

Ứng dụng Xử lý ảnh và Machine Learning để nhận dạng và phân loại xe 2 bánh và 4 bánh trong video

Nguyễn Huỳnh Đức* Nguyễn Hồng Hưng*, Nguyễn Hoàng Phúc* Văn Đình Triều* Lê Tuấn Vũ*

* Lớp Viễn thông - PFIEV - 2014

Đại học Bách Khoa Thành Phố Hồ Chí Minh

Môn học: Xử lý số hình ảnh và âm thanh - GVHD: ThS. Đặng Nguyên Châu

Email: {1510797, 1411608, 1412957, 1414177, 1414741}@hcmut.edu.vn

Tóm tắt nội dung—Trong bài báo này, nhóm ứng dụng những kiến thức đã học về xử lý ảnh, cũng như tìm hiểu về một số giải thuật phân lớp để loại các vật thể. Nhóm áp dụng kỹ thuật trích xuất đặc trưng HOG (Histogram of Oriented Gradient) và phương pháp phân lớp (classification) SVM (Support Vector Machine) để nhận dạng các vật thể trong một video. Nhóm đã tiến hành thực nghiệm nhận dạng và phân loại xe 2 bánh và xe 4 bánh, kết quả đạt được có độ chính xác khá tốt.

Index Terms—Digital Image Processing, Classification, Histogram of Oriented Gradient, Support Vector Machine, Vehicle Detection.

I. GIỚI THIỆU

Phân loại ảnh là một bài toán đã và đang thu hút được sự quan tâm của các nhà nghiên cứu và phát triển, được ứng dụng rộng rãi nhiều ứng dụng hữu ích như: tìm kiếm ảnh, nhận dạng, theo dõi và phát hiện đối tượng... Trong giám sát đối tượng chuyển động từ video chẳng hạn như giám sát phương tiện giao thông, thì phân loại ảnh là bài toán kế tiếp sau bài toán phát hiện đối tượng chuyển động. Khi đó bài toán đối sánh ảnh sẽ quy về bài toán so sánh các đặc trưng trích chọn. Các đặc trưng cho phép biểu diễn ảnh đã được nghiên cứu bao gồm đường biên vùng ảnh, điểm ảnh đặc trưng, lược đồ xám (histogram), lược đồ vector gradient (Histogram of Oriented Gradient)...

Có hai vấn đề cơ bản thường đặt ra trong bài toán phân loại: i) làm sao có thể biểu diễn thông tin một cách hiệu quả nhằm thực hiện việc đối sánh hai ảnh nhanh nhất có thể; ii) làm thế nào để giải pháp đối sánh vẫn hoạt động hiệu quả khi có sự thay đổi của môi trường: nhiễu trong quá trình thu nhận ảnh, sự thay đổi về ánh sáng, sự che khuất,...

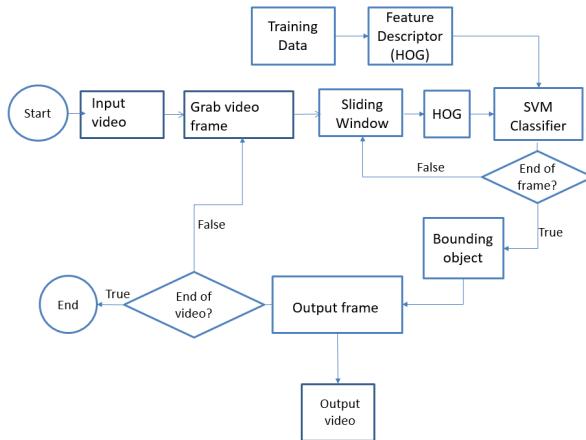
Cùng với sự phát triển của các thuật toán phân loại, các ứng dụng nhận diện có độ chính xác ngày càng gia

tăng. Dù không thể bằng con người nhưng kết quả của các phần mềm nhận diện cho thấy khá khả quan với những sai số có thể chấp nhận. Người ta đã ứng dụng các thuật toán phân loại vào rất nhiều đối tượng trong cuộc sống, cụ thể là nhận diện các loại phương tiện giao thông. Với mục đích tìm hiểu về các thuật toán machine learning và phát triển một phần mềm phân loại và nhận diện, nhóm muốn được thử sức về đề tài “Phân loại các loại trong một video” bằng phương pháp trích đặc trưng Histogram of Oriented Gradient (HOG) và phân loại bằng thuật toán Support Vector Machine (SVM). Lý do nhóm sử dụng các kỹ thuật này là nhờ những ưu điểm mà chúng mang lại: HOG giúp mô tả vật thể trong một bức ảnh và SVM giúp tìm ra được mặt phán chia tối ưu trong không gian nhiều chiều. Các báo cáo về SVM trên thế giới cho thấy, SVM cho kết quả phân loại tốt hơn Perceptron Learning Algorithm nhờ mặt phán chia tối ưu.

