ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC
XỬ LÝ ẢNH VÀ ỨNG DỤNG

ĐỀ TÀI: PHÁT HIỆN LÀN ĐƯỜNG SỬ DỤNG THUẬT TOÁN HOUGH TRANSFORM

GVHD : Mai Tiến Dũng

Lớp : CS406.M11

SVTH : 18521060 – Trịnh Hưng Long
18521157 – Lê Trần Phúc Nguyên

TPHCM, ngày 3 tháng 12 năm 2021

Mục lục

I. Giới thiệu đề tài:

Bài toán phát hiện làn đường được ứng dụng trong việc phát triển xe tự lái, cũng như xây dựng các phần mềm hỗ trợ cho người lái xe.

- *Input:* Ảnh/ Video làn đường từ camera được gắn trên phương tiện tham gia giao thông. Ảnh input là ảnh làn đường có đủ 2 vạch kẻ(lane line) thu được từ camera trên phương tiện tham gia giao thông. Làn đường cần nằm ở trung tâm của bức ảnh.

- *Output:* Ảnh/Video chứa vị trí làn đường phát hiện được.

II. Các bước thực hiện:

1. Gray sclae:

Chuyển đổi ảnh màu sang kênh màu xám, giúp giảm độ phức tạp khi xử lý ảnh cũng như tăng độ chính xác khi phát hiện đường thẳng.

2. Gaussian Blur:

Phép làm mờ ảnh, hay làm mịn ảnh có nhiều ứng dụng, trong bài toán này phép làm mịn Gaussian Blur giúp giảm nhiễu, bỏ qua các chi tiết nhỏ không mong muốn trước khi thực hiện rút trích các đối tượng liên quan

=> Giúp phát hiện cạnh tốt hơn.

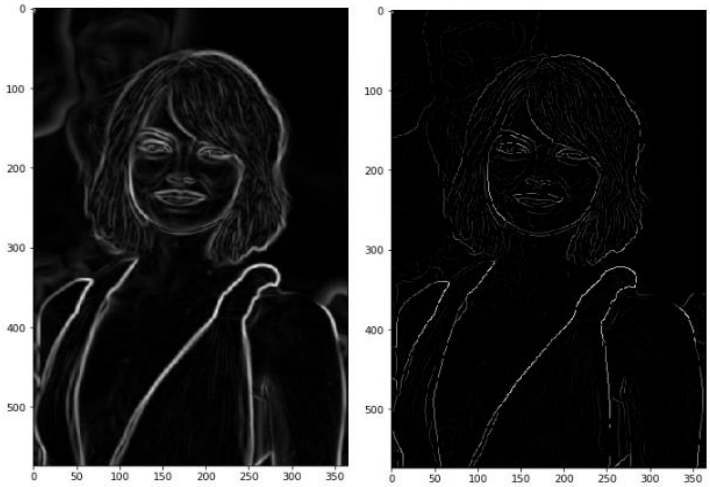
3. Canny Edge Detection:

Tổng quan về Gradient của ảnh:

- Trong xử lý ảnh, gradient ở đây chính là độ dốc về mức sáng. Hay nói cách khác chính là sự thay đổi các giá trị pixel trong ảnh.
- Trong hình ảnh, thường tồn tại các thành phần như: vùng trơn, góc / cạnh và nhiễu.
- Đối với vùng ảnh trơn (smooth) thì các pixel trong vùng ảnh đó có giá trị xấp xỉ bằng nhau => tức là không có sự biến thiên về giá trị, hay gradient của các pixel trong vùng ảnh trơn gần như = 0.
- Còn ở các cạnh thì các pixel trong vùng sẽ có sự biến thiên về giá trị (lên hoặc xuống dốc) => làm tăng giá trị gradient.

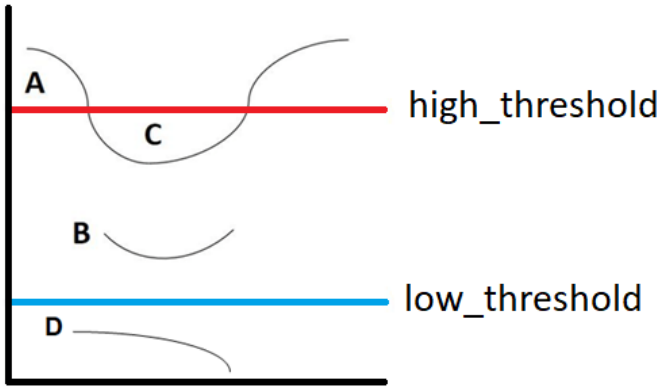
Non-maximum supression:

- Ta thấy các đường biên vẫn còn khá dày do có nhiều pixel cùng miêu tả pixel cạnh của ảnh gốc. Để tìm được pixel cạnh duy nhất => ta cần chọn ra pixel tốt nhất trong số pixel trên bằng bước non-maximum supression.
- Non-maximum supression: so sánh giá trị gradient của pixel với giá trị gradient của 2 pixel lân cận, và chỉ giữ lại pixel có giá trị gradient tốt nhất, đồng thời loại bỏ 2 pixel còn lại.

