**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

---------------o0o---------------

****

**BÁO CÁO**

**TIỂU LUẬN XỬ LÍ ẢNH**

**TÁCH ĐƯỜNG BIÊN VÀ BIẾN ĐỔI HOUGH**

**GVHD: ThS. VÕ TUẤN KIỆT**

**NHÓM 5**

**SVTH: LÊ TIẾN ĐẠT 1710948**

**NGÔ HUY CƯỜNG 1710716**

**TRẦN CHÍ VỸ 1714067**

**LÊ ĐỨC LỢI 1712060**

**TP. HỒ CHÍ MINH, THÁNG 11 NĂM 2020**

MỤC LỤC

[DANH SÁCH HÌNH MINH HỌA 3](#_Toc57148319)

[I. GIẢI THUẬT PHÁT HIỆN CẠNH CANNY (Canny Edge Detection): 3](#_Toc57148320)

[**1. Giới thiệu:** 4](#_Toc57148321)

[*a.* *Giảm nhiễu:* 4](#_Toc57148322)

[*b. Tính cường độ và hướng của gradient:* 5](#_Toc57148323)

[*c. Non-maximum suppression - loại bỏ các giá trị không phải cực đại:* 6](#_Toc57148324)

[*d. Sử dụng threshold để tạo loại bỏ cạnh giả, xác định cạnh thực sự (real edge):* 7](#_Toc57148325)

[**2. Giải thích thuật toán Canny trong OpenCV:** 8](#_Toc57148326)

[II. HOUGH TRANSFORM: 11](#_Toc57148327)

[**1.** **Hough Transform là gì?** 11](#_Toc57148328)

[**2.** **Lý thuyết về Hough Transform:** 11](#_Toc57148329)

[*a.* *Phương trình đường thẳng trong không gian (A):* 11](#_Toc57148330)

[*b.* *Mapping giữa không gian ảnh (A) và không gian Hough (B):* 12](#_Toc57148331)

[**3.** **Cài đặt thuật toán:** 14](#_Toc57148332)

[**4.** **Giải thích thuật toán của Hough Transform trong OpenCV:** 15](#_Toc57148333)

[**5.** **Giải thích thuật toán sử dụng:** 17](#_Toc57148334)

[III. TÀI LIỆU THAM KHẢO: 18](#_Toc57148335)

[IV. Trả lời câu hỏi 19](#_Toc57148336)

# DANH SÁCH HÌNH MINH HỌA

Hình I.1.b……………………………………………………………………………6

Hình I.1.d…………………………………………………………………………… 8

Hình I.2.a…………………………………………………………………………….9

Hình I.2.b…………………………………………………………………………….9

Hình I.2.c…………………………………………………………………………….10

Hình I.2.d…………………………………………………………………………….10

Hình II.1.a……………………………………………………………………………12

Hình II.1.b.1………………………………………………………………………….12

Hình II.1.b.2………………………………………………………………………….13

Hình II.1.b.3………………………………………………………………………….13

Hình II.1.b.4………………………………………………………………………….14

Hình II.3……………………………………………………………………………...15

Hình II.5.1……………………………………………………………………………17

Hình II.5.2……………………………………………………………………………17

# I. GIẢI THUẬT PHÁT HIỆN CẠNH CANNY (Canny Edge Detection):

## **1. Giới thiệu:**

- Trong hình ảnh, thường tồn tại các thành phần như: vùng trơn, góc / cạnh và nhiễu. Cạnh trong ảnh mang đặc trưng quan trọng, thường là thuộc đối tượng trong ảnh (object). Do đó, để phát hiện cạnh trong ảnh, giải thuật Canny là một trong những giải thuật phổ biến trong xử lý ảnh. Thuật toán Canny (Canny Edge Detection) là một giải thuật phát hiện cạnh nổi tiếng được phát triển năm 1986 bởi John. F Canny.

- Một cách tổng quát, thuật toán Canny Edge Detection gồm có 4 bước:

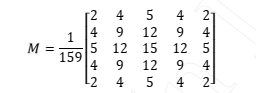
* Loại bỏ nhiễu
* Tính cường độ và hướng của gradient
* Non-maximum supression
* Sử dụng threshold để tạo loại bỏ cạnh giả, xác định cạnh thực sự (real edge)

1. *Giảm nhiễu:*

* Vì tất cả các kết quả phát hiện cạnh đều dễ bị ảnh hưởng bởi nhiễu trong ảnh, nên việc lọc nhiễu để ngăn chặn phát hiện sai gây ra là điều cần thiết. Để làm mịn hình ảnh, một bộ lọc Gaussian được đối chiếu với hình ảnh. Bước này sẽ làm mịn ảnh một chút để giảm ảnh hưởng của nhiễu rõ ràng trên bộ dò cạnh. Phương trình cho bộ lọc Gaussian có kích thước (2k + 1) × (2k + 1) như sau:



* Chẳng hạn sử dụng bộ lọc Gauss 5x5 với hệ số δ = 1.4:



- Điều quan trọng là việc lựa chọn kích thước của bộ lọc Gaussian sẽ ảnh hưởng đến hiệu suất của bộ dò. Kích thước càng lớn thì độ nhạy nhiễu của máy dò càng thấp.