Trong các phần tiếp theo của bài báo này, nhóm xin trình bày phương pháp tiếp cận của nhóm. Trong phần II và phần III, nhóm lần lượt trình bày sơ lược giải thuật và phân tích các bước thực hiện. Trong phần IV, nhóm sẽ kiểm chứng các kết quả phân tích bằng các kết quả mô phỏng trên Python. Cuối cùng, nhóm kết luận bài báo trong phần V.

II. SƠ LƯỢC GIẢI THUẬT

Chúng ta xem xét sơ đồ nguyên lý chung được mô tả như trong Hình 1, với Input là video đầu vào, ta tách từng frame video ra và xử lý trên từng frame đó. Sau khi thực hiện xong, các frame đã qua xử lý sẽ được lưu lại tại bộ nhớ đệm để tạo thành video mà trong đó các xe 2 bánh và 4 bánh đã được nhận dạng. Chúng ta sẽ phân tích kĩ hơn ở các khía cạnh trích xuất đặc trưng HOG



Hình 1: Sơ đồ nguyên lý

khối SVM classifier, sliding window, đây là những khối chính của cả sơ đồ.

III. PHÂN TÍCH CÁC BƯỚC THỰC HIỆN

A. Histogram of Oriented Gradient

HOG(histogram of oriented gradients) là một feature descriptor được sử dụng trong computer vision và xử lý hình ảnh, dùng để detect một đối tượng.[1] [2]

Các khái niệm về HOG được nêu ra từ năm 1986 tuy nhiên cho đến năm 2005 HOG mới được sử dụng rộng rãi sau khi Navneet Dalal và Bill Triggs công bố những bổ sung về HOG. Hog tương tự như các biểu đồ edge orientation, scale-invariant feature transform descriptors(như sift, surf..), shape contexts nhưng HOG được tính toán trên một lưới dày đặc các cell và chuẩn hóa sự tương phản giữa các block để nâng cao độ chính xác. Hog được sử dụng chủ yếu để trích xuất đặc trưng một object trong ảnh Bài toán tính toán Hog thường gồm 5 bước:

- Chuẩn hóa hình ảnh trước khi xử lý
- Tính toán gradient theo cả hướng x và y
- Lấy phiếu bầu cùng trọng số trong các cell
- Chuẩn hóa các block
- Thu thập tất cả các biểu đồ cường độ gradient định hướng để tạo ra feature vector cuối cùng

1) *Chuẩn hóa hình ảnh trước khi xử lý*: Bước chuẩn hóa này hoàn toàn không bắt buộc, nhưng trong một số trường hợp, bước này có thể cải thiện hiệu suất của bộ mô tả HOG. Đối với đề tài này, các hình ảnh đã được đưa về cùng độ phân giải là 64x64 pixels.

2) *Tính toán gradient theo 2 hướng* : Gradient của một bức hình là 1 vector với 2 thành phần:

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Với g_x, g_y lần lượt là gradient theo phương x và y. Để tính giá trị gradient, ta tính tích chập $G = I \otimes D$, với I là hình ảnh đầu vào, D là mặt nạ gradient, hay còn gọi là kernel. Hình dưới đây là một số kernel hay sử dụng.



(a) Sobel kernel size 1

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

(b) Sobel kernel size 3

$$G_x = \begin{bmatrix} -3 & 0 & +3 \\ -10 & 0 & +10 \\ -3 & 0 & +3 \end{bmatrix} \quad G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ +3 & +10 & +3 \end{bmatrix}$$