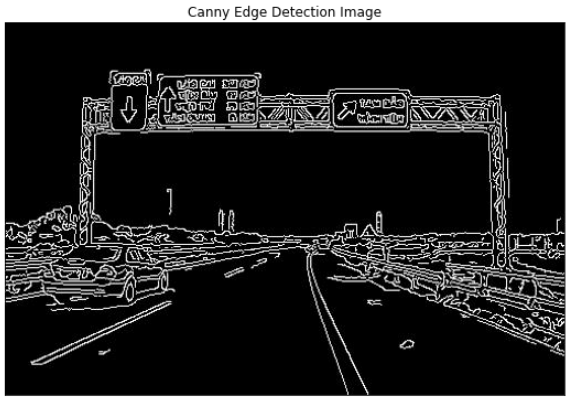


Lọc ngưỡng: Sử dụng 2 giá trị ngưỡng cao(high_threshold) và ngưỡng thấp(low_threshold) để lọc :

- Nếu giá trị gradient của pixel > high_threshold => chắc chắn là cạnh.
- Nếu giá trị gradient của pixel < low_threshold => không phải cạnh => loại bỏ.
- Nếu giá trị gradient của pixel trong khoảng giữa high_threshold và low_threshold thì ta xem xét liên kết của pixel với pixel liền kề. Nếu pixel liền kề là cạnh => chắc chắn là cạnh. Ngược lại => loại bỏ

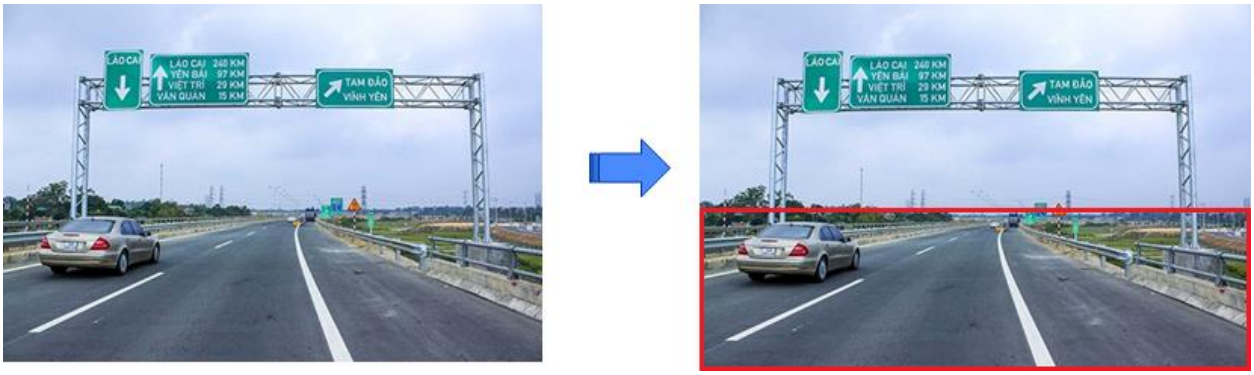


Kết quả thu được sau khi thực hiện Canny Edge detection:



4. Region of interest selection:

Đây là bước xác định vùng quan tâm trong ảnh: Vì đây là bài toán phát hiện làn đường, thông thường những hình ảnh trích xuất từ camera gắn trên phương tiện tham gia giao thông(như xe máy, ô tô) sẽ có phần làn đường nằm ở nửa dưới của bức ảnh => Nên chọn vùng quan tâm nằm ở nửa dưới ảnh để tập trung vào phần làn đường cần detect, đồng thời bỏ qua các chi tiết thừa ở trên.



Tạo mask vùng quan tâm, kết hợp mask_image với canny_image.

=> Loại bỏ cạnh thừa

canny_image *mask_image*

5. Hough Transform:

5.1. Phương trình đường thẳng:

Hough Transform là thuật toán phát hiện đường thẳng khá hiệu quả trong xử lý ảnh sau khi thực hiện quá trình lọc ảnh Canny.

Thông thường, phương trình tổng quát của một đường thẳng cơ bản sẽ được biểu diễn theo 2 tham số m và c như sau:

$$y = mx+c$$

trong đó:

- m là gradient của đường thẳng
- c là giao điểm trên trục y

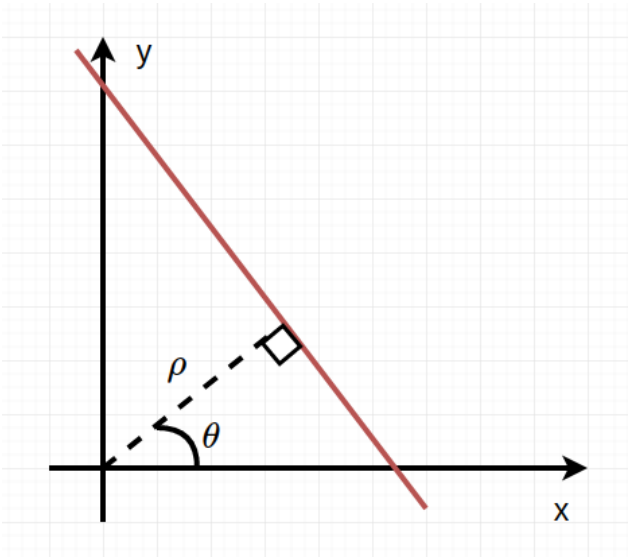
Tuy nhiên, với cách biểu diễn như này, thì giá trị m và c trải dài từ $-\infty$ đến $+\infty$. Nhưng thuật toán Hough Transform yêu cầu các giá trị phải nằm trong một khoảng xác định (hay bị chặn trên hay dưới).

