**Đồ án thị giác máy tính: Đếm vật thể trong ảnh**

**1. Phát biểu bài toán, input và output**

Ở đồ án này, nhóm xây dựng phương pháp để đếm vật thể trong bức ảnh, vật thể ở đây có thể khác nhau tùy thuộc vào tập dữ liệu huấn luyện được đưa vào.

Gọi là bài toán đếm vật thể, nhưng về bản chất thì có thể chia bài toán này thành hai bài toán nhỏ đặc trưng trong thị giác máy tính:

* Bài toán phát hiện vật thể (object detection)
* Bài toán định vị vật thể (localization)

Trong khuôn khổ đồ án này, nhóm chỉ sử dụng những thuật toán phân loại đơn giản (KNN, SVM, Logistic Regression) cùng với phương pháp trích xuất đặc trưng phù hợp cho bài toán (object detection), sử dụng phương pháp sliding window cho bài toán Localization.

Input và output cho bài toán tổng quát lớn có thể được phát biểu như sau:

* Input:

Bức ảnh có chứa vật thể (nhiều hoặc ít) bên trong, góc chụp cố định (sự thay đổ hình dạng của vật thể theo thời gian không đáng kể)

Vd1: bãi đổ xe góc chụp thẳng từ trên cao  


Vd2: ảnh chụp đầu của các ống nước



* Output:

Vị trí của vật thể phát hiện được trong bức ảnh

Vd: Ảnh có chứa văn bản, ảnh chụp màn hình, thức uống có ga, ngoài trời

Mô tả được tạo tự động

Đối với bài toán localization, input và output có thể được phát biểu như sau:

* Input:

Ảnh x => Listx = {xi}N thuộc x

* Output:

(bounding box) Listy = {yi} = flocal(fobject detection({xi}) = flocal(Listyobject detection)

Đối với bài toán object detection, input và output có thể được phát biểu như sau:

* Input:

D = {(xi, yi)}

L = Uyi = {có vật thể, không có vật thể}

ảnh x (vector đặc trưng)

Vd:

  có vật thể,   không có vật thể

* Output:

(predicted class, probability) yobject detection = fobject detection(x) thuộc L

Về cơ bản, từ một bức hình lớn ban đầu, thông qua bài toán Local sẽ có được danh sách những bức hình nhỏ hơn được cắt từ bức hình lớn.

Các bức hình nhỏ hơn sẽ thông qua bài toán object detection và có được kết quả dự đoán cho bức hình nhỏ.

Từ danh sách kết quả dự đoán của các bức hình nhỏ, bài toán Local sẽ lọc và trả về sau cùng những bức hình nhỏ đã đủ diều kiện (tọa độ Bounding box), chính là output sau cùng.

**2. Lý do**

Nhu cầu thực tế:  
a. Quản lý và giám sát:

* Đếm xe giúp quản lý giao thông, giảm ùn tắc, tối ưu đèn tín hiệu.
* Đếm sản phẩm hỗ trợ giám sát năng suất và phát hiện lỗi sản xuất.

b. An ninh và phân tích hành vi:

* Đếm người giúp phát hiện đám đông bất thường, đảm bảo an ninh.

Hỗ trợ ra quyết định:  
a. Tăng hiệu quả:

* Đếm vật thể giúp tối ưu nguồn lực, cảnh báo tự động.
* Ví dụ: Điều chỉnh nhân viên bán lẻ dựa trên số lượng khách hang

b. Dự đoán:

* Cung cấp dữ liệu phân tích xu hướng, dự báo nhu cầu.

Ứng dụng khoa học:  
a. Robot học:

* Robot đếm hàng trong kho đảm bảo độ chính xác khi vận chuyển.  
  b. Phân tích môi trường:
* Đếm động vật hoặc cây cối từ ảnh vệ tinh đánh giá tác động môi trường.

Phát triển AI:

* Trên thực thế, hiệu năng sử dụng phương pháp này để đếm vật thể với số lượng lớn vẫn chưa được tối ưu, vì tốc độ xử lí vẫn còn tương đối chậm trong khi độ chính xác và mức độ hiệu quả của đặc trưng vẫn thua xa so với các kỹ thuật học sâu, tuy nhiên, nhóm nghĩ đây vẫn một phương pháp tiềm năng đối với những bài toán cần nhận diện và định vị vật thể đơn giản hơn.