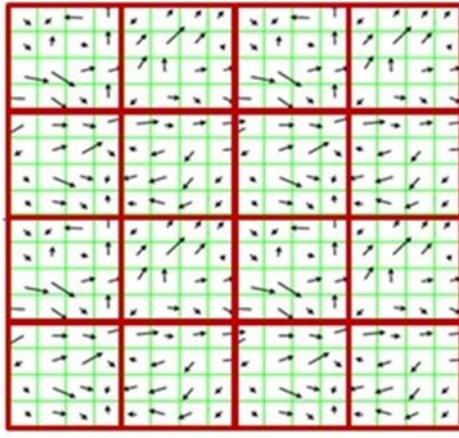
(c) scharr kernel

Hình 2: Các kernel thông dụng

Sau khi có các giá trị gradient theo phương x và y, ta có thể biểu diễn thành một vector với cường độ là $g = \sqrt{g_x^2 + g_y^2}$, hướng $\theta = \arctan \left(\frac{g_y}{g_x} \right)$

Dưới đây là hình mô tả các vector gradient của một bức hình 16x16 pixels, trong đó các ô vuông xanh tương ứng cho các điểm ảnh, các mũi tên đen là các vector gradient.

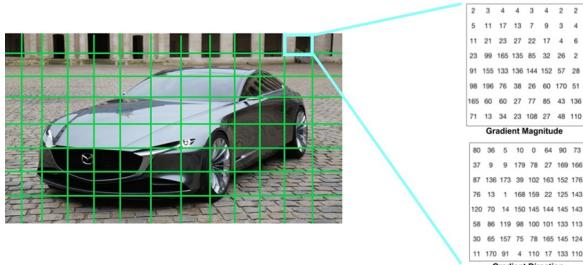
3) *Lấy phiếu bầu cùng trọng số trong các cell*: Một trong những lý do quan trọng sử dụng HOG đó là chúng ta cần tìm ra đặc trưng của bức ảnh (feature descriptor)



Hình 3: Minh họa gradient vector của một bức ảnh 16x16 pixels

một cách súc tích nhất. Do đó thay vì dùng đến 256 vector gradient (ứng với 256 điểm ảnh) để biểu thị một bức ảnh, ta có thể giảm số vector này xuống mà vẫn thể hiện được đặc trưng của bức ảnh đó. Trong bước này, bức ảnh trên sẽ được chia thành 16 ô vuông, tức 4x4 cells, và tương ứng sẽ chỉ còn 16 vector gradient. Công việc tiếp theo là ta sẽ lập histogram của gradient cho mỗi cell bằng phương pháp lấy phiếu bầu cùng trọng số trong một cell.

Ví dụ ta có hình dưới đây đã được chia thành các cell, mỗi cell có kích thước 8x8 pixels, sau khi thực hiện bước 2 ta có các giá trị vector gradient (gồm biên độ và hướng) của mỗi pixel.

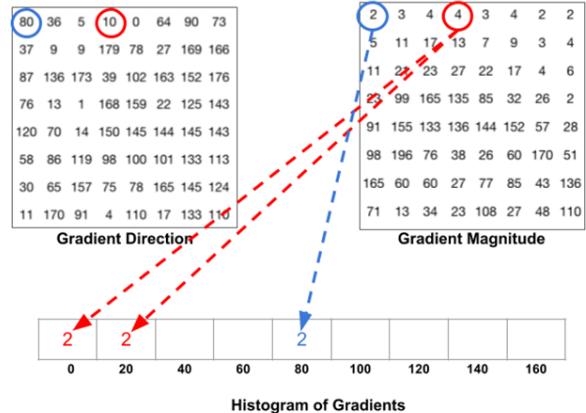


Hình 4: Hình được chia thành các cell 8x8 pixel

Với mỗi cell trong bức ảnh, ta cần xây dựng 1 biểu đồ cường độ gradient. Mỗi pixel sẽ được vote vào vào biểu đồ, trọng số của mỗi vote chính là cường độ gradient tại pixel đó. Cuối cùng, mỗi pixel đóng góp một phiếu bầu có trọng số vào biểu đồ - trọng lượng của phiếu chỉ đơn giản là cường độ gradient $|G|$ tại pixel đó. Lúc này, chúng ta có thể thu thập và ghép các biểu đồ này để tạo

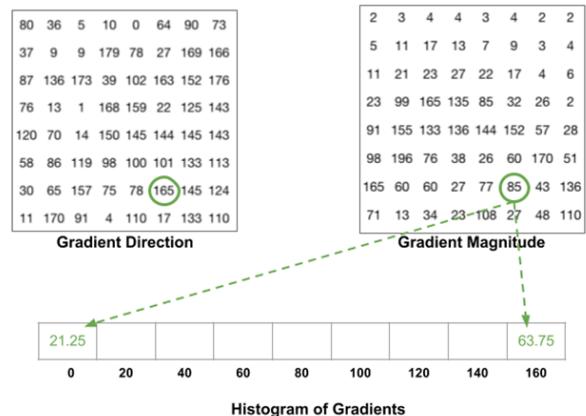
ra feature vector cuối cùng.

