**Chương I: TỔNG QUAN**

**Lý do chọn đề tài.**

Việc ước lượng đường chân trời trong video là một bài toán quan trọng trong lĩnh vực thị giác máy tính và xử lý ảnh. Việc ước lượng đường chân trời giúp xác định vị trí của các đối tượng trong video. Điều này có thể hỗ trợ trong việc phát hiện và theo dõi các hành vi đáng ngờ hoặc cảnh báo các hành vi không mong muốn, ví dụ cho việc đó chúng ta có thể ứng dụng vào ngành công nghệ lái xe như ước lượng khoảng cách với các xe phía trước, từ đó có thể cảnh báo khi khoảng cách quá gần, hay chế độ tự động lái bằng cách giữ khoảng cách với xe phía trước. Cho đến tận ngày nay, khi mà điều khiển một chiếc máy bay theo quy tắc VFR (viết tắt của Vision flight rules) là tập hợp những quy tắc hướng dẫn phi công trong điều kiện thời tiết cho phép có thể dùng mắt thường định vị những vị trí, đường đi, né tránh chướng ngại vật của máy bay, phi công cũng sử dụng các mối quan hệ trực quan giữa mũi của máy bay và đường chân trời để điều khiển máy bay. Một phi công có thể dựa vào đường chân trời để nhằm mục đích thực hiện việc định hướng không gian.

**Phương pháp tiếp cận.**

Đọc, xem các tài liệu hướng dẫn liên quan đến cách phát hiện, ước lượng đường chân trời