- Ngoài ra, lỗi bản địa hóa để phát hiện cạnh sẽ tăng lên một chút khi kích thước hạt nhân bộ lọc Gaussian tăng lên. Kích thước 5 × 5 là một kích thước tốt cho hầu hết các trường hợp, nhưng điều này cũng sẽ khác nhau tùy thuộc vào các tình huống cụ thể.

### *b. Tính cường độ và hướng của gradient:*

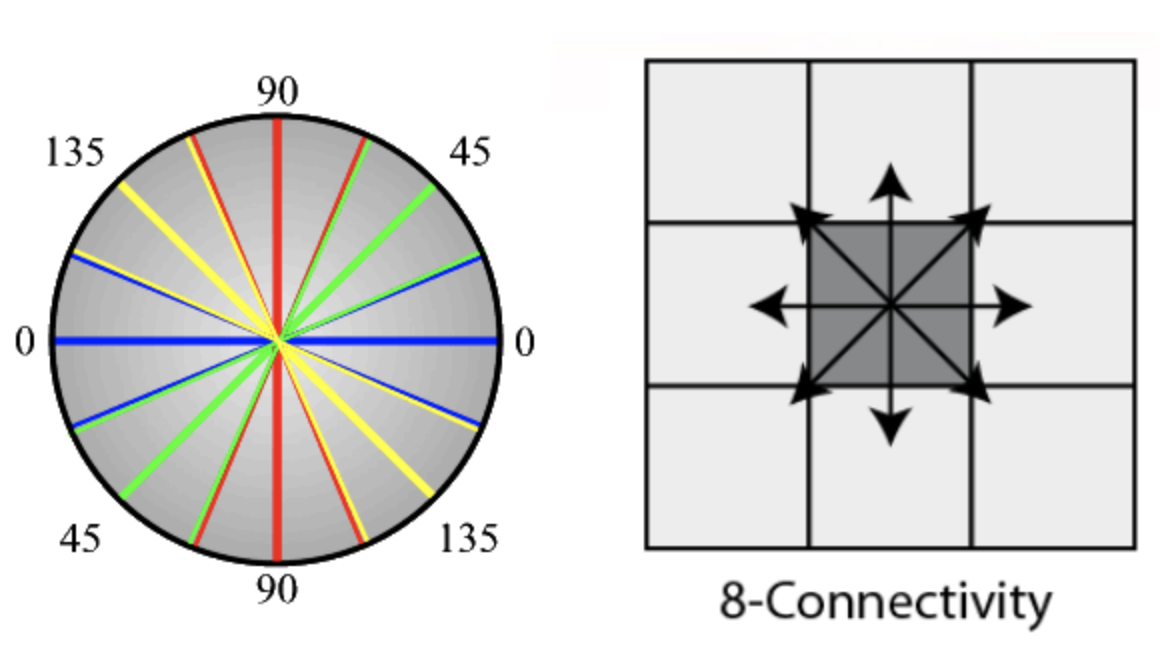
- Đường biên trong ảnh là nơi phân cách giữa các đối tượng khác nhau, nên tại đó gradient của nó sẽ có biến đổi mạnh mẽ nhất. Để tính toán gradien trong ảnh, ta có thể sử dụng bộ lọc Sobel, hoặc trực tiếp nhân chập ma trận ảnh với các mặt nạ theo hướng x và y, chẳng hạn:

https://codelearn.io/Media/Default/Users/NTN/IMG3/02.png

* Sau đó tính độ lớn gradien trong ảnh:

https://codelearn.io/Media/Default/Users/NTN/IMG3/03.png

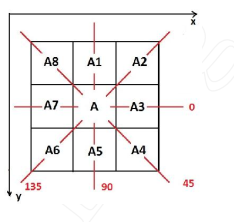
* Trong đó, Gx và Gy chính là đạo hàm theo hướng x, y của ảnh ta đang xét. Góc θ sẽ được làm tròn theo các hướng thẳng đứng, nằm ngang và theo hướng chéo. nghĩa là nó sẽ được làm tròn để nhận các giá trị trong 0, 45, 90 và 135 độ.



*Hình I.1.b*

### *c. Non-maximum suppression - loại bỏ các giá trị không phải cực đại:*

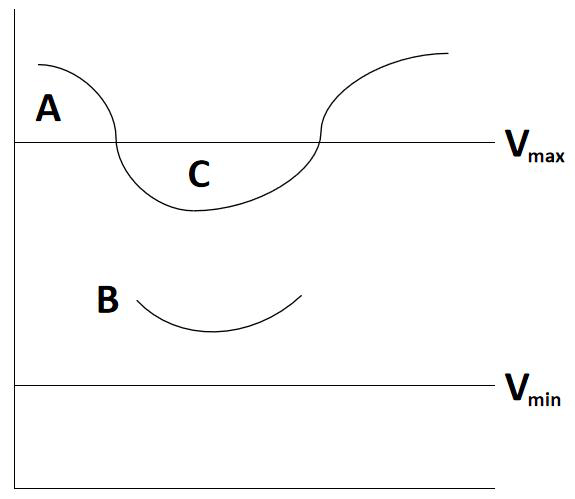
- Bước này sẽ tìm ra những điểm ảnh có khả năng là biên ảnh nhất bằng cách loại đi những giá trị không phải là cực đại trong bước tìm gradien ảnh ở trên. Ta thấy rằng, với giá trị của góc θ  ở trên thì biên của đối tượng có thể tuân theo bốn hướng, và ta có bốn khả năng sau:

* Nếu θ = 0 , khi đó điểm A sẽ được xem xét là một điêm trên biên độ nếu độ lớn gradien tại A lớn hơn gradien của các điểm tại A3, A7.
* Nếu θ = 45, khi đó điểm A được xem là một điểm trên biên độ nếu độ lớn gradien tại A lớn hơn độ lớn gradien của các điểm tại A4, A8
* Nếu θ = 90, khi đó điểm A sẽ được coi là một điểm nằm trên biên độ nếu độ lớn gradien tại A lớn hơn độ lớn gradien của các điểm tại A1, A5.
* Nếu θ = 135, khi đó điểm A được xem là một điểm nằm trên biên độ nếu độ lớn gradien tại A lớn hơn độ lớn gradien của các điểm tại A2, A6

### *d. Sử dụng threshold để tạo loại bỏ cạnh giả, xác định cạnh thực sự (real edge):*

- Bước này sẽ quyết định một cạnh ta dự đoán ở các bước trên nó có phải là một cạnh thật sự hay không. Giá trị threshold ở đây có hai ngưỡng Vmax và Vmin . Ta triển khai thuật toán dựa vào hai giá trị này như sau:

* Nếu cường độ của gradient(Magnitude) > Vmax thì đó là chắc chắn là một cạnh (strong edge)
* Nếu Magnitude < low threshold < Vmin thì đó là noise
* Nếu Vmin < Magnitude < Vmax thì đó là một cạnh yếu chưa xác định được là cạnh hay nhiễu (weak edge)
* Đối với những weak edge, nếu cạnh nào có kết nối với một strong edge thì weak edge đó là cạnh, nếu không sẽ là noise



*Hình I.1.d*

* Ví dụ ở hình trên A nằm trên ngưỡng Vmax nên A chắc chắn là cạnh (strong edge). C nằm giữa ngường Vmax và Vmin nên C là cạnh yếu (weak edge) nhưng C kết nối với strong edge là A nên C cũng là một cạnh. Tuy nhiên B cũng nằm trong ngưỡng giống C nhưng không kết nối với cạnh nào chắc chắn là cạnh (strong edge) nên B là nhiễu

## **2. Giải thích thuật toán Canny trong OpenCV:**

- OpenCV thực hiện tất cả các bước trên bằng một hàm duy nhất cv2.Canny(). Tham số đầu tiên là ảnh đầu vào.

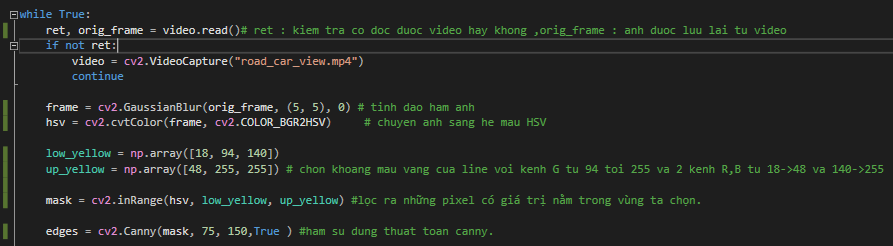
* Tham số thứ hai và thứ ba tương ứng với minVal và maxVal.
* Tham số thứ tư là aperture\_size cho biết kích thước của Sobel Kernel, mặc định là 3.
* Tham số cuối cùng là L2gradient cho biết công thức tính gradient.

+ Nếu L2gradient=True sẽ sử dụng công thức đã được đề cập ở trên, ngược lại nó sẽ được tính theo công thức:

Edge\_Gradient(G)=|Gx|+|Gy|(G)=|Gx|+|Gy|.