Dựa vào các số liệu này, ta sẽ tạo ra một histogram gồm có 9 mức, ứng với 9 hướng của vectơ gradient: 0, 20, 40 ... 160°. Hình dưới đây minh họa cho quá trình tính toán này.



Hình 5: Quá trình tạo histogram gradient bằng voting

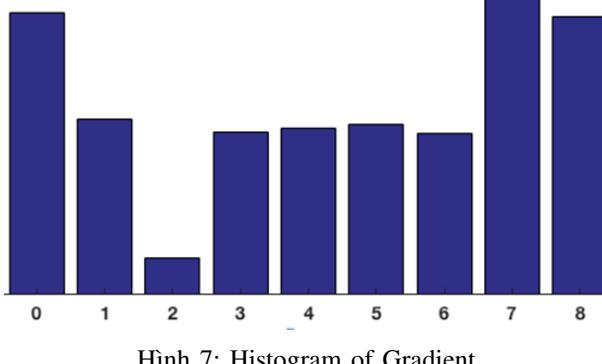
Ô màu xanh biểu thị cho vector gradient có biên độ là 2, góc 80°, ở trên thang 80° của histogram ta gán giá trị 2 vào. Đối với ô màu đỏ, vector gradient có biên độ 4, góc 10°, ở trên thang histogram không có giá trị này, vì vậy nó sẽ vote cho 2 thang gần nhất với nó, tức là thang 0° và 20°. Lưu ý tỉ lệ vote sẽ bằng 1:1 bởi vì giá trị góc của vector là 10°, cách đều 2 thang 0° và 20°.



Hình 6: Histogram gradient calculating

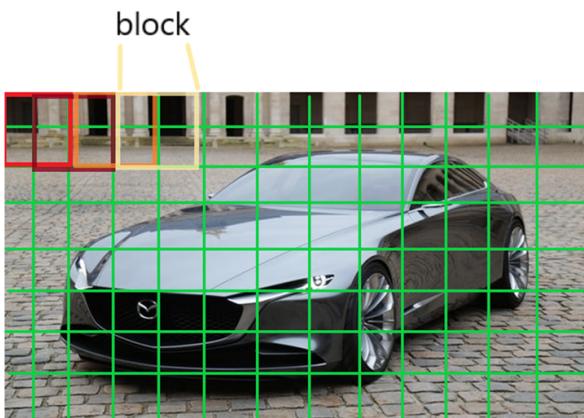
Một trường hợp khác đó là góc của vector nằm trong khoảng 160° - 180°, lúc này ta vẫn làm tương tự như trên, tuy nhiên thay vì vote cho thang 180°, ta sẽ vote cho thang 0°. Sau khi tổng hợp tất cả các điểm ảnh trong

1 cell, ta được histogram gradient có dạng như hình. Như vậy sau bước 3, số vectơ gradient đặc trưng cho 1 cell giảm từ 64 vector xuống còn 9 vector.



Hình 7: Histogram of Gradient

4) Chuẩn hóa các block: Ở bước trước, chúng ta đã tạo ra một biểu đồ dựa trên gradient của hình ảnh. Gradient của hình ảnh rất nhạy cảm với ánh sáng tổng thể. Nếu hình ảnh được làm cho tối hơn bằng cách chia tất cả các giá trị pixel cho 2, cường độ gradient sẽ thay đổi một nửa và do đó, các giá trị biểu đồ sẽ thay đổi một nửa. Một cách lý tưởng, chúng ta muốn các đặc trưng của một bức hình độc lập với sự thay đổi của độ sáng. Để làm được điều này, chúng ta phải chuẩn hóa các giá trị vector gradient.



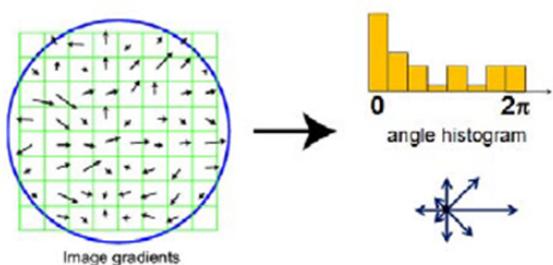
Hình 8: Chuẩn hóa các block

Ta xem các giá trị gradient của một cell như là 1 vector $1 \times n$ (n là số vectơ gradient), đối với trường hợp trên là vector 1×9 . Ta gom 4 cell (2×2) thành 1 block, và như vậy, ta có tổng cộng 36 vector gradient trong 1 block, được biểu diễn thành 1 array có kích thước 1×36 .