Tìm hiểu và cài đặt các phần mềm, thư viện để hỗ trợ thực hiện đề tài

Rút kinh nghiệm thực tế từ quá trình thực đề tài

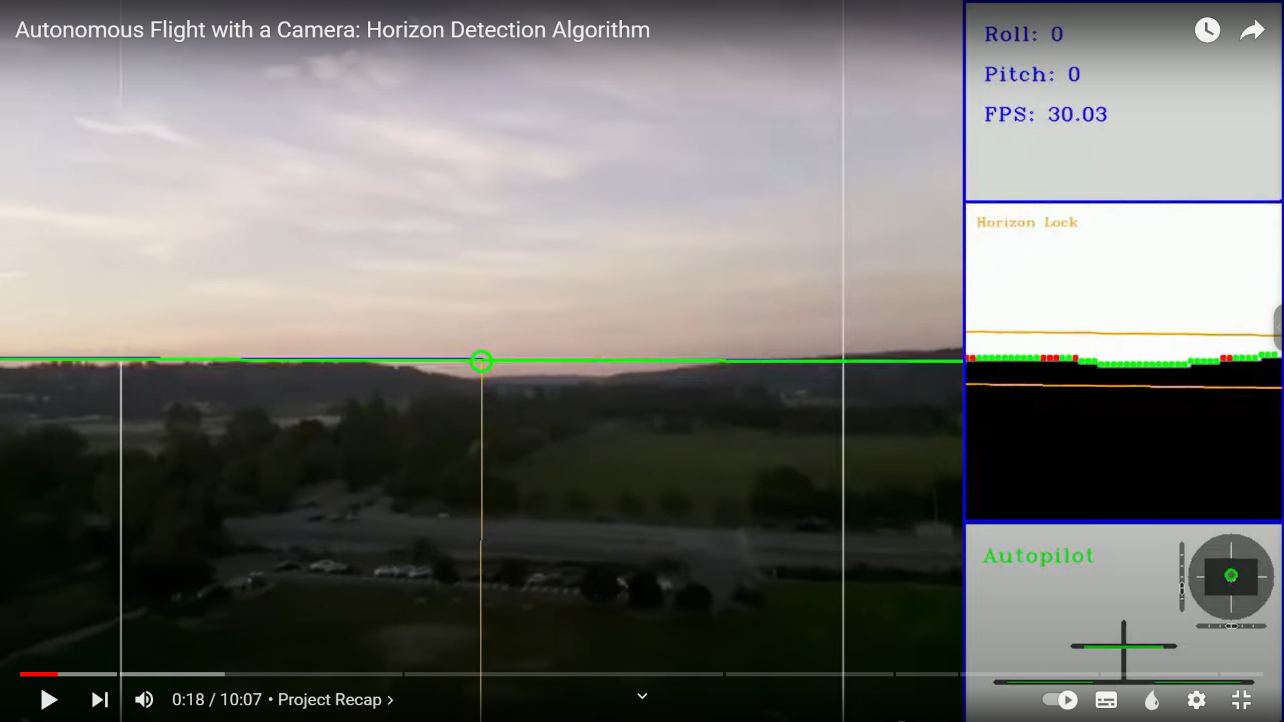
**Chương II: Cơ sở lí thuyết**

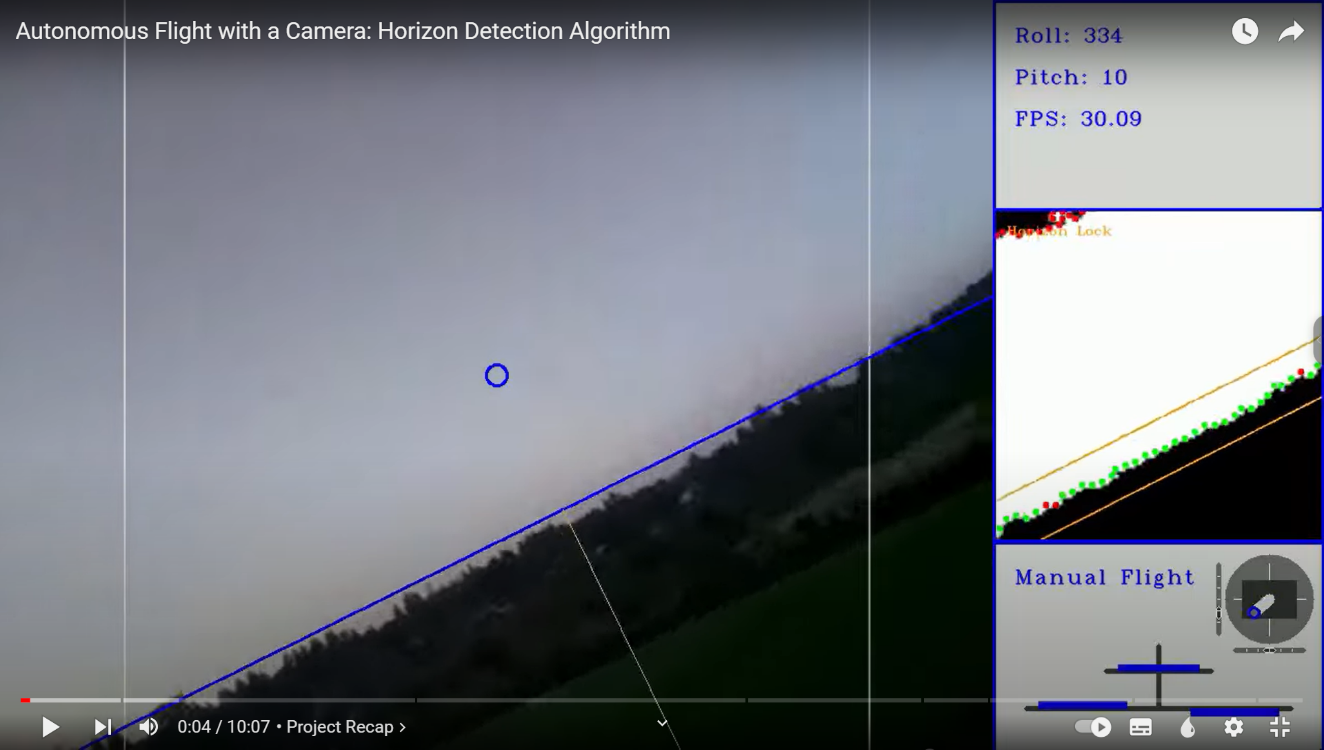
**Khái niệm đường chân trời.**

Đường chân trời được hiểu chính là một đường thẳng mà chúng ta sẽ không thấy rõ ràng ở vô cực mà mọi điểm, đường thẳng khác đều quy về nó. Hiểu một cách khác đường chân trời là 1 tên gọi để chỉ rìa mép vòng cung của Trái đất trong tầm nhìn đôi mắt chúng ta quan sát được. Nói đơn giản hơn đường chân trời là một đường phân cách giữa mặt đất với bầu trời mà ta có thể nhìn thấy rõ ràng.

Ví dụ đường chân trời và ứng dụng:

Như ảnh bên dưới được lấy từ chiếc máy bay tự động lái , đường kẻ màu xanh chính là đường chân trời, đường chân trời phân ra bầu trời và mặt đất. Nếu đường chân trời trong video bị nghiêng sang phải, tức là chiếc máy bay đã bị nghiêng sang phải và ngược lại. Từ đó chúng ta có thể xử lí để chiếc máy bay có thể cân bằng.



**  
  
  
  
  
  
Hoạt động phần mềm**

Nhận dữ liệu video: Phần mềm nhận dữ liệu video từ nguồn cung cấp, chẳng hạn như webcam, camera an ninh hoặc bất kỳ nguồn nào khác cung cấp video liên tục.

Tiền xử lý: Dữ liệu video được tiền xử lý để chuẩn bị cho quá trình phát hiện đường chân trời. Các bước tiền xử lý có thể bao gồm chuyển đổi sang không gian màu khác nhau, giảm nhiễu và cân bằng độ sáng.

Phát hiện đường chân trời: Sử dụng các thuật toán và phương pháp phát hiện đường chân trời như Hough Transform, phân đoạn hình ảnh hoặc mạng nơ-ron học sâu, phần mềm phát hiện các đường thẳng hoặc các vùng tương ứng với đường chân trời trong khung hình.

Vẽ đường chân trời trên video: Sau khi đường chân trời được phát hiện, phần mềm vẽ các đường này lên khung hình video để thể hiện cho người dùng thấy.

Hiển thị kết quả: Cuối cùng, phần mềm hiển thị video đã được xử lý, với đường chân trời được vẽ lên trên đó, để người dùng có thể quan sát và sử dụng.

Quá trình này diễn ra liên tục và nhanh chóng để đảm bảo rằng kết quả được cung cấp gần như ngay lập tức, tạo ra trải nghiệm xử lý video real-time. Đồng thời, phần mềm cũng cần đảm bảo rằng hiệu suất của nó đủ để xử lý dữ liệu video một cách hiệu quả, đặc biệt khi xử lý video ở độ phân giải cao và tốc độ khung hình nhanh.

**Đối tượng sử dụng**

Các cá nhân, tập thể có nhu cầu dùng đường chân trời để ứng dụng dô các nghiên cứu, phát triển dự án phát hiện vị trí, đối tượng. Hoặc các đối tượng có nhu cầu nghiên cứu về thuật toán được dùng trong phần mềm để tối ưu và phát triển thuật toán

**Mục đích phần mềm**

Mục đích của hệ thống phát hiện đường chân trời real-time là cung cấp một giải pháp tự động và hiệu quả để phát hiện và theo dõi đường chân trời trong các hình ảnh hoặc video trong thời gian thực. Dưới đây là các mục đích chính của hệ thống:

Hỗ trợ trong lái xe tự động: Hệ thống có thể được tích hợp vào các hệ thống lái xe tự động để cung cấp thông tin về môi trường xung quanh và giúp xe tự động theo dõi và duy trì đường chân trời khi di chuyển.

Ứng dụng trong công nghiệp và sản xuất: Trong lĩnh vực công nghiệp, hệ thống có thể được sử dụng để giám sát quá trình sản xuất và vận hành, đảm bảo rằng các robot và máy móc di chuyển an toàn trong môi trường làm việc.