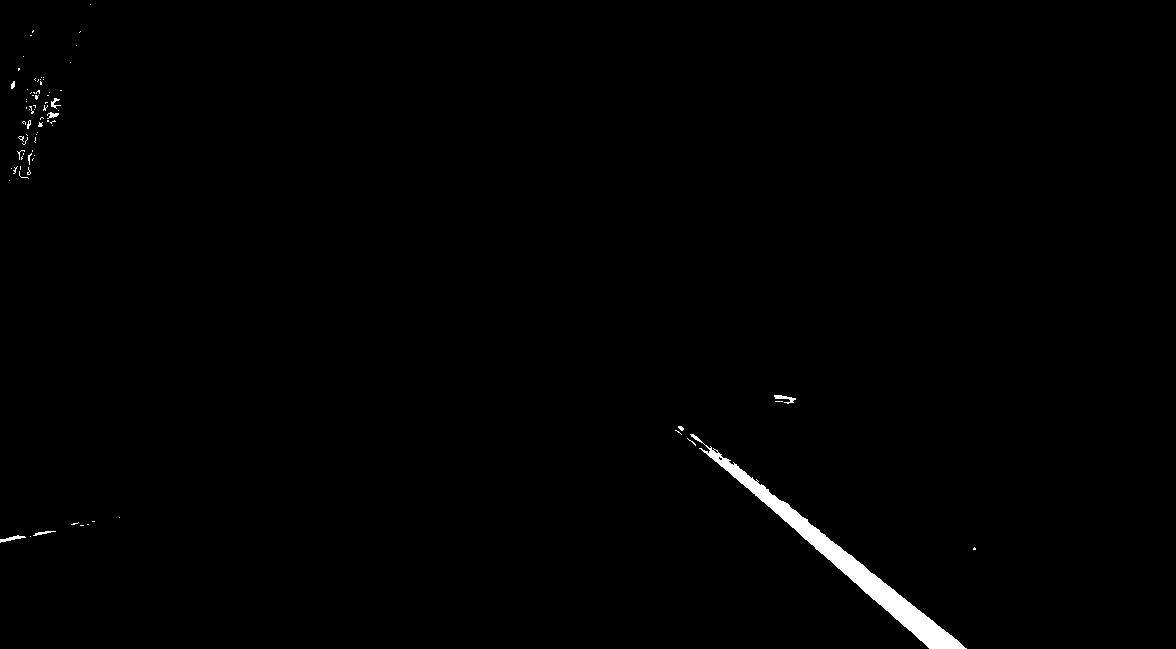
+ Mặc định, L2gradient=False.

*Giải thích thuật toán sử dụng:*

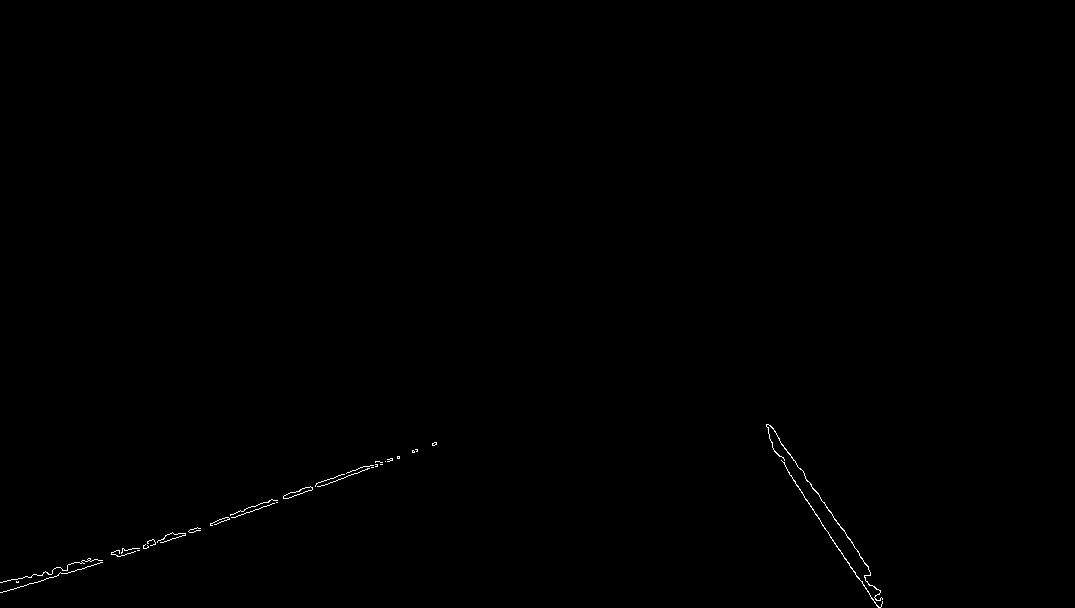
*Hình I.2.a*



*Hình I.2.b Hệ màu HSV trong video*



*Hình I.2.c Pixel màu vàng được chọn trong video*



*Hình I.2.d Cạnh phát hiện từ những pixel đường kẻ line màu vàng*

# II. HOUGH TRANSFORM:

1. **Hough Transform là gì?**

Biến đổi Hough là một kỹ thuật trích xuất đặc trung được sử dụng trong phân tích hình ảnh, thị giác máy tính và xử lý hình ảnh số. Mục đích của kỹ thuật này là tìm các trường hợp không hoàn hảo của các đối tượng trong một lớp hình dạng nhất định bằng thủ tục bình chọn (voting). Thủ tục bình chọn này được thực hiện trong một khoảng tham số, từ đó các đối tượng nhận được dưới dạng cực đại cục bộ trong một không gian được gọi là bộ tích lũy, bộ tích lũy được xây dựng bằng thuật toán tính toán biến đổi Hough.

1. **Lý thuyết về Hough Transform:**

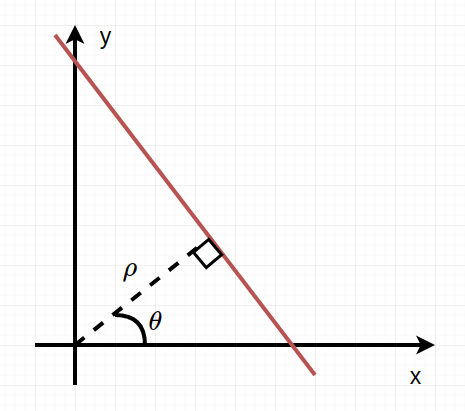
Ý tưởng chung của việc phát hiện đường thẳng trong thuật toán này là tạo mapping từ không gian ảnh (A) sang một không gian mới (B) mà mỗi đường thẳng trong không gian (A) sẽ ứng với một điểm trong không gian (B).