**3. Phương pháp**

**3.1. Bài toán Object detection**

**Đặc trưng**



**Đặc trưng canny egde:**

Canny Edge Detection là một phương pháp phát hiện biên trong ảnh, được giới thiệu bởi John F. Canny vào năm 1986. Đây là một kỹ thuật dựa trên toán học và tối ưu hóa, được thiết kế để phát hiện các cạnh rõ ràng và giảm nhiễu trong ảnh.

Cách hoạt động:

1. Làm mờ ảnh: Ảnh được làm mờ bằng bộ lọc Gaussian để giảm nhiễu.
2. Tính gradient: Xác định độ lớn và hướng gradient tại mỗi điểm ảnh bằng cách sử dụng các đạo hàm.
3. Non-maximum suppression: Loại bỏ các điểm không phải là cực đại cục bộ để làm mỏng các cạnh.
4. Double thresholding: Phân loại cạnh mạnh (giữ lại), cạnh yếu (xét tiếp), và loại bỏ nhiễu dựa trên hai ngưỡng giá trị.
5. Edge tracking by hysteresis: Liên kết các cạnh yếu với các cạnh mạnh nếu chúng liền kề, tạo thành các cạnh đầy đủ.

Canny edge nổi bật nhờ khả năng cân bằng giữa độ chính xác và khả năng chống nhiễu tốt.

Trong **thực nghiệm** của nhóm, đặc trưng Canny edge được cải thiện bằng cách:

* Mở rộng biên để làm rõ các cạnh.
* Loại bỏ các khe hở nhỏ trong đường biên

Ảnh có chứa biểu đồ, văn bản, thiết kế

Mô tả được tạo tự động, Ảnh có chứa Phông chữ, biểu tượng, Đồ họa, văn bản

Mô tả được tạo tự động, Ảnh có chứa Phông chữ, biểu tượng, Đồ họa, thiết kế

Mô tả được tạo tự động

Accuracy KNN, độ đo tương quan:

|  |  |  |
| --- | --- | --- |
|  | Chỉ canny edge | CE đã chỉnh sửa |
| Car k = 10 | 0.84 | 0.83 |
| Pipe k = 8 | 0.81 | 0.88 |
| Human k = 9 | 0.68 | 0.64 |

**Đặc trưng HOG**

HOG (Histogram of Oriented Gradients) là một đặc trưng thường dùng trong xử lý ảnh và thị giác máy tính để phát hiện đối tượng, được giới thiệu bởi Navneet Dalal và Bill Triggs vào năm 2005 trong bài báo về phát hiện người.

Hoạt động của HOG:

1. Phân vùng ảnh: Ảnh được chia thành các ô nhỏ (cells).
2. Tính gradient: Tính toán độ lớn và hướng của gradient tại mỗi pixel để xác định cách thay đổi cường độ sáng.
3. Lập histogram: Với mỗi ô, histogram được tạo dựa trên các hướng gradient.
4. Chuẩn hóa: Để tăng tính ổn định, các histogram trong khối (block) lân cận được chuẩn hóa, làm giảm ảnh hưởng của thay đổi ánh sáng.

Ưu điểm:

* Nổi bật trong việc phát hiện các mẫu hoặc hình dạng vật thể

**pixels\_per\_cell** (số pixel trên mỗi ô):  
Đây là kích thước của mỗi ô (cell) trong đơn vị pixel. Ví dụ, nếu pixels\_per\_cell=(8, 8), điều đó có nghĩa là mỗi ô sẽ gồm 8 pixel theo chiều ngang và 8 pixel theo chiều dọc. Trong mỗi ô, một histogram sẽ được tính toán dựa trên hướng gradient của các pixel.

**cells\_per\_block** (số ô trên mỗi khối):  
Đây là kích thước của mỗi khối (block) trong đơn vị số lượng ô. Mỗi khối chứa nhiều ô và được sử dụng để chuẩn hóa giá trị của histogram trong các ô nhằm giảm nhiễu và tăng độ chính xác. Ví dụ, nếu cells\_per\_block=(2, 2), điều đó có nghĩa là mỗi khối sẽ bao gồm 2 ô theo chiều ngang và 2 ô theo chiều dọc.