Phát triển và nghiên cứu trong lĩnh vực thị giác máy tính: Hệ thống cung cấp một nền tảng cho các nhà nghiên cứu và phát triển để thử nghiệm và phát triển các thuật toán và phương pháp phát hiện đường chân trời trong thời gian thực.

**Yêu cầu hệ thống**

Thời gian thực: Hệ thống phải hoạt động trong thời gian thực, tức là phải xử lý dữ liệu video và phát hiện đường chân trời một cách nhanh chóng và hiệu quả, đảm bảo rằng thông tin được cung cấp gần như ngay lập tức.

Độ chính xác: Hệ thống cần có khả năng phát hiện đường chân trời một cách chính xác và đáng tin cậy trong nhiều điều kiện ánh sáng và môi trường khác nhau, bao gồm cả trong điều kiện ánh sáng yếu hoặc nền đất phức tạp.

Tính di động: Hệ thống cần có khả năng hoạt động trên các thiết bị di động và tích hợp vào các hệ thống phần cứng như camera an ninh, thiết bị giám sát và xe tự hành.

Giao diện người dùng thân thiện: Hệ thống cần có giao diện người dùng dễ sử dụng và trực quan, cho phép người dùng tương tác và cấu hình các thiết lập của hệ thống một cách dễ dàng.

**Cách ước lượng đường chân trời .**

Có nhiều cách để ước lượng đường chân trời :

Sử dụng các mô hình học máy : Thu thập các dữ liệu về bầu trời và mặt đất, mặt biển, núi sau đó chọn một mô hình máy học để phát hiên đường chân trời.

Sử dụng Clustering: Sử dụng các phương pháp phân cụm như K-means để phân tách các vùng bầu trời và đất trong ảnh. Xác định ranh giới giữa các vùng có thể giúp ước lượng đường chân trời.

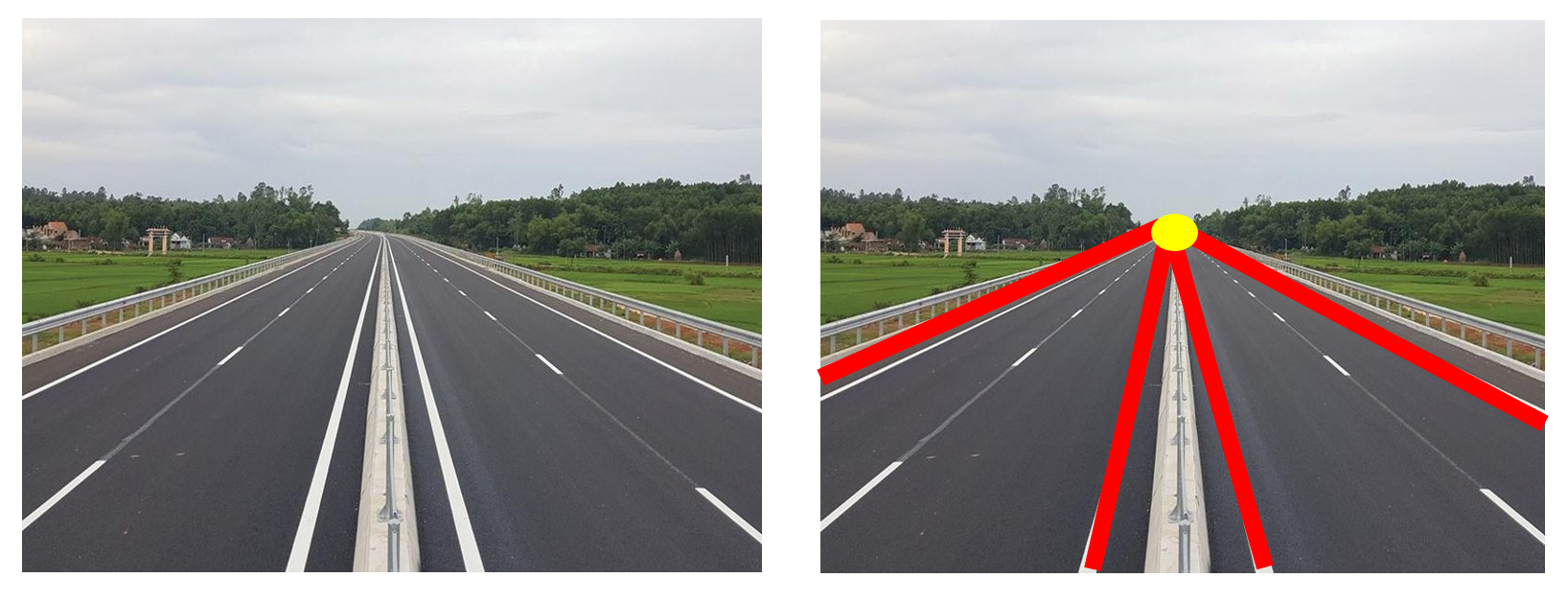
Phương pháp Hough Transform: Hough Transform để phát hiện các đường thẳng trong ảnh đã được phát hiện biên cạnh. Và dùng các cạnh đó để xử lí và tìm ra đường chân trời

Ở đây tôi sẽ sử dụng phương pháp Hough Tranform.

Như đã phân tích ở trên phương pháp Hough Transform là phát hiện các đường thẳng và xử lí nó.

Áp dụng vào bài toán ở đề tài này chúng ta sẽ lấy bối cảnh lái xe ô tô trên đường cao tốc. Khi lái xe chúng ta sẽ có một máy quay hành trình đặt ở trên giữa kính xe ô tô. Máy quay đó sẽ thu được các khung ảnh hay gọi là frame. Mục đích chính của tôi sẽ dựa vào các đường thẳng phát hiện được dựa trên vạch kẻ đường và tìm điểm giao của hai đường thẳng đó, tại đó sẽ là đường chân trời.

Để dể dàng kiểm thử và kiểm tra kết quả của chương trình, tôi sẽ ước lượng đường chân trời trong 1 video được quay trên đường cao tốc thay vì camera hành trình. Nếu ước lượng trên video có kết quả tốt thì kết quả trên camera hành trình cũng sẽ tương tự vì suy cùng cũng là xử lí trên từng frame ảnh.