$\Rightarrow$ Sử dụng hệ tọa độ cực để biểu diễn phương trình đường thẳng:

$$\rho = x\cos(\theta) + y\sin(\theta)$$

trong đó:

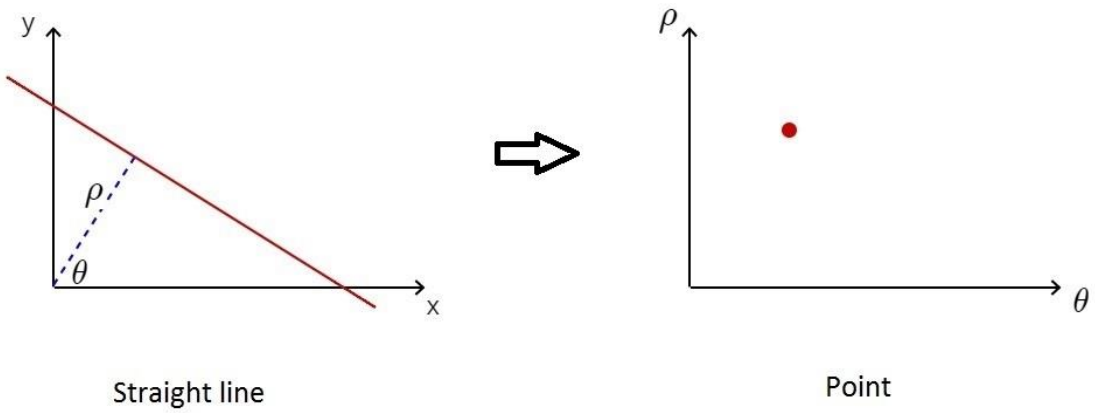
- ρ là khoảng cách góc tọa độ đến đường thẳng đang xét. Giá trị của cạnh ρ bị chặn bởi cạnh chéo của ảnh (đường thẳng màu đỏ)
- θ là góc tạo bởi ρ và trục Ox. Giá trị của góc θ giới hạn trong [0, 180)



Hình 5.1.1 Đường thẳng được biểu diễn trong hệ tọa độ cực

5.2 Mapping đường thẳng từ không gian ảnh sang không gian Hough:

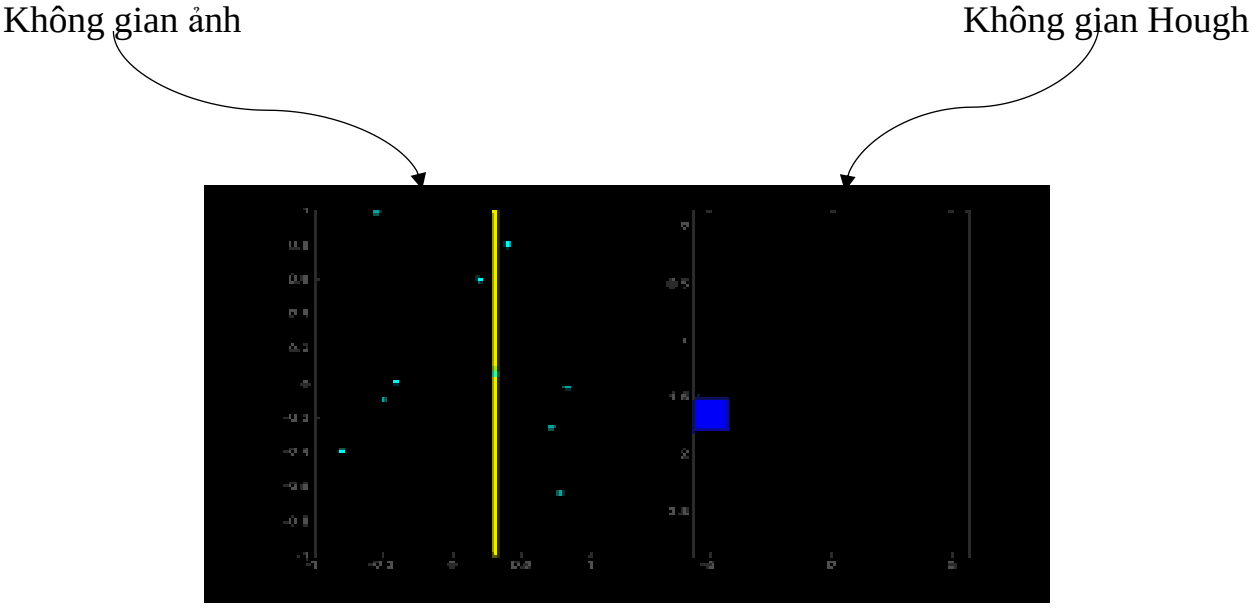
Từ một đường thẳng trong không gian ảnh, ta mapping sang không gian Hough thành một điểm theo 2 tham số ρ và θ.



Hình 5.2.1 Mapping một đường thẳng từ không gian ảnh sang không gian Houg

5.3 Bình chọn đường thẳng:

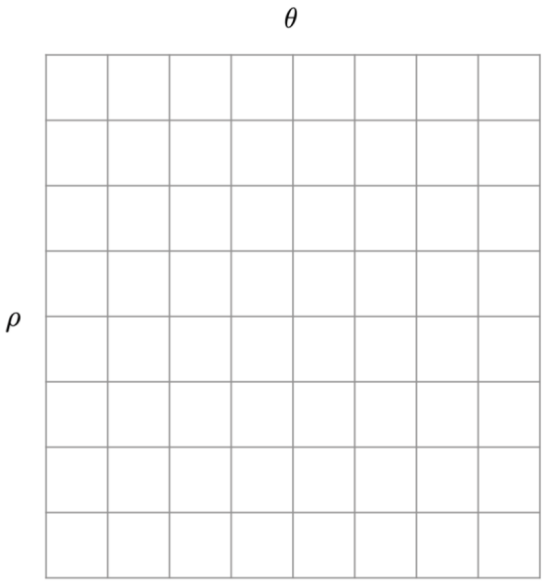
Thuật toán Hough Transform sẽ thực hiện trên 2 không gian: không gian ảnh và không gian Hough.



