt sẽ đếm các điểm ảnh có trong khung ảnh. Nếu số lượng điểm ảnh đủ ngưỡng cho trước khung trượt sẽ chồng lên tại vị trí trung bình cộng các điểm ảnh. Nếu không có điểm ảnh nào được phát hiện quanh khung trượt, khung trượt sẽ chồng lên khung liền trước. Điều này tiếp tục cho đến hết khung hình. Sau khi trượt qua hết các ô sẽ trả về lần lượt các vector lưu giá trị của hai điểm theo hướng x,y như *Hình 13*.

```
array([269, 270, 271, ..., 448, 449, 450], dtype=int64), // leftx
array([640, 640, 640, ..., 79, 79, 79], dtype=int64), // lefty
array([1008, 1009,1010,..., 1227, 1228, 1229],dtype=int64),//rightx
array([640, 640, 640, ..., 79, 79, 79], dtype=int64) //righty
```

Hình 13: Tọa độ các điểm x,y của 2 bên làn đường trái phải

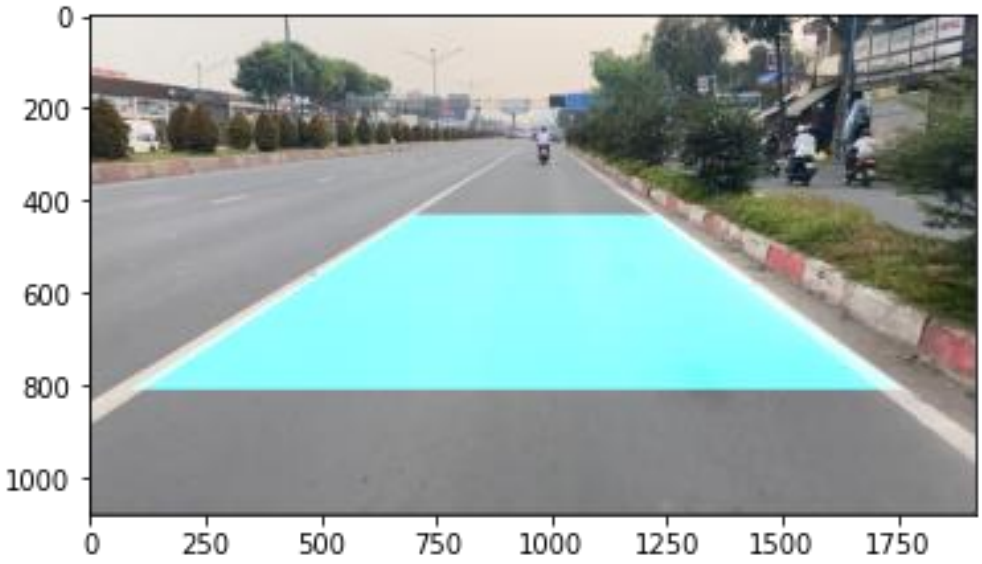
Từ các cặp giá trị tọa độ x,y đã được tính toán. Sử dụng hàm Polyfit để tìm ra hệ số tương ứng với hai vạch kẻ đường. Với x y lần lượt là các tọa độ, left right là hai vạch kẻ đường bên trái và bên phải.



Hình 14: Hình ảnh mô phỏng vạch kẻ đường được vẽ từ phương trình đường cong của làn đường

2.5. Vẽ làn đường

Sau khi có được hai phương trình đường cong tương ứng với làn đường, nhóm tiến hành vẽ làn đường lên ảnh đầu vào bằng cách tô màu làm nổi bật làn đường giữa hai phương trình đường cong như *Hình 15*.



Hình 15: Hình ảnh mô phỏng làn đường được vẽ lên ảnh đầu vào

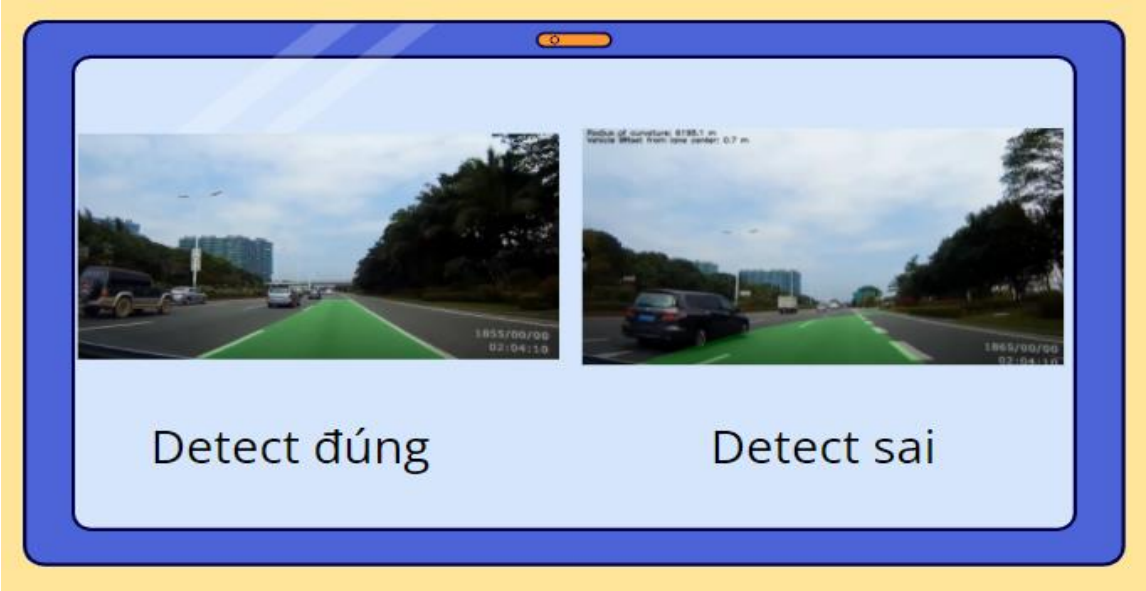
III. Kết quả thực nghiệm

Để đánh giá kết quả một cách trực quan, nhóm tiến hành thử nghiệm trên nhiều bộ dữ liệu khác nhau bao gồm *bối cảnh nước ngoài* và *dữ liệu do nhóm tự quay ở Việt Nam*.