Ở hình trên các vạch màu đỏ là các đoạn thẳng phát hiện được nhờ các vạch kẻ đường, chấm vàng là điểm giao nhau của các đường thẳng đó từ đó suy ra đường chân trời nằm xấp xỉ ở vị trí đó.

Phương pháp Hough Transfrom

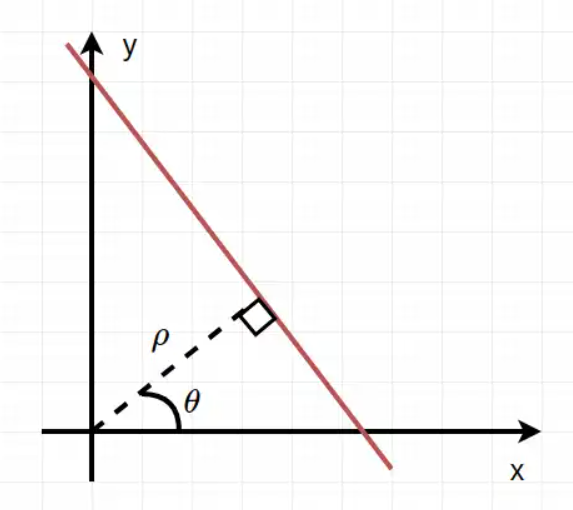
Hough Transform là thuật toán phát hiện đường thẳng khá hiệu quả trong xử lý ảnh. Ý tưởng chung của việc phát hiện đường thẳng trong thuật toán này là tạo mapping từ không gian ảnh (A) sang một không gian mới (B) mà mỗi đường thẳng trong không gian (A) sẽ ứng với một điểm trong không gian (B).

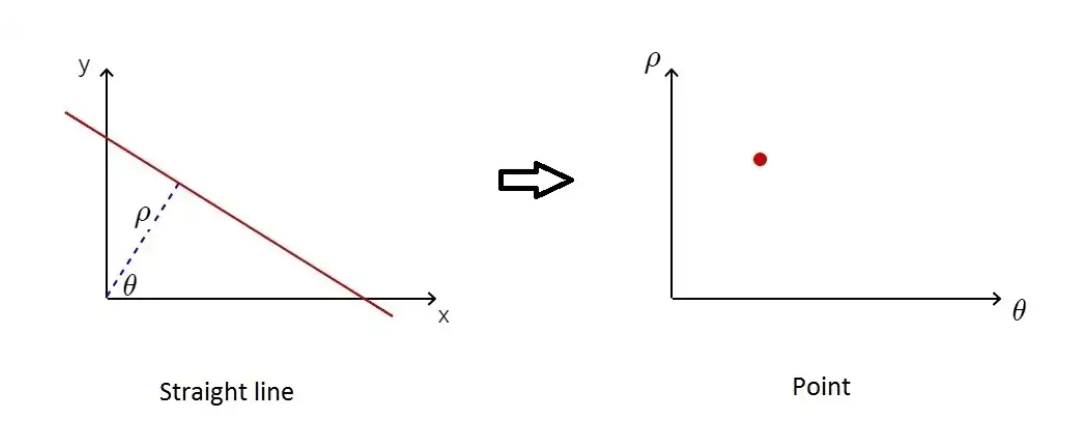
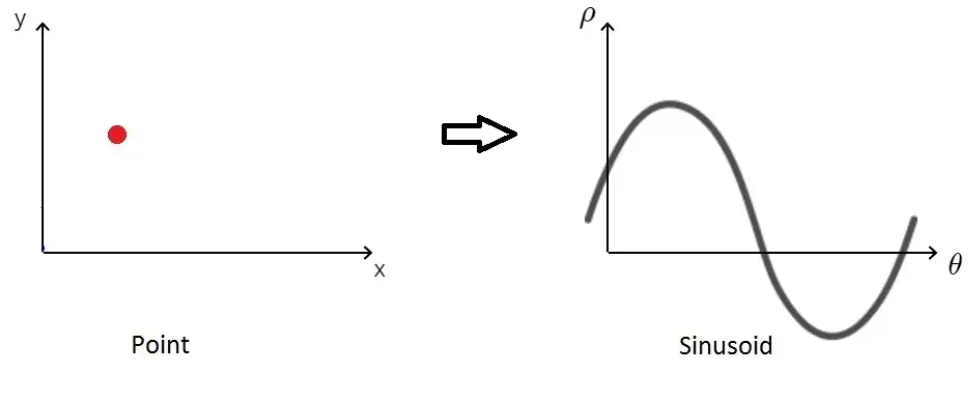
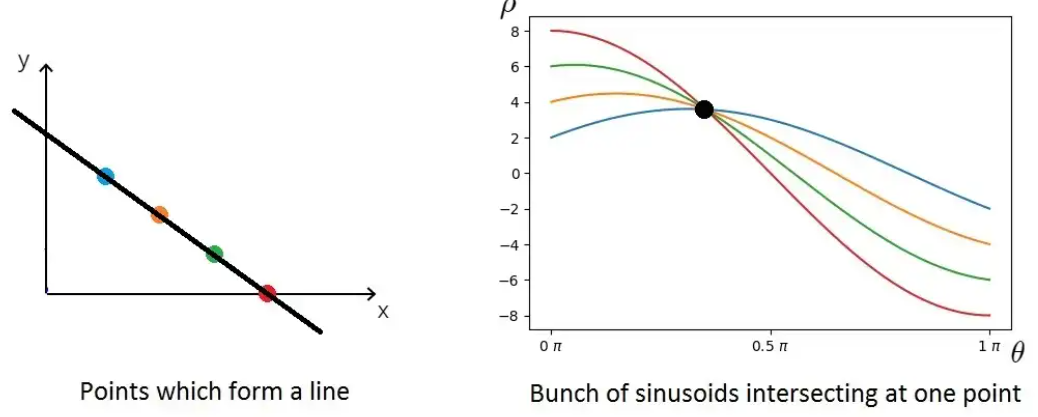
Phương trình đường thẳng cơ bản sẽ được biểu diễn theo 2 tham số a và b như sau:

� = � � + � y=ax+b

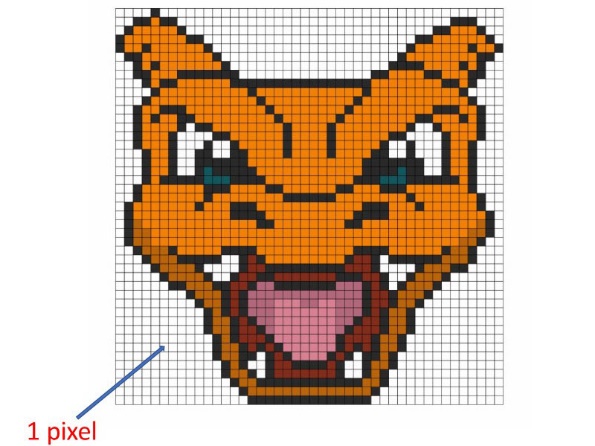
Tuy nhiên, với cách biểu diễn này, giá trị của góc nghiêng � a trải dài từ – ∞ –∞ đến + ∞ +∞. Có thể lấy ví dụ, để có được phương trình đường Oy ( � = 0 x=0) thì � a phải tiến tới ∞ ∞. Thuật toán Hough Transform yêu cầu các giá trị a, b nằm trong một khoảng xác định (hay bị chặn trên dưới), ta phải sử dụng hệ tọa độ cực để biểu diễn phương trình đường thẳng. Cách biểu diễn này cũng nằm trong chương trình toán trung học: � = � cos ⁡ ( � ) + � sin ⁡ ( � ) 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.

  
  
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.

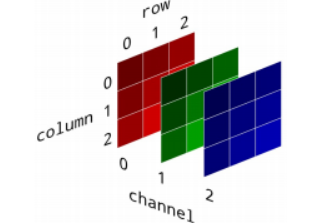
   
Từ một điểm trong không gian ảnh, chúng ta lại có được một hình sin trong không gian Hough:  
  
Các điểm nằm trên cùng một đường thẳng lại có biểu diễn là các hình sin giao nhau tại một điểm trong không gian Hough. Đây là nơi xuất phát ý tưởng của thuật toán Hough Transform. Chúng ta sẽ dựa vào các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

**Cấu trúc của 1 bức ảnh**

Ảnh được tạo thành từ một tập hợp các điểm ảnh, hay còn gọi là pixel. Mỗi pixel thường được biểu diễn dưới dạng một ô vuông nhỏ trên màn hình, và mỗi pixel này chứa thông tin về màu sắc và vị trí trên bức ảnh. Khi bạn kết hợp nhiều pixel lại với nhau theo cấu trúc và màu sắc khác nhau, bạn sẽ tạo ra một hình ảnh toàn vẹn.

Cấu trúc dữ liệu hình ảnh trong OpenCV được tổ chức dưới dạng một ma trận (số học) 3 chiều. Thứ tự các chiều của ma trận được sắp xếp theo chiều cao, chiều rộng và kênh màu (tiếng Anh: height, width, channel). Mỗi phần tử trong ma trận này có kiểu dữ liệu là số nguyên (0-255) hoặc số thực (0-1) mô tả giá trị của mức sáng (tiếng Anh: intensity).

Trong Xử Lý Ảnh, mức sáng của ảnh chạy trong miền giá trị 0-255. Màu đen "tuyệt đối" mang giá trị 0, màu trắng "tinh khôi" mang giá trị 255. Màu xám đậm sẽ có giá trị lệch về 0 nhiều, và màu xám nhạt lệch về phía 255. Do đó, đối với ảnh xám ta chỉ cần một kênh màu để mô tả và lưu trữ.

Ảnh màu được biểu diễn bằng cách pha ba màu cơ bản là Đỏ (Red), Xanh Lá (Green) và Xanh Dương (Blue) - còn gọi là hệ màu RGB. Ở mỗi kênh màu đỏ hoặc xanh dương hoặc xanh lá được biểu diễn mức sáng trong miền giá trị 0-255. Một bộ giá trị 3 kênh RGB (trong lập trình gọi là tuple) biểu diễn một sắc màu. Như vậy, tổng cộng ta sẽ có tổ hợp của 256 x 256 x 256 = 16,777,216 - 16 triệu màu

**Các phần mềm, công cụ và thư viện hỗ trợ**

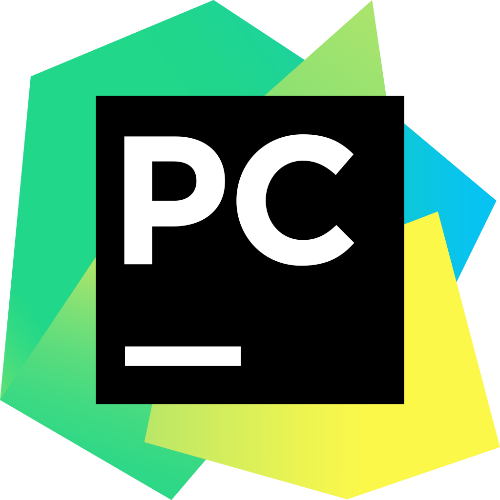
Ngôn ngữ code

Python

Python là một ngôn ngữ lập trình thông dịch, đa mục đích và dễ đọc. Nó được phát triển bởi Guido van Rossum và được công bố lần đầu tiên vào năm 1991. Python nổi tiếng với cú pháp đơn giản, dễ hiểu và linh hoạt, làm cho nó trở thành một trong những ngôn ngữ lập trình phổ biến nhất trên thế giới.

Lí do tôi chọn python là ngôn ngữ code cho đề tài bởi vì python hỗ trợ nhiều thư viện thuận tiện cho việc xử lí ảnh và video.  
  