Hình 5.3.1 Không gian thực hiện bình chọn đường thẳng

Không gian ảnh: chứa các điểm sáng là các pixel của ảnh sau khi thực hiện bước 4.

Không gian Hough: sẽ được chia thành một lưới ô vuông nhỏ với các hàng là trục ρ và các cột là trục θ như hình dưới. Các ô trong lưới ô vuông nhỏ được khởi tạo giá trị ban đầu = 0 để tiến hành voting.

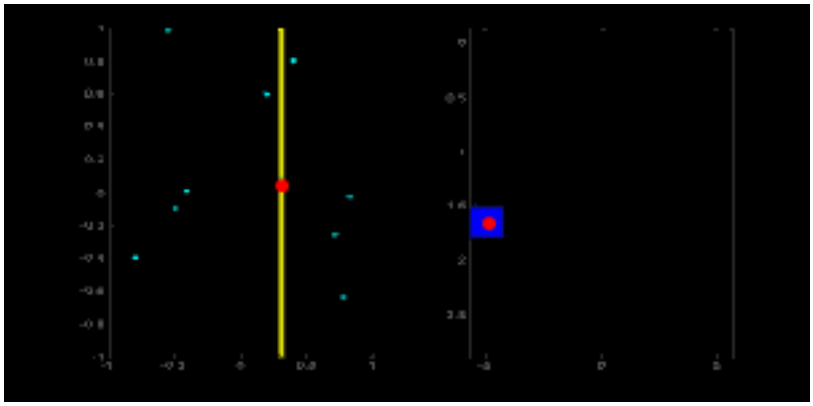


Hình 5.3.2 Biểu diễn không gian Hough

Tiến hành bình chọn (voting) đường thẳng trên 2 không gian ảnh và Hough trên theo ngưỡng *threshold* ta truyền vào để xác định các cặp (ρ, θ) của đường thẳng tương ứng cần tìm. (*threshold* – số lượng voting tối thiểu để xác định đường thẳng mong muốn).

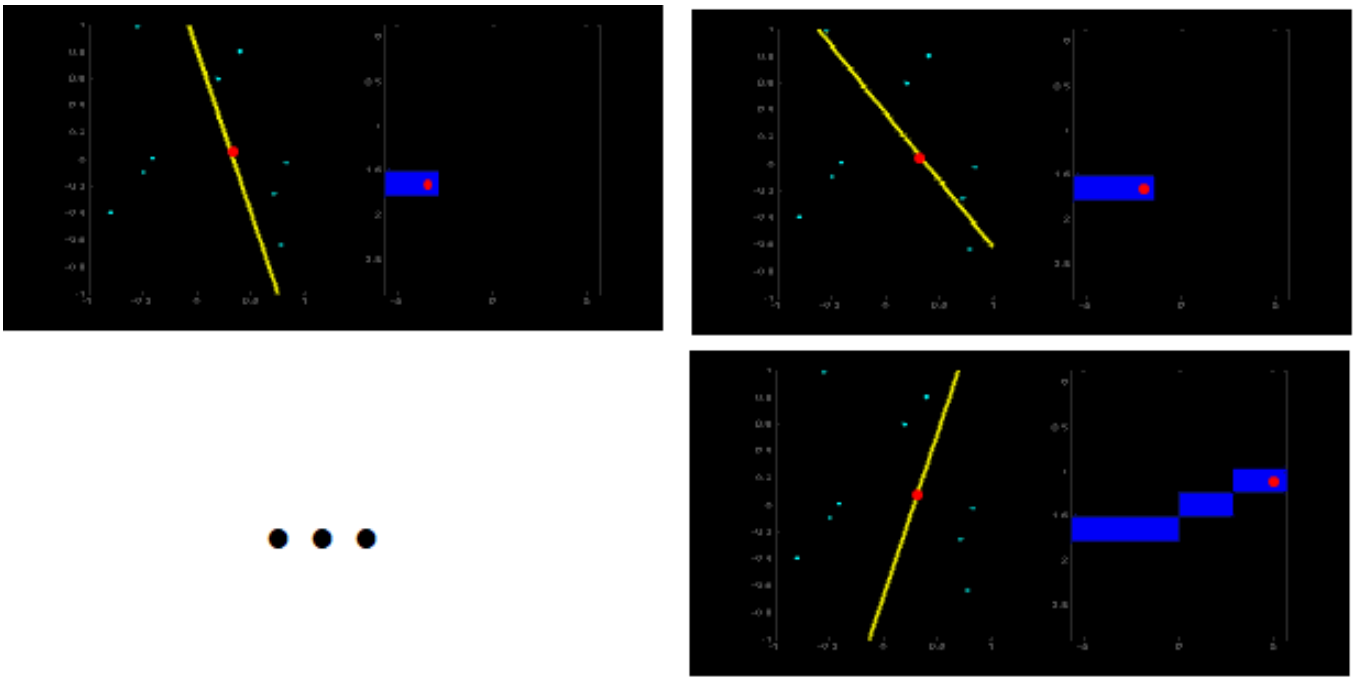
Khởi tạo: xét bất kì 1 điểm sáng trong không gian ảnh cho chạy góc θ từ 0° đến 180°.

Bắt đầu xét trường hợp góc $\theta = 0$, vẽ được đường thẳng màu vàng như hình dưới. Có được góc θ và tọa độ (x, y) là tọa độ điểm sáng đang xét, dễ dàng tính được ρ theo công thức: $\rho = x\cos(\theta) + y\sin(\theta)$, sau đó mapping qua không gian Hough thu về được cặp (ρ, θ) tương ứng. Tăng số lượng voting của ô vuông này lên 1 đơn vị là bằng 1 (ô màu xanh như hình dưới).