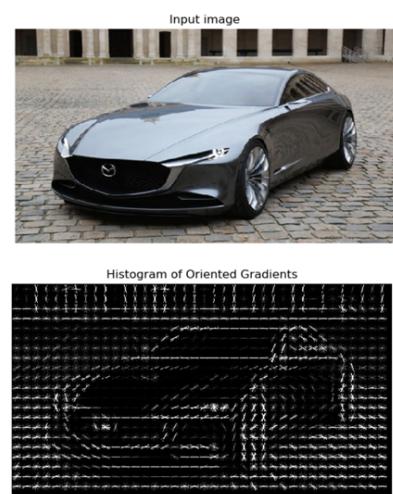
Sau đó chuẩn hóa array trên, với array $x = (x_1, x_2, x_3, \dots, x_n)$, công thức chuẩn hóa sẽ là:

$$x_{norm} = \frac{x}{\|x\|_2} = \frac{x}{\sqrt{x_1^2 + x_2^2 + x_3^2 + \dots + x_n^2}}$$

Tương tự, dịch block sang 1 cell (giống như sliding window) và tiếp tục chuẩn hóa các cell trong block đó. Sau khi chuẩn hóa xong, ta tổng hợp các vector gradient trong 1 block lại như hình:



Hình 9: Tổng hợp các vector gradient của các cell trong 1 block

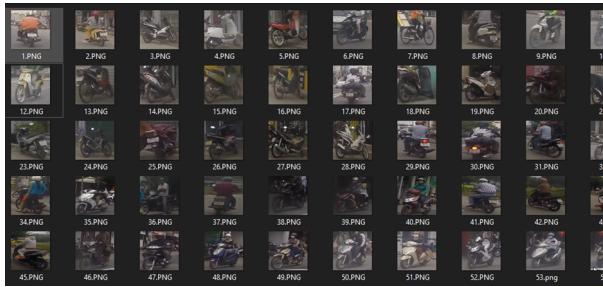


Hình 10: Kết quả trích xuất đặc trưng HOG

5) Thu thập tất cả các biểu đồ cường độ gradient định hướng để tạo ra feature vector cuối cùng: Feature Vector cuối cùng sẽ là tổng của các vector của mỗi block.

B. Bộ dữ liệu huấn luyện

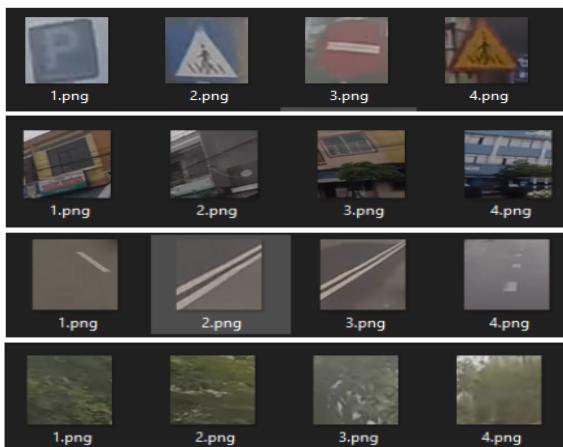
Bộ dữ liệu được xây dựng từ các ảnh về xe tham gia giao thông trên đường phố Việt Nam, mỗi ảnh có kích thước 64x64 pixels. Ngoài tập dữ liệu dành cho nhận diện, bộ dữ liệu về background và các đối tượng nền cũng cần được thu thập.



Hình 11: Tập dữ liệu mẫu cho xe máy



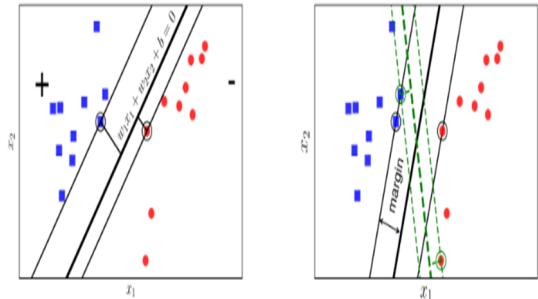
Hình 12: Mẫu dữ liệu cho xe ô tô



Hình 13: Mẫu dữ liệu nền để loại bỏ trong quá trình nhận diện

C. SVM (Support Vector Machine) Classifier

Ý tưởng chính của thuật toán Support Vector Machine (SVM) là tìm được mặt phẳng phân chia 2 lớp dữ liệu, sao cho cách đều 2 (hoặc nhiều) điểm dữ liệu gần nhau nhất của 2 lớp, khoảng cách từ 2 điểm gần nhất đến mặt phẳng phân chia gọi là margin (lề), mục tiêu là tìm được mặt phẳng chia sao cho margin là lớn nhất. [3]