Với bộ dữ liệu đã được thu thập ở nước ngoài, làn đường cao tốc thông thoáng, ít bị nhiễu bởi điều kiện tự nhiên, đồng thời do có sẵn thông số của camera để hiệu chỉnh độ sai lệch nên kết quả trên bộ dữ liệu này khá tốt.

Hình 16: Kết quả demo với bộ dữ liệu được thu thập sẵn ở nước ngoài

Với một bộ dữ liệu khác nhóm thu thập từ Youtube. Việc không có các thông số của camera để hiệu chỉnh dẫn đến độ chính xác trong việc detect chưa được cao, có những vị trí còn sai.



Hình 17: Kết quả nhận diện làn đường với dữ liệu thu thập trên youtube

Với bộ dữ liệu do nhóm tự thu thập ở Việt Nam di chuyển trên đường bằng xe máy, với góc camera điện thoại cầm tay và thử nghiệm tại TP Hồ Chí Minh. Trái ngược với giao thông ở nước ngoài, ở Việt Nam mật độ xe dày đặc nhiều yếu tố nhiễu làm ảnh hưởng rất nhiều trong quá trình thực nghiệm.

Hình 18: Kết quả nhận diện làn đường với dữ liệu thu thập ở Việt Nam

Cụ thể với bộ dữ liệu này, việc nhận dạng gặp khó khăn khi có các yếu tố gây nhiễu như vạch kẻ ở giữa làn đường, xe máy lấn làn… dẫn đến kết quả chưa thật sự tốt.

IV. Kết Luận

1. Nhận xét

Phân loại làn đường là một kỹ thuật phức tạp và bao gồm rất nhiều yếu tố ảnh hưởng (ánh sáng, môi trường, thời tiết), vì vậy chỉ dựa trên thư viện OpenCV để xử lý gặp nhiều khó khăn và thời gian xử lý còn khá lâu (Khoảng 15p cho một video 30s đầu vào)

Thuật toán phân loại làn đường đã có thể hoạt động trên các bộ dữ liệu được kiểm tra, tuy nhiên hiệu năng và độ chính xác vẫn chưa thật sự tốt. Đặc biệt, khi áp dụng thực tế vào điều kiện cụ thể ở Việt Nam (đường xấu, mật độ xe cao, các vạch kẻ trên đường…) cho kết quả khá tệ và cần được cải thiện thêm. Thông qua đồ án, chúng em đã có góc nhìn và những kiến thức liên quan đến xử lý ảnh và phát hiện làn đường căn bản từ đó áp dụng để xây dựng một ứng dụng thực tế.

2. Hướng phát triển

Với các kỹ thuật xử lý ảnh, việc phải tính toán lại các phương trình đường cong của làn đường đôi khi mất thời gian và không cần thiết. Làn đường chỉ thay đổi đột ngột khi vào các khúc cua hoặc đi qua các ngã rẽ. Chính vì vậy, sắp tới nhóm dự định sẽ xây dựng bổ sung thêm kỹ thuật nhỏ lưu lại các làn đường đã được phát hiện ở khung ảnh liền trước và chỉ cập nhật lại khi số lượng điểm ảnh của làn đường thay đổi nhiều hơn ngưỡng định trước nhằm giảm thời gian chạy của thuật toán.

Bên cạnh đó, với sự phát triển mạnh của các kỹ thuật về học sâu, việc phát hiện làn đường trở nên dễ dàng hơn, nhanh chóng hơn. Cụ thể như các kỹ thuật về mạng Neuron, các kỹ thuật về Segment cũng sẽ được nhóm tìm hiểu và triển

khai. Lợi thế lớn nhất của các kỹ thuật về học sâu hiện nay là có thể nhận điện được làn đường ngay cả trong nhiều thời tiết và môi trường khác nhau, kể cả việc làn đường bị đổ bóng. Cùng với đó, phương pháp học sâu còn có khả năng xử lý trên thời gian thực, một điểm rất quan trọng trong các mô hình xe tự hành ngày này. Các ưu điểm này khiến cho các kỹ thuật học sâu được sử dụng phổ biến hơn so với phương pháp xử lý dựa trên màu điểm ảnh truyền thống.

V. Tài liệu thực nghiệm

- Dữ liệu do nhóm tự quay và thu thập: data
- Thực nghiệm trên nhiều video dữ liệu: result
- File source code nhóm tiến hành thử nghiệm: Lane Detection

VI. Tài liệu tham khảo

[1] "OpenCV: Camera Calibration," [Online]. Available: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html. [Accessed 2 December 2021].

[2] kemfic, "Curved Lane Detection," 24 May 2018. [Online]. Available: https://www.hackster.io/kemfic/curved-lane-detection-34f771#toc-perspective-warp-3. [Accessed 2 December 2021].

[3] J.-c. Sung, "Advanced lane detection using computer vision," 8 February 2007. [Online]. Available: https://github.com/georgesung/advanced_lane_detection. [Accessed 2 December 2021].

[4] A. Sears, "The Ultimate Guide to Real-Time Lane Detection Using OpenCV – Automatic Addison," 12 April 2021. [Online]. Available: https://automaticaddison.com/the-ultimate-guide-to-real-time-lane-detection-using-opencv/. [Accessed 2 December 2021].