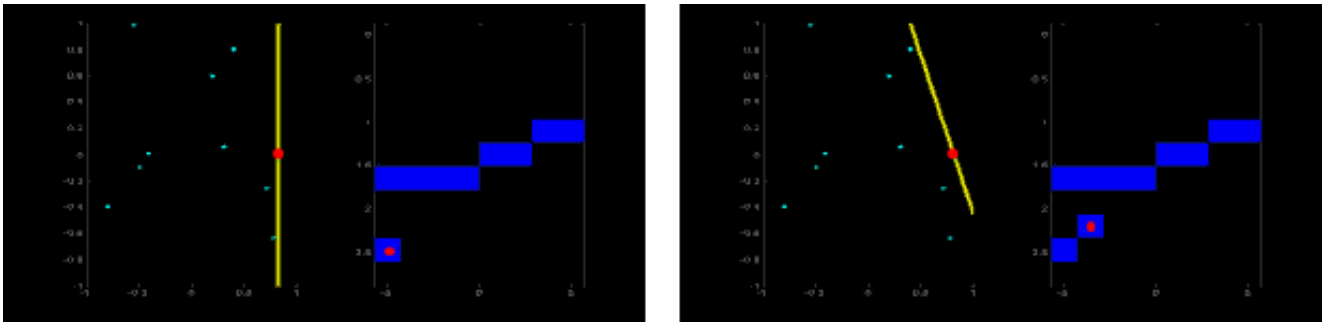
Hình 5.3.3 Trường hợp $\theta = 0$

Tiếp tục cho góc θ tăng lên, mapping qua không gian Hough thu về cặp (ρ, θ) tương ứng, và tăng số lượng voting tương ứng lên 1 đơn vị.



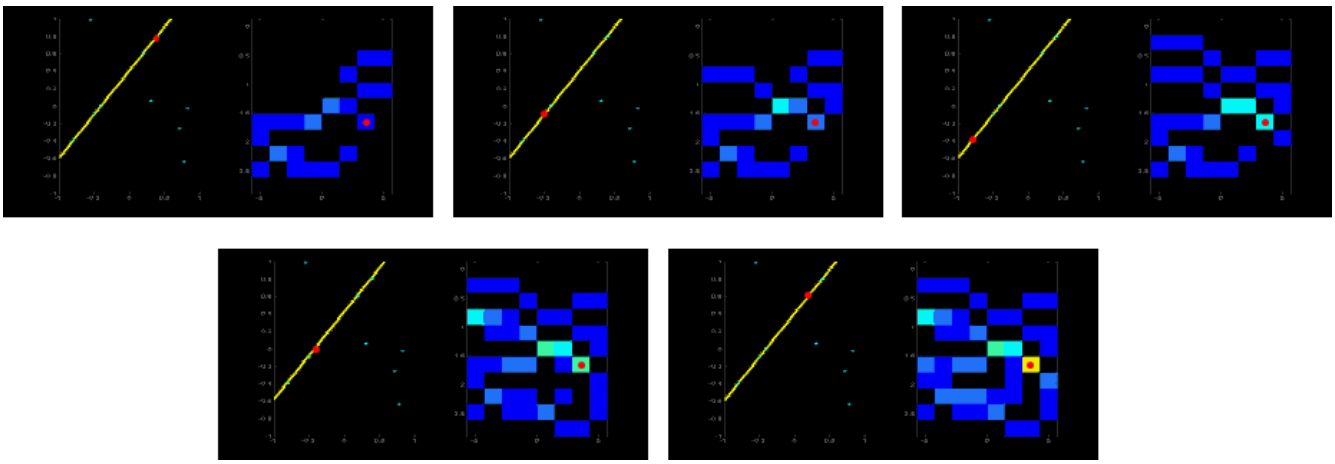
Hình 5.3.4 Tiếp tục tăng góc θ

Tăng góc θ đến khi lớn hơn hoặc bằng 180 thì chuyển sang điểm sáng khác và tiếp tục tiến hành voting.



Hình 5.3.5 Chuyển sang điểm sáng khác và tiếp tục voting

(*) Ta có thể nhận ra là 1 đường thẳng càng đi qua nhiều điểm sáng trong không gian ảnh thì số lượng voting tương ứng của nó sẽ càng cao:



Hình 5.3.6 Trường hợp 1 đường thẳng đi qua nhiều điểm sáng

Như hình *5.3.6* tương ứng với 5 điểm sáng bất kì trong không gian ảnh, ta đều cùng thu về 1 đường thẳng, mapping qua không Hough đều cùng thu về cùng 1 cặp *(ρ, θ)* và số lượng voting tương ứng của nó cũng tăng lên 5 đơn vị là 5.

Sau khi xét hết tất cả điểm sáng trong không gian ảnh, thuận toán mới xem xét các cặp *(ρ, θ)* nào có số lượng voting lớn hơn hoặc bằng *threshold* ta truyền vào, thì cho đường thẳng tương ứng với *(ρ, θ)* đó là đường thẳng cần tìm.