Ảnh có chứa đen và trắng, mẫu, Phông chữ, đơn sắc

Mô tả được tạo tự động 8x8, Ảnh có chứa đen và trắng, màu đen, màu trắng, ảnh chụp màn hình

Mô tả được tạo tự động 16x16

So sánh accuracy (KNN, độ đo tương quan):

|  |  |  |  |
| --- | --- | --- | --- |
| Đặc trưng | Canny edge | HOG 8x8 | HOG 16x16 |
| Car | 0.83 (k = 10) | 0.95 (k = 4) | 0.93 (k = 4) |
| Pipe | 0.88 (k = 8) | 0.92 (k = 4) | 0.94 (k = 4) |
| Human | 0.64 (k = 9) | 0.88 (k = 4) | 0.89 (k = 4) |

**Thuật toán phân loại:**

**1. KNN (K-Nearest Neighbors)**

* **Nguyên lý:** KNN là thuật toán phân loại hoặc hồi quy dựa trên khoảng cách. Một điểm mới được gán nhãn hoặc giá trị dựa vào K điểm lân cận gần nhất trong tập huấn luyện.
* Cách hoạt động:
  1. Tính khoảng cách giữa điểm mới và tất cả các điểm trong tập dữ liệu
  2. Chọn K điểm gần nhất.
  3. Dựa vào đa số nhãn (phân loại) hoặc giá trị trung bình (hồi quy) của các điểm này để dự đoán.
* **Ưu điểm:** Đơn giản, không yêu cầu huấn luyện.
* **Nhược điểm:** Hiệu quả phụ thuộc vào giá trị K và cách đo khoảng cách; tốn tài nguyên nếu dữ liệu lớn.
* **Thực nghiệm:** đối với thuật toán KNN này, nhóm thử nghiệm nhiều tham số K và độ đo, trong số đó, độ đo tương quan (correlation) cho kết quả tốt hơn độ đo mặc định thường thấy là Euclid.

Độ đo

|  |  |  |
| --- | --- | --- |
|  | Euclid | Correlation |
| Car, HOG8x8, KNN k=4 | 0.9 | 0.95 |
| Pipe, HOG16x16, KNN k = 4 | 0.93 | 0.94 |
| Human, HOG16x16, KNN k = 4 | 0.79 | 0.89 |

Sử dụng độ đo tương quan cho thuật toán KNN.

Car, HOG8x8:

Ảnh có chứa văn bản, hàng, Sơ đồ, biểu đồ

Mô tả được tạo tự động

Pipe, HOG16x16:

Ảnh có chứa văn bản, biểu đồ, hàng, Sơ đồ

Mô tả được tạo tự động

Human, HOG16x16:

Ảnh có chứa văn bản, hàng, Sơ đồ, biểu đồ

Mô tả được tạo tự động

**2. Logistic Regression**

* **Nguyên lý:** Logistic Regression là một thuật toán phân loại dựa trên hồi quy tuyến tính nhưng sử dụng hàm sigmoid để chuyển đổi giá trị dự đoán thành xác suất.
* **Cách hoạt động:**
  1. Xác định mối quan hệ giữa đầu vào X và đầu ra y qua hàm tuyến tính.
  2. Dùng hàm sigmoid σ để tính xác suất.
  3. Gán nhãn dựa trên ngưỡng.
* **Ưu điểm:** Dễ hiểu, hiệu quả cho phân loại nhị phân và có thể mở rộng sang phân loại đa nhãn (One-vs-Rest).
* **Nhược điểm:** Giới hạn với bài toán tuyến tính, không tốt khi dữ liệu không tách rời được.
* **Thực nghiệm:**