Hình 14: Margin của hai lớp dữ liệu là bằng nhau và lớn nhất có thể

Như vậy bài toán sẽ được biểu diễn dưới dạng toán học như sau: Khoảng cách từ một điểm x_0 tới mặt phân chia $w^T x + b = 0$ được xác định bởi

$$\frac{|w^T x_0 + b|}{\|w\|_2}$$

Giả sử nhãn của mỗi điểm dữ liệu được xác định bởi

$$y_i = 1$$

hoặc

$$y_i = -1$$

(class 2) Với cặp dữ liệu (x_n, y_n) bất kỳ, khoảng cách từ điểm đó tới mặt phân chia là:

$$\frac{y_n (w^T x_n + b)}{\|w\|_2}$$

Điều này có thể dễ nhận thấy vì theo giả sử ở trên, y_n luôn cùng dấu với phía của x_n . Từ đó suy ra y_n cùng dấu với $w^T x_n + b$, và tử số luôn là 1 số không âm. Với mặt phân chia như trên, margin được tính là khoảng cách gần nhất từ 1 điểm tới mặt đó (bất kể điểm nào trong hai classes):

$$\text{margin} = \min_n \frac{y_n (w^T x_n + b)}{\|w\|_2}$$

Theo mục đích bài toán, chúng ta cần tìm w và b sao cho margin lớn nhất, tức là

$$\begin{aligned} (w, b) &= \arg \max \left\{ \min_n \frac{y_n (w^T x_n + b)}{\|w\|_2} \right\} \\ &= \arg \max_{w,b} \left\{ \frac{1}{\|w\|_2} \min_n y_n (w^T x_n + b) \right\} \quad (1) \end{aligned}$$

Không mất tính tổng quát, ta có thể giả sử $y_n (w^T x_n + b) = 1$ với những điểm nằm gần mặt phân chia nhất.

Như vậy, với mọi n , ta có: $y_n (w^T x_n + b) \geq 1$. Vậy bài toán tối ưu (1) có thể đưa về bài toán tối ưu có ràng buộc sau đây:

$$(w, b) = \arg \max \frac{1}{\|w\|_2}$$

subject to : $y_n (w^T x_n + b) \geq 1, \forall n = 1, 2, \dots, N$ (2)

Bằng 1 biến đổi đơn giản, ta có thể đưa bài toán này về bài toán dưới đây

$$\begin{aligned} (w, b) &= \arg \min_{w,b} \frac{1}{2} \|w\|_2^2 \\ \text{subject to : } &1 - y_n (w^T x_n + b) \leq 0, \forall n = 1, 2, \dots, N \quad (3) \end{aligned}$$

Ở đây, chúng ta đã lấy nghịch đảo hàm mục tiêu, bình phương nó để được một hàm khả vi, và nhân với $1/2$ để biểu thức đạo hàm đẹp hơn. Để tìm nghiệm của bài toán này, ta sử dụng phương pháp nhân tử Lagrange. Tuy nhiên, việc giải bài toán này trở nên phức tạp khi số chiều d của không gian dữ liệu và số điểm dữ liệu N tăng lên cao. Thực tế, người ta thường giải bài toán này bằng bài toán đối ngẫu. Thứ nhất, bài toán đối ngẫu có những tính chất thú vị hơn khiến nó được giải hiệu quả hơn. Thứ hai, trong quá trình xây dựng bài toán đối ngẫu, người ta thấy rằng SVM có thể được áp dụng cho những bài toán mà dữ liệu không linearly separable, tức các đường phân chia không phải là một mặt phẳng mà có thể là các mặt có hình thù phức tạp hơn.