1. *Phương trình đường thẳng trong không gian (A):*

* Phương trình đường thẳng cơ bản sẽ được biểu diễn theo 2 tham số a và b như sau:

y = ax + b

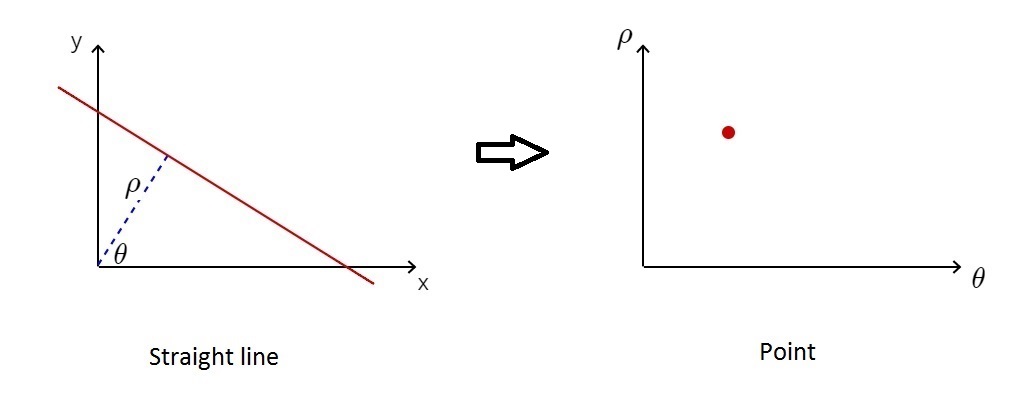
* Tuy nhiên, với cách biểu diễn này, giá trị của góc nghiêng a trải dài từ -∞ đến +∞. Có thể lấy ví dụ, để có được phương trình đường Oy (x=0) thì a phải tiến tới ∞. Thuật toán Hough Transform yêu cầu các giá trị a, b nằm trong một khoảng xác định (hay bị chặn trên dưới), ta phải sử dụng hệ tọa độ cực để biểu diễn phương trình đường thẳng: ρ = xcos(θ) + ysin(θ).
* Xét thấy trong phương trình tọa độ cực, giá trị của góc θ có thể bị chặn lại trong khoảng [0, π). Trên thực tế, không gian ảnh là không gian hữu hạn (bị chặn lại bởi các cạnh của ảnh), do vậy giá trị ρ cũng bị chặn.



*Hình II.1.a Đường thẳng trong hệ tọa độ cực*

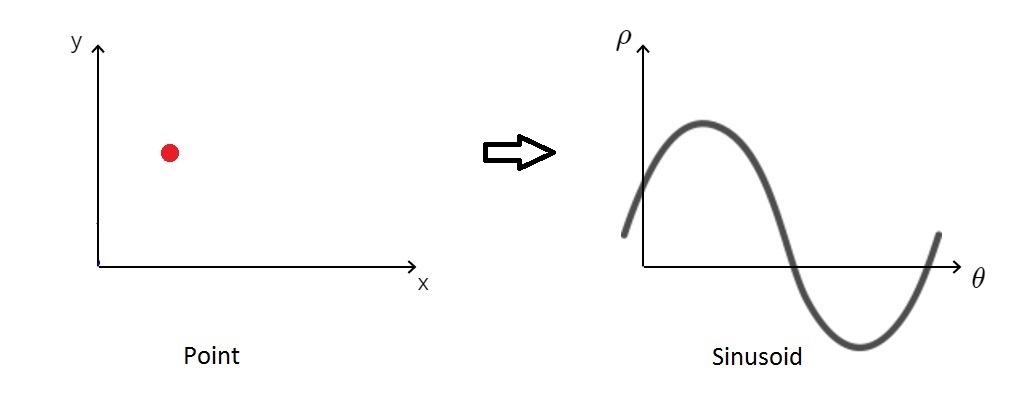
1. *Mapping giữa không gian ảnh (A) và không gian Hough (B):*

* Từ một đường thẳng trong không gian ảnh (A) với 2 tham số ρ và θ, chúng ta sẽ map sang không gian Hough (B) thành một điểm.