**Công cụ hỗ trợ**  
Pycharm

PyCharm là một môi trường phát triển tích hợp (IDE - Integrated Development Environment) được JetBrains phát triển cho ngôn ngữ lập trình Python. Nó cung cấp một loạt các tính năng mạnh mẽ giúp lập trình viên Python phát triển các dự án phần mềm một cách nhanh chóng và hiệu quả.

**Các thư viện cần thiết**

tkinter: Đây là thư viện tiêu chuẩn của Python được sử dụng để tạo giao diện người dùng đồ họa. Tkinter cung cấp các widget và phương thức để tạo và quản lý các thành phần giao diện như cửa sổ, nút bấm, hộp văn bản, và nhiều hơn nữa.

filedialog: Đây là một module trong thư viện tkinter, cung cấp các hộp thoại để chọn và mở tệp tin từ hệ thống tệp của người dùng. Cụ thể trong đoạn mã, filedialog được sử dụng để mở hộp thoại chọn tệp để chọn ảnh hoặc video.

cv2 (OpenCV): OpenCV (Open Source Computer Vision Library) là một thư viện mã nguồn mở cung cấp các hàm và công cụ để xử lý ảnh và video. OpenCV có thể được sử dụng để đọc, ghi, xử lý, và phân tích ảnh và video, bao gồm cả nhận dạng đối tượng, phát hiện biên cạnh, và nhiều tác vụ khác trong thị giác máy tính.

numpy: NumPy là một thư viện Python cung cấp cấu trúc dữ liệu mảng nhiều chiều và các hàm toán học và logic để làm việc với mảng này. NumPy là một trong những thư viện quan trọng nhất trong khoa học dữ liệu và tính toán số học trong Python.

PIL.Image và PIL.ImageTk: Đây là các phần của thư viện Pillow (Python Imaging Library), một thư viện được sử dụng để làm việc với hình ảnh trong Python. PIL.Image được sử dụng để mở và xử lý hình ảnh, trong khi PIL.ImageTk cung cấp các công cụ để hiển thị hình ảnh trên giao diện người dùng tkinter.

**Triển khai chương trình**

**Tiền xử lí video**

Cấu trúc của một video được xây dựng từ nhiều frame ảnh, được phát liên tiếp nhau để tạo ra hiệu ứng chuyển động. Mỗi frame là một hình ảnh tĩnh, và khi các frame này được phát liên tiếp nhau ở tốc độ nhanh, sẽ tạo ra cảm giác chuyển động. Nên để xử lí tôi sẽ lấy từng frame trong video

**Cân bằng độ sáng ảnh**

Với 1 số khung hình, độ sáng của ảnh có thể sẽ quá tối gây ra khó phát hiện được các cạnh trong ảnh, đặc biệt là lane đường. Để giải quyết vấn đề đó, tôi tính trung bình độ sáng của ảnh bằng cách cộng các giá trị độ sáng của từng pixel trong ảnh và chia cho số lượng pixel của ảnh.

|  |  |  |
| --- | --- | --- |
| 10 | 20 | 30 |
| 40 | 30 | 60 |
| 70 | 80 | 20 |

Ví dụ : Ở trên là 1 bức ảnh, có 9 pixel, mỗi pixel có một giá trị mức sáng riêng. Tôi tính bằng cách cộng 9 giá trị đó lại và chia cho 9.

giá trị độ sáng = (10+20+30+40+30+60+70+80+20)/9 = 40

Vậy ảnh có độ sáng trung bình là 40.

Để tìm một mức sáng để làm ngưỡng đo, tôi tìm 1 bức ảnh có độ sáng ổn sau đó tính trung bình độ sáng ảnh đó, sau đó chọn kết quả đó làm ngưỡng.

**Phát hiện cạnh bằng thuật toán Canny**

Phát hiện cạnh Canny (Canny edge detector) là một thuật toán bao gồm nhiều giai đoạn để phát hiện một loạt các cạnh trong hình ảnh. Nó được phát triển bởi John F. Canny vào năm 1986. Canny cũng đưa ra một lý thuyết tính toán về phát hiện cạnh giải thích tại sao kỹ thuật này hoạt động.

Thuật toán Canny dựa trên ảnh xám. Do đó, điều kiện tiên quyết là phải chuyển hình ảnh sang thang độ xám trước khi thực hiện.

**Chuyển đổi frame sang ảnh xám**

Ảnh màu có ba kênh màu (đỏ, xanh lá cây, xanh lam) trong khi ảnh xám chỉ có một kênh màu duy nhất. Việc chuyển đổi sang ảnh xám giúp giảm bớt dữ liệu, làm cho việc xử lý nhanh chóng hơn và tiết kiệm bộ nhớ máy tính. Tính toán đơn giản hơn khi tìm cạnh trên ảnh màu, việc xác định đường biên sẽ phức tạp hơn do phải xem xét các kênh màu khác nhau. Trong ảnh xám, chỉ có một giá trị màu cho mỗi pixel, điều này làm cho việc tính toán cạnh đơn giản hơn và nhanh chóng hơn.

**Làm mờ hình để giảm nhiểu**

Việc làm mờ ảnh trong thuật toán Canny Edge Detection có một số lợi ích:

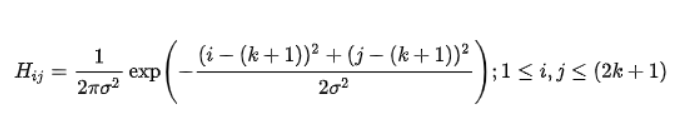
Giảm nhiễu: Trước khi tìm cạnh, làm mờ ảnh giúp giảm nhiễu. Các nhiễu nhỏ thường xuất hiện trong hình ảnh có thể tạo ra các cạnh giả mạo nếu không được loại bỏ trước khi áp dụng thuật toán tìm cạnh. Làm mờ giúp làm giảm ảnh hưởng của nhiễu đối với quá trình phát hiện cạnh, làm cho kết quả tìm cạnh chính xác hơn.