**3. SVM (Support Vector Machine)**

* **Nguyên lý:** SVM phân loại dữ liệu bằng cách tìm siêu phẳng (hyperplane) tối ưu, tối đa hóa biên (margin) giữa các lớp.
* **Cách hoạt động:**
  1. Chọn một siêu phẳng tách rời các điểm thuộc hai lớp (tuyến tính).
  2. Với dữ liệu không tuyến tính, sử dụng kernel trick để ánh xạ dữ liệu sang không gian cao hơn, giúp tách rời dễ hơn.
* **Ưu điểm:** Hiệu quả cao với dữ liệu nhỏ và tách biệt rõ ràng; linh hoạt với kernel.
* **Nhược điểm:** Không phù hợp với dữ liệu lớn.
* **Thực nghiệm:**

Car:

Ảnh có chứa văn bản, hàng, biểu đồ, Sơ đồ

Mô tả được tạo tự độngẢnh có chứa hàng, văn bản, biểu đồ, Sơ đồ

Mô tả được tạo tự động

Pipe:

Ảnh có chứa văn bản, biểu đồ, hàng, Sơ đồ

Mô tả được tạo tự động Ảnh có chứa văn bản, biểu đồ, hàng, Sơ đồ

Mô tả được tạo tự động

Human

Ảnh có chứa văn bản, hàng, biểu đồ, Sơ đồ

Mô tả được tạo tự độngẢnh có chứa văn bản, biểu đồ, hàng, Sơ đồ

Mô tả được tạo tự động

So sánh:

|  |  |  |  |
| --- | --- | --- | --- |
| Thuật toán | KNN | LR | SVM |
| Car | Accuracy: 0.9515151515151515  Precision: 0.9377379619260918  Recall: 0.9515151515151515  Confusion matrix:  [[ 86 7]  [ 9 228]] | Accuracy: 0.9515151515151515  Precision: 0.9401116101810263  Recall: 0.9515151515151515  Confusion matrix:  [[ 85 8]  [ 8 229]] | Accuracy: 0.9696969696969697  Precision: 0.9686232458389669  Recall: 0.9696969696969697  Confusion matrix:  [[ 86 7]  [ 3 234]] |
| Pipe | Accuracy: 0.9423076923076923  Precision: 0.9351648351648352  Recall: 0.9423076923076923  Confusion matrix:  [[58 2]  [ 7 89]] | Accuracy: 0.9615384615384616  Precision: 0.9571036376115305  Recall: 0.9615384615384616  Confusion matrix:  [[58 2]  [ 4 92]] | Accuracy: 0.9743589743589743  Precision: 0.9704873026767331  Recall: 0.9743589743589743  Confusion matrix:  [[59 1]  [ 3 93]] |
| Human | Accuracy: 0.8906942392909897  Precision: 0.8984035527650696  Recall: 0.8906942392909897  Confusion matrix:  [[220 56]  [ 18 383]] | Accuracy: 0.8508124076809453  Precision: 0.848344756239493  Recall: 0.8508124076809453  Confusion matrix:  [[217 59]  [ 42 359]] | Accuracy: 0.8951255539143279  Precision: 0.8964780224886163  Recall: 0.8951255539143279  Confusion matrix:  [[230 46]  [ 25 376]] |

**3.1. Bài toán Localization**

**Sliding window**

quét qua toàn bộ ảnh để phân tích từng vùng nhỏ, nhằm tìm kiếm đối tượng. Vì đối tượng có thể xuất hiện ở nhiều vị trí và kích thước khác nhau trong khung hình, kỹ thuật này chia nhỏ và quét ảnh theo từng bước.

**Quy trình:**

1. Chia ảnh thành các vùng nhỏ (sub-windows):
   * Một cửa sổ (window) với kích thước cố định sẽ được di chuyển từ góc trên bên trái của ảnh đến góc dưới bên phải.
   * Cửa sổ này "trượt" qua ảnh theo một bước nhảy cố định (step\_size), cả theo chiều ngang và chiều dọc.
2. Lấy từng vùng ảnh:
   * Tại mỗi vị trí của cửa sổ, vùng ảnh tương ứng được cắt ra.
   * Các vùng này có thể được thay đổi kích thước để phù hợp với đầu vào của mô hình.
3. Lặp lại với nhiều kích thước cửa sổ khác nhau:
   * Vì đối tượng có thể có kích thước lớn/nhỏ, bạn cần dùng nhiều kích thước cửa sổ (multi-scale sliding window) để quét.