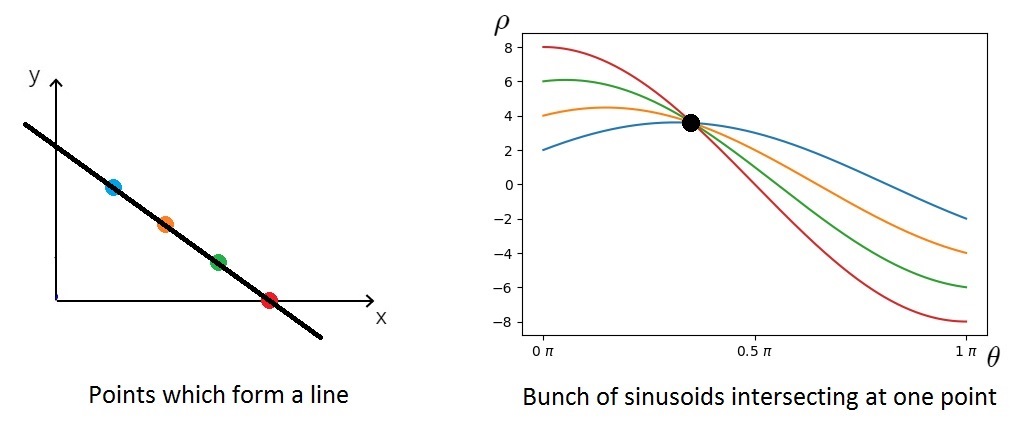
*Hình II.1.b.1 Mapping một đường thẳng từ không gian ảnh sang không gian Hough*

* Từ một điểm trong không gian ảnh, chúng ta lại có được một hình sin trong không gian Hough:



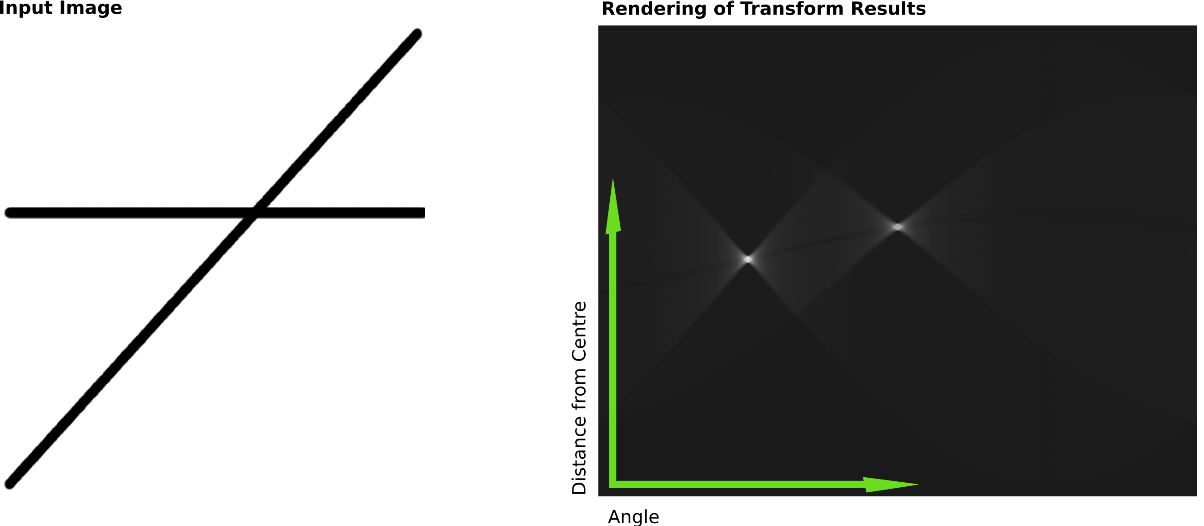
*Hình II.1.b.2 Mapping một điểm từ không gian ảnh sang không gian Hough*

* Các điểm nằm trên cùng một đường thẳng lại có biểu diễn là các hình sin giao nhau tại một điểm trong không gian Hough. Đây là nơi xuất phát ý tưởng của thuật toán Hough Transform. Chúng ta sẽ dựa vào các điểm giao nhau này để suy ngược lại phương trình đường thẳng trong không gian ảnh.



*Hình II.1.b.3 Mapping nhiều điểm thẳng hàng từ không gian ảnh sang không gian Hough*

* Mỗi đường thẳng khác nhau sẽ tạo thành một điểm sáng (nơi giao nhau của nhiều hình sin) trên không gian Hough. Dưới đây là sự biểu diễn 2 đường thẳng trong không gian Hough.