Loại bỏ chi tiết không cần thiết: Đôi khi, các chi tiết không cần thiết hoặc nhỏ có thể gây ra các cạnh giả mạo. Bằng cách làm mờ ảnh trước khi tìm cạnh, các chi tiết này có thể được làm mờ hoặc loại bỏ, làm tăng độ chính xác của việc phát hiện cạnh

Làm cho cạnh trở nên liền mạch hơn: Làm mờ ảnh giúp làm cho các cạnh trong ảnh trở nên liền mạch hơn, giúp giảm bớt các đỉnh và suy biến không mong muốn trong kết quả tìm cạnh.

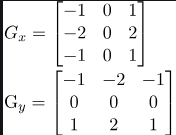
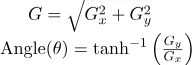
Giảm độ nhạy cảm đối với nhiễu: Bằng cách giảm đi độ sắc nét của ảnh thông qua việc làm mờ, thuật toán Canny trở nên ít nhạy cảm hơn với nhiễu, điều này giúp cải thiện chất lượng của các cạnh được phát hiện.

Một cách để loại bỏ nhiễu là áp dụng Gaussian Filter giúp làm mịn ảnh. Để làm như vậy, kỹ thuật tích chập hình ảnh được áp dụng với Gaussian Kernel (kích thước 3×3, 5×5, 7×7, v.v.). Kích thước kernel phụ thuộc vào hiệu ứng làm mờ mong muốn. Về cơ bản, kernel càng nhỏ, hiệu ứng mờ càng ít. Ví dụ dưới đây sử dụng 5 x 5 Gaussian kernel.

Phương trình cho một kernel bộ lọc Gaussian có kích thước (2k + 1) × (2k + 1):

**Tìm cường độ gradient**

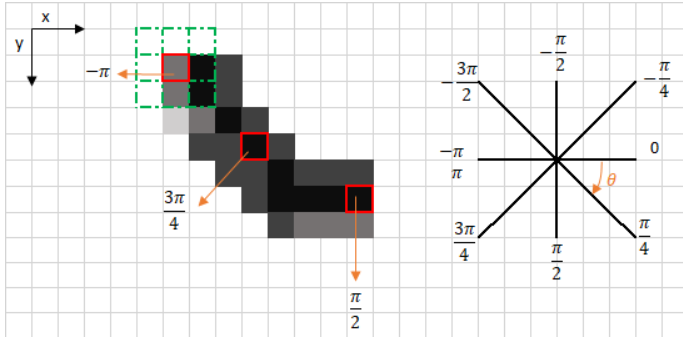
Trong bước này, gradient cường độ của hình ảnh được tìm thấy, giúp xác định vị trí các vùng thay đổi cường độ đột ngột tương ứng với các cạnh trong ảnh. Kỹ thuật này, được gọi là thuật toán Sobel, được sử dụng để tính đạo hàm bậc nhất của hình ảnh theo cả hướng ngang (Gx) và dọc (Gy). Độ dốc ngang (Gx) và độ dốc dọc (Gy) được cho bởi các phép toán tích chập sau:

  
Các phương trình sau đây có thể được sử dụng để tính cường độ (G) và hướng (θ) của gradient tại mỗi pixel:

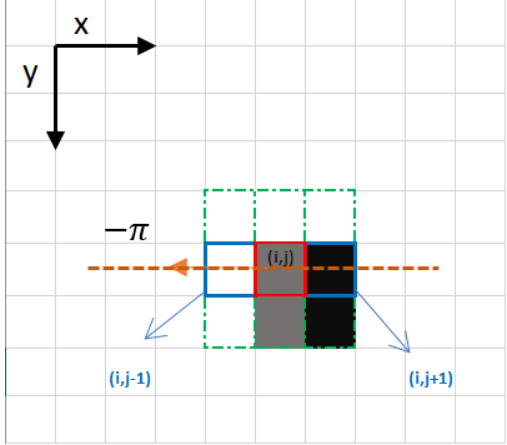
**Non-maximum suppression**

Tốt nhất, hình ảnh cuối cùng nên có các cạnh mỏng. Vì vậy, ta phải thực hiện thuật toán Non-maximum suppression để làm mỏng chúng.

Nguyên tắc rất đơn giản: thuật toán đi qua tất cả các điểm trên ma trận cường độ gradient và tìm các pixel có giá trị lớn nhất theo các hướng cạnh.

Hãy lấy một ví dụ dễ hiểu:

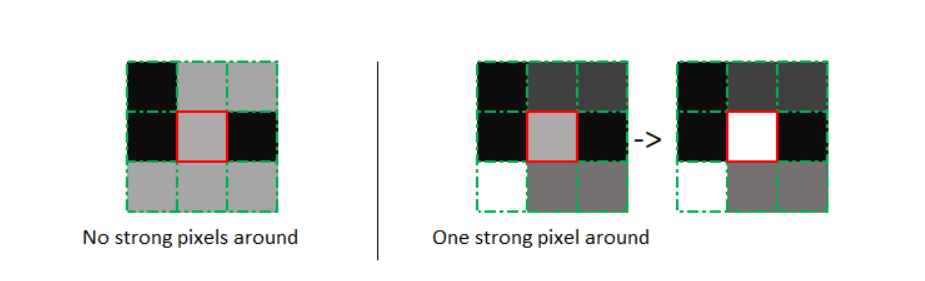
Hộp màu đỏ ở góc trên bên trái đại diện cho một pixel cường độ của ma trận Cường độ Gradient đang được xử lý. Hướng gradient tương ứng được biểu diễn bằng mũi tên màu cam với góc -pi radian (+/- 180 độ).