Thuật toán Hough TransForm trong OpenCV:

lines = cv2.HoughLines(edges,1,np.pi/180,200)

Với các tham số được sử dụng lần lượt:

- ✓ *lines*: vector lưu kết quả dưới dạng cặp *(ρ, θ)*
- ✓ *edges*: ảnh sau khi thực hiện lọc biên cạnh
- ✓ *rho*: độ phân giải của ρ theo đơn vị pixel. Ở đây có giá trị là 1 pixel

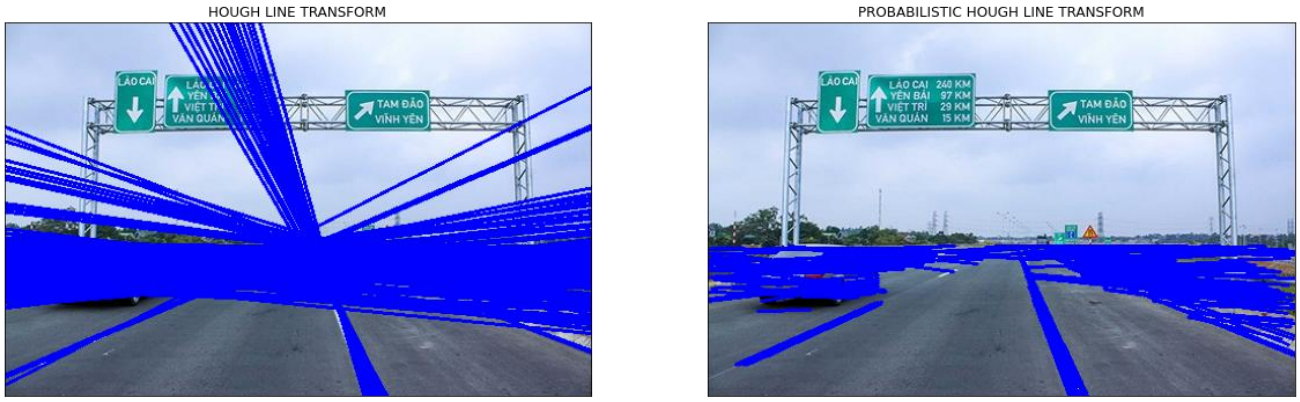
- ✓ *theta*: độ phân giải của θ theo đơn vị radian. Ở đây có giá trị là 1 độ (np.pi/180)
- ✓ *threshold*: Số lượng voting tối thiểu để xác định một đường thẳng

Ngoài ra thư viện OpenCV còn hỗ trợ thêm hàm Probabilistic Hough Line Transform là một bản chỉnh sửa của hàm Hough Line Transform, hàm này có thêm 2 tham số để lọc kết quả là *minLinLength* và *maxLineGap*:

linesP = cv2.HoughLinesP(edges,1,np.pi/180,200,minLineLength=20, maxLineGap = 180)

- ✓ *linesP*: các tọa độ *x1,y1,x2,y2* của đường thẳng phát hiện được
- ✓ *minLinLength*: số điểm tối thiểu tạo nên một đường thẳng. Các đường tìm được có số điểm nhỏ hơn số này sẽ được loại bỏ khỏi kết quả
- ✓ *maxLineGap*: Khoảng cách lớn nhất giữa 2 điểm để có thể coi chúng vẫn cùng một đường thẳng. Trong ảnh gốc rất có thể nhiều đường thẳng bị mờ, nhiễu, khi cho qua bộ lọc biên Canny, các đường thẳng sẽ trở thành các đường đứt quãng (hay các điểm riêng biệt). Tham số này quyết định việc có coi các đoạn đứt quãng đó thuộc cùng một đường hay không

Kết quả phát hiện đường thẳng sau khi chạy 2 hàm *cv2.HoughLines* và *cv2.HoughLinesP*:



Hình 5.3.7: Kết quả khi chạy 2 hàm cv2.HoughLines và cv2.HoughLinesP

(*) Nhận xét:

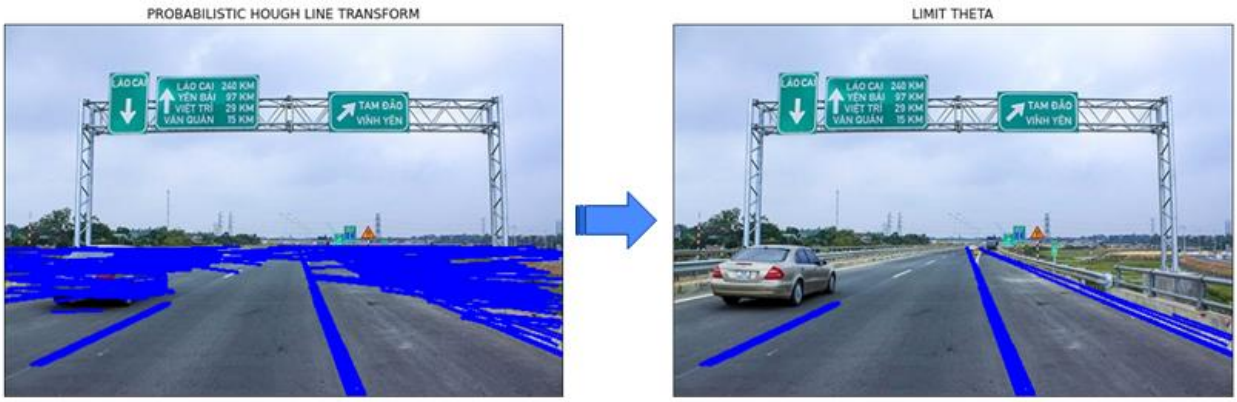
- Hàm *cv2.HoughLines* tốn quá nhiều chi phí tính toán để phát hiện đường thẳng và không xác định được điểm đầu và điểm cuối của đường thẳng.
- Hàm *cv2.HoughLinesP* tốn ít chi phí tính toán hơn, nó không kiểm tra tất cả pixel mà chỉ sử dụng 1 cụm pixel vừa đủ để phát hiện đường thẳng, và xác định được điểm đầu và điểm cuối của đường.