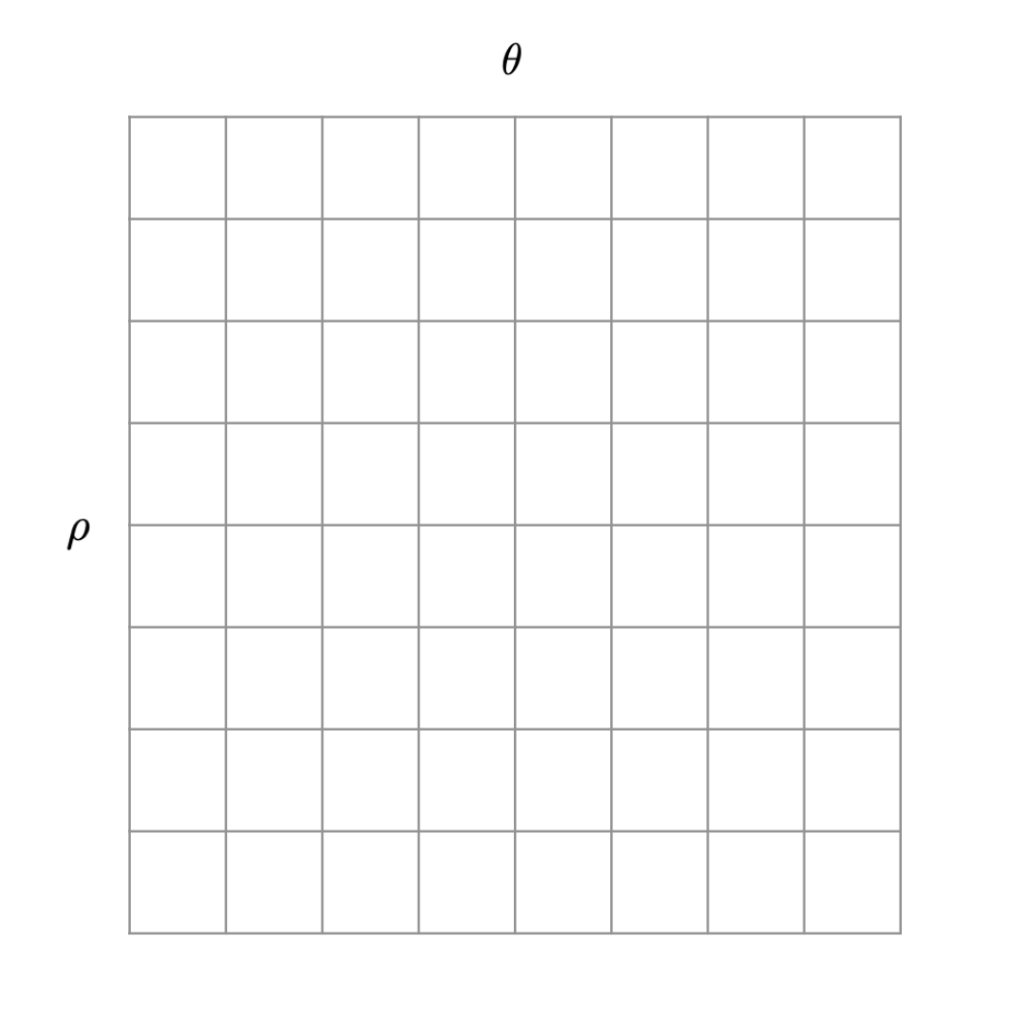


*Hình II.1.b.4 Biểu diễn 2 đường thẳng trong không gian Hough*

1. **Cài đặt thuật toán:**

* *Tiền xử lý:* Hough Transform yêu cầu đầu vào là một ảnh nhị phân. Trên thực tế ảnh sẽ được đưa về dạng ảnh xám, áp dụng các thuật toán lọc biên để xác định các đường biên trong ảnh. Ở đây chúng ta sẽ sử dụng thuật toán Canny để lọc biên.
* *Bình chọn đường thẳng (voting):*

+ Ta chia không gian Hough ra thành một lưới ô vuông nhỏ. Ta sẽ có một lưới ô vuông với các hàng là trục ρ và các cột là trục θ như hình dưới. Độ chính xác của thuật toán phụ thuộc vào số lượng các ô vuông bạn chọn cho mỗi cạnh. Giả sử bạn muốn độ chính xác của θ là 1 độ (theta resolution = 1), bạn cần 180 cột. Giá trị ρ bị chặn bởi cạnh chéo của ảnh đầu vào. Do vậy khi lấy độ chính xác của ρ là 1 (pixel) (rho resolution = 1) thì số hàng bằng độ dài đường chéo ảnh theo đơn vị pixel.



*Hình II.3*

+ Thực hiện voting để tìm đường thẳng:

* Các ô trong lưới ô vuông, được đặt giá trị ban đầu là 0. Xét các điểm trên ảnh đầu vào (chính là ảnh nhị phân thu được sau quá trình lọc biên sử dụng Canny), với mỗi điểm sáng, ta xét θ trong khoảng [0, 180). Vì đã biết tọa độ điểm (x, y), ta dễ dàng tính được giá trị ρ. Với mỗi cặp (θ,ρ), ta thực hiện vote (tăng giá trị tương ứng trên lưới ô vuông lên 1 đơn vị). Cuối cùng ta lấy một ngưỡng (threshold) để xác định trên lưới ô vuông, cặp giá trị (θ,ρ) nào ứng với một đường thẳng trên thực tế.

1. **Giải thích thuật toán của Hough Transform trong OpenCV:**

* Áp dụng bộ lọc Hough Line Transform để lọc đường thẳng:

lines = cv.HoughLines(edges, 1, np.pi / 180, 150, None, 0, 0)

* Các tham số được sử dụng lần lượt là:

+ edges: đầu ra của bộ lọc biên

+ lines: vector lưu kết quả dưới dạng (ρ, θ)

+ rho: độ phân giải của ρ theo đơn vị pixel, có giá trị là 1 pixel

+ theta: độ phân giải của θ tính theo radian, có giá trị là 1 độ (np.pi/180)

+ threshold: số lượng voting tối thiểu để xác định một đường thẳng

+ srn, stn: 2 tham số cuối. Khi sử dụng thuật toán Hough transform nguyên bản thì 2 tham số này bằng 0