Hướng gradient là đường chấm màu cam (nằm ngang từ trái sang phải). Mục đích của thuật toán là để kiểm tra xem các pixel trên cùng một hướng có cường độ cao hơn hay thấp hơn các pixel đang được xử lý. Trong ví dụ trên, pixel (i, j) đang được xử lý và các pixel trên cùng một hướng được đánh dấu bằng màu xanh lam (i, j-1) và (i, j + 1). Nếu một trong hai pixel đó có cường độ cao hơn pixel đang được xử lý, thì chỉ có một pixel có cường độ cao hơn được giữ lại. Pixel (i, j-1) có vẻ sáng hơn, vì nó có màu trắng (giá trị 255). Do đó, giá trị cường độ của pixel hiện tại (i, j) được đặt thành 0. Nếu không có pixel nào ở hướng cạnh có giá trị cường độ cao hơn thì giá trị của pixel hiện tại được giữ nguyên.

**Ngưỡng hóa (Thresholding)**

Các pixel sau quá trình loại bỏ non-maximum sẽ được so sánh với hai ngưỡng: một ngưỡng dưới (low threshold) và một ngưỡng trên (high threshold). Các pixel có độ lớn gradient lớn hơn ngưỡng trên sẽ được coi là biên cạnh chắc chắn, trong khi các pixel có độ lớn gradient nhỏ hơn ngưỡng dưới sẽ bị loại bỏ. Các pixel nằm giữa hai ngưỡng sẽ được xem xét nếu chúng được kết nối với các biên cạnh chắc chắn.

Theo dõi cạnh theo độ trễ (Edge Tracking by Hysteresis) Dựa trên kết quả sau khi áp dụng ngưỡng, Hysteresis Tracking chuyển đổi các pixel yếu thành mạnh, khi và chỉ khi ít nhất một trong các pixel xung quanh pixel đang xét là pixel mạnh, các pixel yếu còn lại bị loại bỏ, như được mô tả bên dưới:

**  
Ước lượng đường chân trời**

Tìm các đoạn thẳng biểu diễn bởi vạch kẻ đường

Để tìm 2 lane bên trái và bên phải tôi thực hiện , chúng ta thiết lập ngưỡng để xác định số lượng điểm cần thiết để coi một đoạn thẳng là một đoạn thẳng, độ dài tối thiểu của đoạn thẳng và khoảng cách tối đa giữa hai điểm trên đoạn thẳng để coi chúng là một đoạn thẳng liên tục. Sau khi tìm được các đoạn thẳng biểu diễn vạch kẻ đường, chúng được phân loại thành hai nhóm: lane bên trái và lane bên phải. Tôi tìm lane trái và phải dựa trên độ dốc của cạnh dựa trên công thức

Slope =

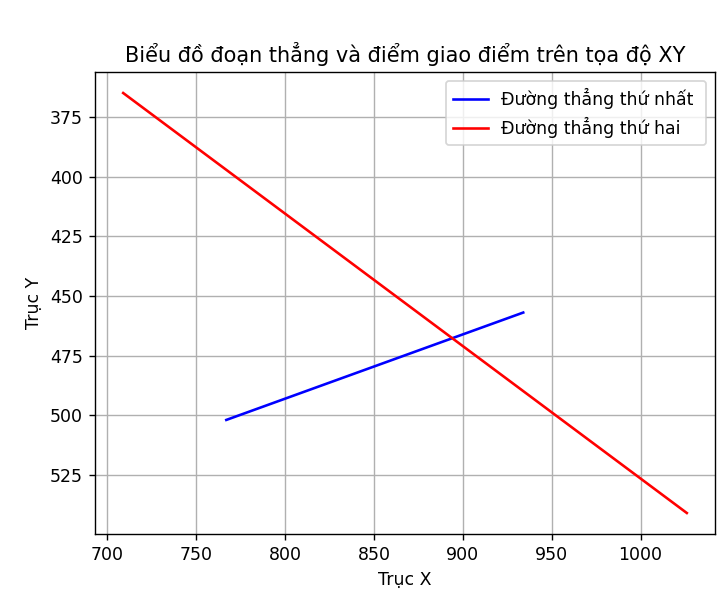
Nếu kết quả slope là dương thì đoạn thẳng thuộc lane bên phải, và nếu có dộ dốc âm thì đoạn thẳng thuộc lane bên trái

Ví dụ: Để tính slope của đoạn thẳng thứ nhất được định nghĩa bởi (, và (, (934, 457) và đoạn thẳng thứ hai được định nghĩa bởi (, và (, (1026, 541) ta thay giá trị vào công thức:

Slope =

Slope

Đoạn dốc thẳng này thuộc dốc bên phải



Tìm giao điểm của hai vạch kẻ đường

Để tính toán điểm giao của hai lane, tôi sử dụng phương pháp giao điểm của hai đường thẳng trong hệ tọa độ Euclide. Phương pháp này dựa trên việc sử dụng độ dốc và tọa độ của các đoạn thẳng biểu diễn lane để tính toán tọa độ của điểm giao.

Vẽ giao điểm lên ảnh gốc

Sau khi tìm được điểm giao tôi sẽ vẽ một đường thẳng tượng trưng cho đường chân trời vừa được ước lượng

Kết luận

Kết quả đạt được:

Chương trình cơ bản có thể ước lượng được đường chân trời trong video

Hiểu được cơ bản về xử lí ảnh

Nâng cao khả năng tìm kiếm thông tin trên mạng

Hạn chế:

Chưa xử lí được trong tình huống thiếu một hoặc hai lane đường, video thiếu sáng. Gây ra việc ước lượng có độ lệch cao so với thực tế và có trường hợp tệ hơn là không ước lượng được.

Chưa tối ưu xử lí được các cạnh gây nhiễu gây ra việc ước lượng đường chân trời bị nhiễu không đáng có.

Chưa xử lí được tình huống có các vật cản đè lên vạch kẻ đường, gây ra chương trình không nhận diện được vạch kẻ đường từ đó ước lượng đường chân trời không chính xác.

Các hướng phát triển:

Hoàn thiện các phần ở phần hạn chế.

Tích hợp trí tuệ nhân tạo, máy học vào chương trình để có thể ước lượng được tốt hơn

Ứng dụng vào việc phát triển chương trình cho các phương tiện giao thông như xác định khoảng cách với xe phía trước.