D. Sliding Window

Sau khi huấn luyện xong bộ phân loại (classifier), bước tiếp theo chúng ta sẽ làm đó là áp dụng bộ phân loại này lên từng phần của 1 bức ảnh để có thể xác định được vị trí của vật cần nhận diện (xe máy hay ô tô). [4]

Cửa sổ trượt (sliding window) là kỹ thuật mà ta dùng 1 cửa sổ (window, hay tên gọi khác là kernel) để trượt trên mỗi pixel của ảnh. Tại mỗi pixel trong quá trình trượt, ta áp dụng phép biến đổi giữa các pixel trên cửa sổ và các pixel tương ứng trên vùng ảnh.

Một vấn đề quan trọng cần đặt ra đó là chọn kích thước của cửa sổ, bởi vì trong một bức hình, sẽ có những vật thể ở phía xa và phía gần, do đó kích thước của chúng trên bức hình sẽ không giống nhau, ta phải chọn nhiều kích thước cửa sổ thì mới có thể xác định được chính xác và đầy đủ nhất các vật thể.

Ví dụ như bức hình dưới đây, ta chọn các kích thước cửa sổ cũng như độ chồng lặp giữa các cửa sổ liên tiếp ứng với mỗi phần của bức hình:

Window Size	Overlap	Y Start	Y Stop
64x64	85	400	464
80x80	80	400	480
96x96	70	400	612
128x128	50	400	660

Hình 15: Các kích thước cửa sổ ứng với vùng hoạt động của nó trên một bức hình



Hình 16: Minh họa sliding window

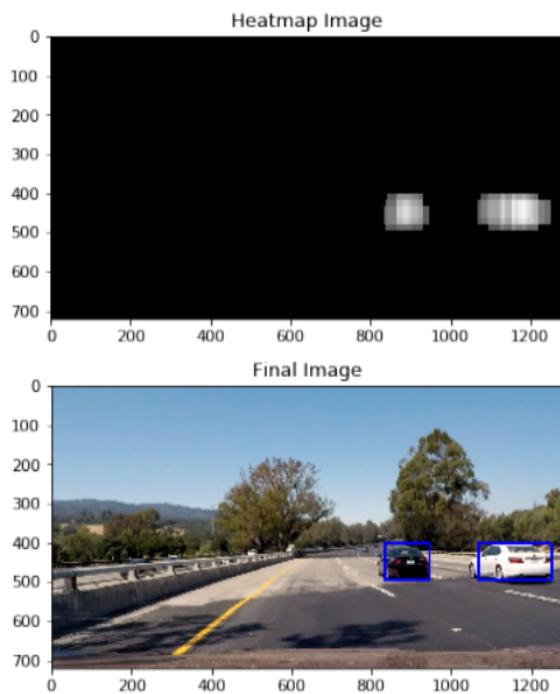
Sau khi đã xác định kích thước của các cửa sổ, ta tiến hành trượt chúng trên bức ảnh để nhận dạng ra vật thể (xe máy, ô tô), lưu ý rằng mẫu huấn luyện của chúng ta có kích thước 64x64 pixel, nên đối với các cửa sổ không cùng kích thước trên, ta phải resize lại thành 64x64 pixel.

E. Bounding Object

Sau khi sử dụng cửa sổ trượt, ta có thể xác định được vị trí vật thể, vấn đề tiếp theo đặt ra là làm sao có thể vẽ được một khung bao quanh vật thể (bounding box), Heatmap có thể giúp ta giải quyết vấn đề này.

Đầu tiên, tạo một bức ảnh trống màu đen (blank black

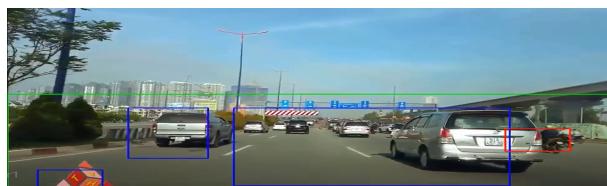
image) có cùng kích thước với bức hình gốc, tại các cửa sổ mà được xác định là chứa vật thể, ta sẽ cộng giá trị của các điểm ảnh trong cửa sổ này lên 1. Theo cách này, chúng ta sẽ có các vùng có cường độ khác nhau và vùng chứa vật thể chính là vùng có cường độ mạnh nhất. Dưới đây là hình minh họa việc sử dụng heatmap để vẽ bounding box.