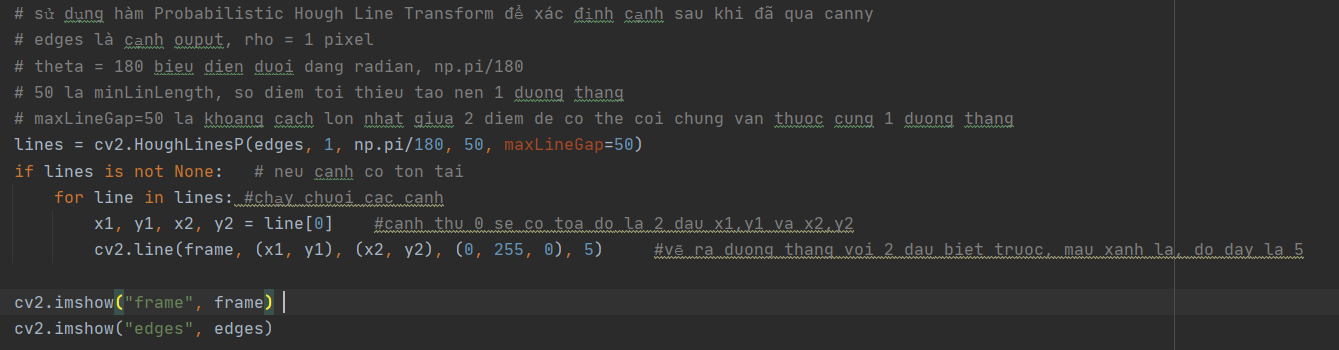
* Ngoài ra thư viện openCV còn hỗ trợ thêm Probabilistic Hough Line Transform, là một bản chỉnh sửa, thêm vào các chức năng lọc kết quả:

linesP = cv.HoughLinesP(edges, 1, np.pi / 180, 50, None, 50, 10)

+ Hàm này ngoài các tham số giống cv.HoughLines như edges, rho, theta, threshold còn có thêm:

* 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 qủa
* 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.

1. **Giải thích thuật toán sử dụng:**



*Hình II.5.1*

* Kết quả nhận dạng được line đường từ giải thuật xác định cạnh:



*Hình II.5.2*

# III. TÀI LIỆU THAM KHẢO:

[1] ThS.Võ Đức Khánh, GS.TSKH Hoàn Kiếm, “Giáo trình XỬ LÝ ẢNH”, Đại học Công Nghệ Thông Tin, Đại học Quốc gia thành phố Hồ Chí Minh

[2] Minh Nguyen, “Phát hiện đường thẳng bằng Hough Transform (Hough Line)”, minhng.info/

[3] AI Curious, “Phát hiện đường thẳng với Hough Transform – OpenCV”, aicurious.io/

[4] Thị Giác Máy Tính, “Tìm đường thẳng bằng hàm Houghlines”, thigiacmaytinh.com/

[5] RohanDeySarkar, “line detection”, gist.github.com/

# IV. Trả lời câu hỏi

1/ Tại sao phải chuyển ảnh qua hệ HSV?

Chuyển ảnh/video qua hệ màu HSV là để lọc màu. Bời vì với hệ màu HSV chúng ta xác định 1 màu cụ thể ( dễ hơn so với hệ RGB) dựa trên các màu sắc và các dải bão hòa, với sự khác biệt về giá trị pixel

Lý do cần lọc màu (nhóm em chọn cụ thể là màu vàng) là vì ta cần xét sự chênh lệch giữa các giá trị pixel lân cận (giá trị pixel có thể hiểu là mức sáng của điểm ảnh). Vùng pixel lân cận mà có độ lệch mức sáng càng lớn => biên

2/ Sự khác biệt giữa Canny và Hough?

Canny là tách biên đối tượng trong ảnh/video. Biên của Canny có hình dạng bất kỳ

Hough là lấy kết quả Canny làm tiếp tục, cụ thể biên là đường thẳng thì lấy, còn hình dạng khác thì loại bỏ đi

3/ Các tham số trong 2 câu lệnh “cv2.Canny(mask,75,150)” và “cv2.HoughLinesP(edges, 1, np.pi/180, 50, maxLineGap=50)” là gì?

* cv2.Canny(mask,75,150)

+mask là biến chứa ảnh nhị phân đầu ra ở bước 3(Non-maximum Suppression) trong giải thuật Canny 4 bước

+75: ngưỡng dưới, các pixel có giá trị < 75 bị loại bỏ

+150:ngưỡng trên, các pixel có giá trị > 150 là pixel cạnh

* cv2.HoughLinesP(edges, 1, np.pi/180, 50, maxLineGap=50)

+edges là biến chứa ảnh chỉ có pixel cạnh sau khi được lọc ngưỡng ở bước 4 giải thuật Canny

+”1” là đại lượng p (khoảng cách từ O đến đường thẳng y=ab+b bên tọa độ Decart). ở đây lấy độ khít của p = 1 pixel

+”np.pi/180” là góc theta hợp bởi khoảng cách p và trục Ox bên tọa độ Decart. ở đây góc theta tính bằng đơn vị radian

+”50”: là threshold, số lượng voting tối thiểu để xác định một đường thẳng

+”maxLineGap”: Khoảng cách lớn nhất giữa 2 điểm để có thể xem chúng vẫn cùng thuộc 1 đường thẳng. Trong ảnh/video gốc 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 1 đường thẳng hay không