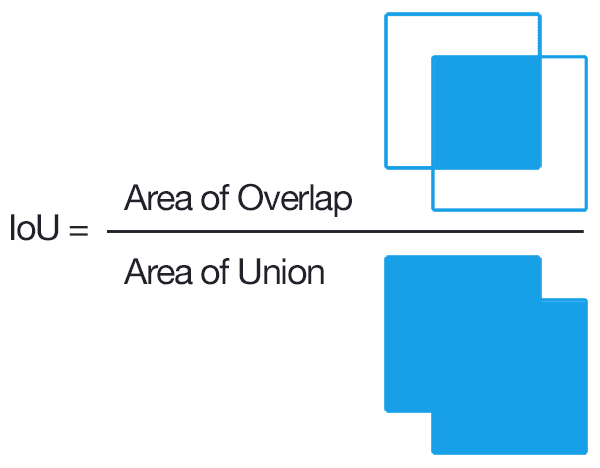
**Non – Maximum Suppression**

**Mục đích:**

Khi sử dụng sliding window, nhiều vùng có thể chồng lấn nhau và cùng phát hiện một đối tượng. **Non-Maximum Suppression** được dùng để loại bỏ các vùng chồng lấn này, chỉ giữ lại vùng có xác suất cao nhất.

Quy trình:

1. Sắp xếp bounding box theo xác suất:
   * Mỗi bounding box có một "điểm số" xác suất (ví dụ: khả năng là xe hơi). Sắp xếp các box theo thứ tự giảm dần của xác suất.
2. Lấy box có xác suất cao nhất:
   * Chọn box đầu tiên trong danh sách (xác suất cao nhất) và thêm nó vào danh sách "giữ lại".
3. Loại bỏ các box chồng lấn:
   * Với các box còn lại, tính Intersection over Union (IoU):
     + Intersection: Phần diện tích giao nhau giữa 2 box.
     + Union: Tổng diện tích của 2 box, trừ đi phần giao.
     + IoU = Intersection / Union



* + Loại bỏ các box có IoU lớn hơn ngưỡng (threshold). Điều này nghĩa là nếu box nào chồng lấn quá nhiều với box đã chọn, nó sẽ bị loại bỏ.

1. Lặp lại:
   * Tiếp tục với box có xác suất cao tiếp theo và lặp lại quá trình cho đến khi không còn box nào.

Thực nghiệm (IoU):

|  |  |
| --- | --- |
| Car (SVM gamma = 0.1, C = 10, HOG8x8) | 0.75 |
| Pipe (SVM gamma = 0.1, C = 100, HOG16x16) | 0.63 |
| Human (SVM gamma = 0.084, C = 10, HOG16x16) | 0.75 |

**Kết luận**

Cần được tối ưu hơn ở khâu train mô hình (bài toán phân loại và bài toán định vị hoạt động vẫn còn tương đối độc lập). Xây dựng lại tập dataset với định dạng phù hợp hơn).

Hiệu năng thấp, không hiệu quả đối với những bài toán đếm những vật thể có hình dạng phức tạp. Để giải quyết, cần tối ưu phép trích xuất đặc trưng đối với mỗi loại vật thể khác nhau (xây dựng phương pháp chuyên biệt cho từng loại vật thể).

Ổn hơn nếu sử dụng trên những vật thể đơn giản như ống nước, đồng xu, …

**Một số dự đoán:**

Car (SVM gamma = 0.1, C = 10, HOG8x8)

Ảnh có chứa ảnh chụp màn hình, mạch điện

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, mạch điện

Mô tả được tạo tự động

Pipe (SVM gamma = 0.1, C = 100, HOG16x16)

Ảnh có chứa màu xanh lá cây, tác phẩm nghệ thuật

Mô tả được tạo tự động

Human (SVM gamma = 0.084, C = 10, HOG16x16)

Ảnh có chứa trang phục, giày dép, người, ngoài trời

Mô tả được tạo tự độngẢnh có chứa giày dép, người, trang phục, ngoài trời

Mô tả được tạo tự động

Ảnh có chứa ngoài trời, trang phục, bầu trời, người

Mô tả được tạo tự độngẢnh có chứa giày dép, đàn ông, trang phục, người

Mô tả được tạo tự động