⇒ Sử dụng hàm *cv2.HoughLinesP* thư viện OpenCV cung cấp để phát hiện đường thẳng, xây dựng chương trình.

6. Limit Theta (θ)

Đa phần các ảnh chúng em thử nghiệm thu được từ camera ô tô, các vạch kẻ đường đều là các đường chéo, ít trường hợp là các đường nằm ngang và nằm dọc. Nên tụi em tiến hành giới hạn góc θ lại từ 15° đến 75° so với trục *Ox*, các đường có góc θ không nằm trong khoảng chỉ định ở trên đều sẽ bị loại bỏ (các đường nằm ngang và dọc - gầm xe ô tô, hàng rào, cột điện,…), chỉ giữ lại các đường chéo.

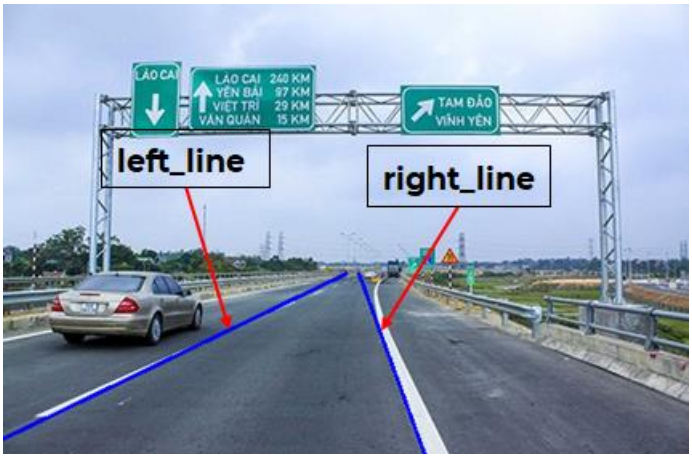
Hình 6.1 Minh họa thực hiện giới hạn góc θ



Hình 6.2: Kết quả sau khi thực hiện giới hạn góc θ

7. Draw lines

Xác định tọa độ (*x1, y1, x2, y2*) của 2 đoạn thẳng *left_line* và *right_line* trong tập hợp các đoạn thẳng phát hiện được ở bước trên, từ đó dựa vào tọa độ 2 đoạn thẳng đó để tô màu phần làn đường xe chạy. *left_line* là đoạn thẳng từ trái vào giữa trong cùng nhất và *right_line* là đoạn thẳng từ phải và giữa trong cùng nhất.



Hình 7.1: Minh họa left_line và right_line

Để xác định tọa độ 2 đoạn thẳng *left_line* và *right_line* ta cần tìm được giá trị *x0* lớn nhất đối với tập hợp *left_line* và *x0* nhỏ nhất đối với tập hợp *right_line* dựa trên tọa độ 2 đỉnh của các đoạn thẳng tìm được ở bước trên (*x1*,*y1*) (*x2*,*y2*).

Tương ứng với mỗi tọa độ (*x1*,*y1*) (*x2*,*y2*) của các đoạn thẳng tìm được, ta tính *m* và *c* tương ứng theo 2 công thức:

- $m = \frac{y2-y1}{x2-x1}$ với:

m<0: đoạn thẳng đó là *left_line*, *m>=0:* đoạn thẳng đó là *right_line*

- *c = y1 – m * x1*

Ta cho: y2 = chiều cao ảnh, *y1 = 0.6 * y1* (đây là 2 tham số tinh chỉnh, có thể thay đổi để phù hợp với đồ án).

Có được *m, c, y1, y2* ta dễ dàng tính được *x1* và *x2*.

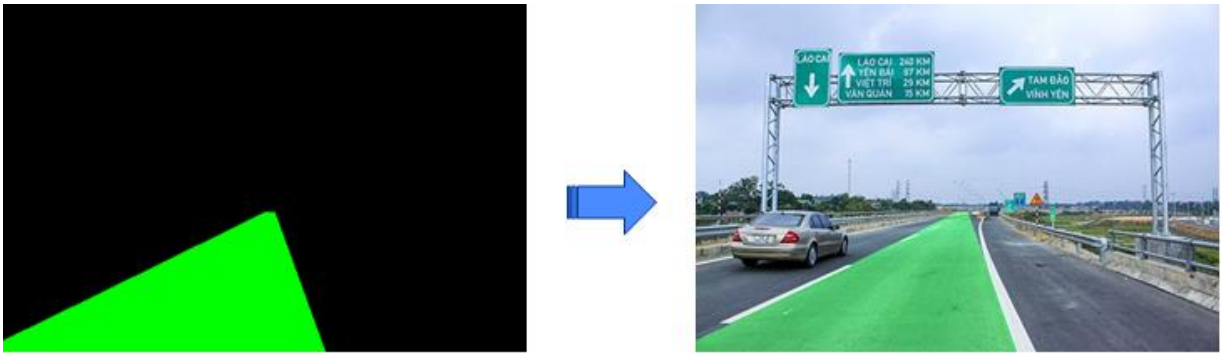
Tiếp theo tính *x0* theo công thức:

$$x0 = \frac{x1+x2}{2}$$ (hoành độ trung điểm)

Sau khi chạy hết các tọa độ (*x1*,*y1*) (*x2*,*y2*) của các đoạn thẳng tìm được, ta thu về 2 tập hợp *x0 left_line và right_line* tương ứng. Đối với tập *left_line* ta lấy giá trị lớn nhất thu về *x0max*, còn tập *right_line* ta lấy giá trị nhỏ nhất thu về *x0min*.