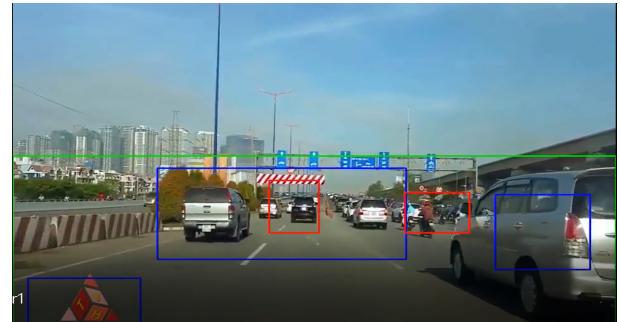
Hình 17: Dùng heatmap để xác định bounding box cho object

IV. KẾT QUẢ

Trong phần này, nhóm sẽ thiết lập chương trình mô phỏng bằng Python để kiểm chứng các kết quả phân tích ở phần trên. Với stepsize = 0.3, thuật toán nhận diện khá chính xác, tuy nhiên vẫn có một số trường hợp nhận dạng sai vì dữ liệu huấn luyện chưa đủ lớn.

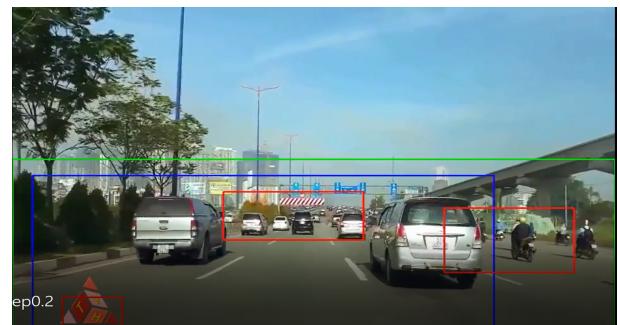


Hình 18: Kết quả nhận diện với stepsize=0.3



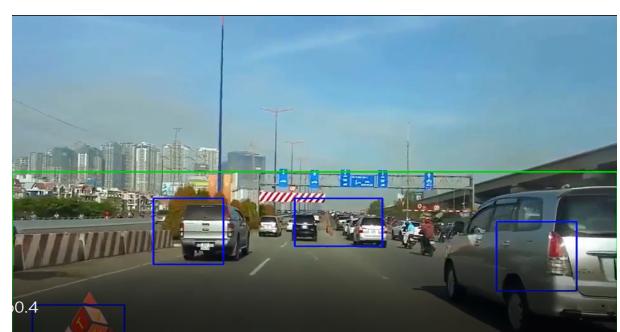
Hình 19: Trường hợp nhận dạng nhầm xe máy thành ô tô

Với stepsize=0.2, ta thấy đường bao lớn hơn. Lý do là vì bước nhảy nhỏ hơn tạo ra nhiều khung, sau đó heatmap gộp các khung này lại tạo thành khung lớn



Hình 20: Kết quả nhận diện với stepsize=0.2

Stepsize=0.4, bước nhảy lớn thì sẽ không xác định được xe máy, do kích thước chiều ngang của người đi xe máy khá nhỏ



Hình 21: Kết quả với stepsize=0.4

V. KẾT LUẬN

Phương pháp HOG là phương pháp trích xuất đặc trưng giúp giảm số chiều của dữ liệu -> hiệu quả về tốc độ tính toán. Thuật toán SVM là thuật toán cho kết quả phân loại tốt. Kết quả thực hiện tốt, tuy nhiên vẫn có trường hợp nhận dạng sai do database chưa đủ lớn. Chưa có sự tối ưu về mặt thời gian và kích thước đường bao.

LỜI CẢM ƠN

Để có được bài nghiên cứu này, nhóm xin cảm ơn Thầy Đặng Nguyên Châu đã hỗ trợ rất nhiều về các kiến thức về xử lý ảnh cũng như các phương pháp phân loại, ngoài ra thầy cũng tạo điều kiện thuận lợi để nhóm có thời gian làm bài báo cáo được tốt hơn.

TÀI LIỆU

- [1] Nguyễn Phương Lan, "Tìm hiểu về hog(histogram of oriented gradients), "<https://viblo.asia/p/tim-hieu-ve-hoghistogram-of-oriented-gradients-m68Z0wL6KkG>, 8/2017
- [2] Satya Mallick, Histogram of Oriented Gradients, <https://www.learnopencv.com/histogram-of-oriented-gradients/>, 6/12/2016
- [3] "Support Vector Machine", machinelearningcoban.com, 9/4/2017
- [4] Harveen Singh, "Vehicle Detection and Tracking", towardsdatascience.com, 18/3/2018