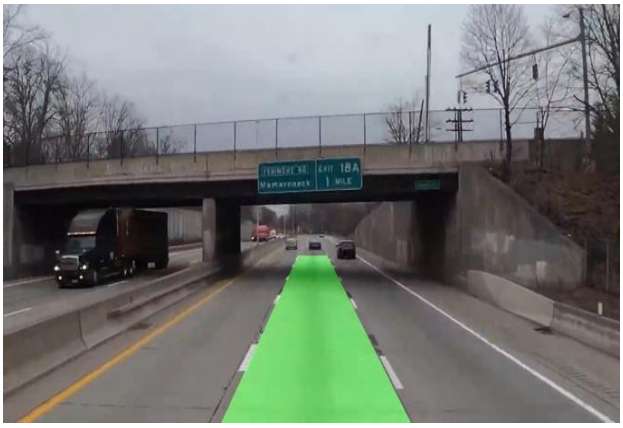
Có được 2 giá trị *x0max* và *x0min*, ta lấy tọa độ (*x1*,*y1*) (*x2*,*y2*) tương ứng với giá trị *x0max* thì sẽ thu về được tọa độ đoạn thẳng *left_line* cần tìm, tiếp tục lấy tọa độ (*x1*,*y1*) (*x2*,*y2*) tương ứng với giá trị *x0min* thì sẽ thu về được tọa độ đoạn thẳng *right_line* cần tìm.

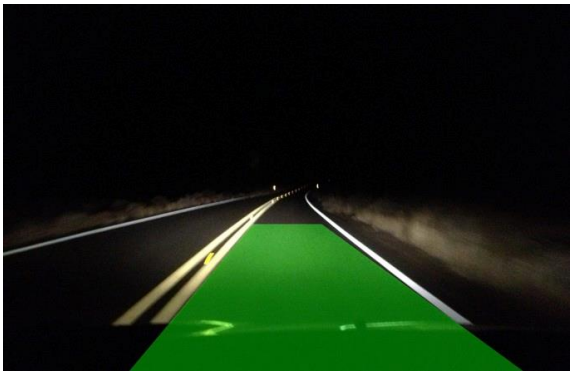
Sau khi có được tọa độ 2 đoạn thẳng *left_line* và *right_line*, ta sử dụng hàm *cv2.fillPoly()* để lấp đầy vùng được chọn - ảnh *fill_image*. Sau đó sử dụng hàm *cv2.addWeighted* để trộn 2 ảnh *input* và *fill_image*, thu về ảnh *output_image* - ảnh chứa phần tô màu làn đường xe chạy.

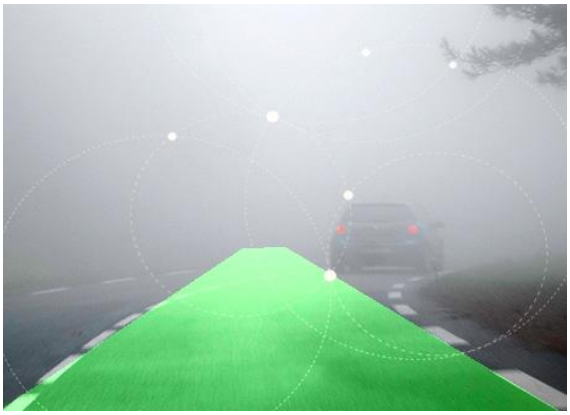


Hình 7.2: Kết quả sau khi thực hiện draw lines

III. Kết quả thử nghiệm







IV. Hạn chế

- Sẽ không detect được nếu vạch kẻ đường quá mờ, không rõ ràng.

- Không áp dụng được cho các đoạn đường cong, lúc quẹo cua, vì đây là thuật toán phát hiện đường thẳng.

- Sẽ có lỗi khi có vật cản trước mặt gây nhiễu và khi đổi sang làn đường khác.

V. Giao diện Demo

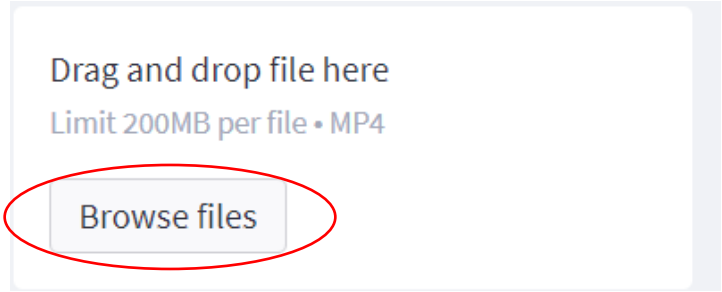
Xây dựng giao diện bằng streamlit.

Giao diện thiết kế được

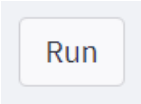
Nhấp vào combobox *image* sẽ có 2 option:

- image: thử nghiệm trên ảnh (giới hạn 200MB, định dạng PNG, JPG, JPEG)
- video: thử nghiệm trên video (giới hạn 200MB, định dạng MP4)

Nhấp vào button "*Browse files*" để load file và video lên thử nghiệm.



Nhấp vào button "*Run*" để thực thi chương trình.



Kết quả thực thi:

VI. Tài liệu tham khảo

- https://aicurious.io/posts/2019-10-24-hough-transform-phat-hien-duong-thang/
- https://minhng.info/tutorials/phat-hien-duong-thang-hough-transform.html
- https://inthiepcuoi.vn/hough-transform-la-gi/
- https://viblo.asia/p/part-2-edge-detection-voi-opencv-eW65Gv4YlDO
- https://www.kaggle.com/soumya044/lane-line-detection#5.-Let's-Make-a-Lane-Detection-Pipeline