BÀI TẬP LỚN 2

Trí tuệ nhân tạo

Nhóm 3

Ngày 9 tháng 3 năm 2024

Sinh viên thực hiện
106200284 - Hồ Đức Vũ - 20KTMT2
106200240 - Nguyễn Minh Phương - 20KTMT1
106200241 - Huỳnh Vũ Đình Phước - 20KTMT1

Giáo viên hướng dẫn
TS. Hoàng Lê Uyên Thục

Môn học
Trí tuệ nhân tạo

Đề bài
Phương pháp Histogram of Oriented Gradients

Xuất bản
Đà Nẵng, Ngày 9 tháng 3 năm 2024

Số trang
14

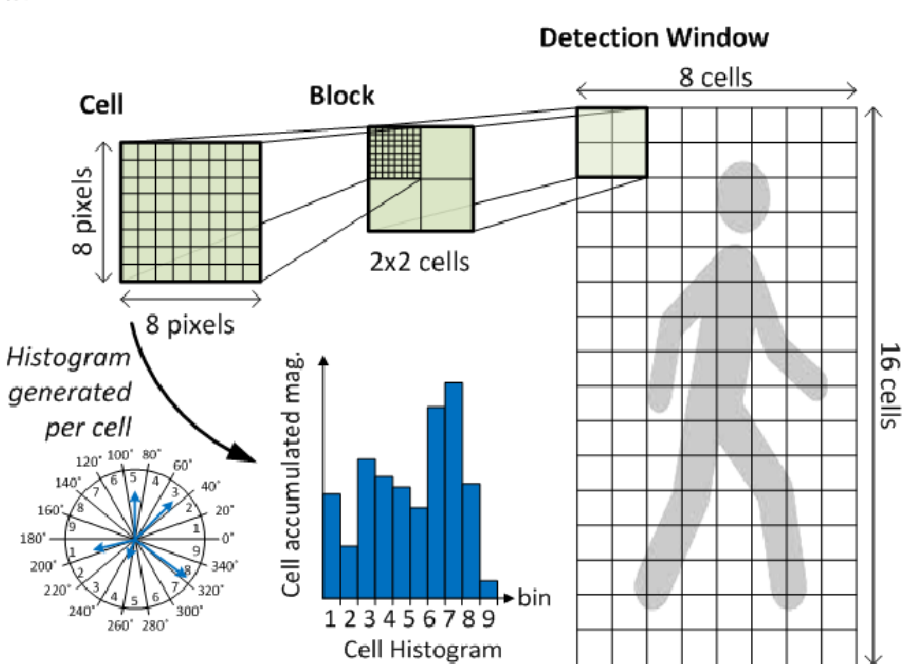
Mục lục

Nội dung báo cáo

1 Giới thiệu phương pháp Histogram of Oriented Gradient

Phương pháp Histogram of Oriented Gradient (thường được gọi là HOG) là một kỹ thuật được sử dụng trong lĩnh vực thị giác máy tính và nhận diện đối tượng. Phương pháp này được sử dụng phổ biến trong các ứng dụng như nhận diện khuôn mặt, nhận diện người và phát hiện vật thể trong ảnh.

Trước tiên để hiểu rõ hơn về HOG, ta cần hiểu về histogram và gradient là gì. Histogram là một biểu đồ thống kê của sự phân phối giá trị cường độ màu sắc của các pixel trong ảnh. Còn Gradient thể hiện độ thay đổi của giá trị một hàm theo mỗi hướng, nếu xem xét trong lĩnh vực xử lý ảnh và thị giác máy tính thì ta có thể định nghĩa nó như là một đạo hàm của các vector cường độ màu sắc, nó sử dụng để xác định cường độ và hướng thay đổi màu sắc từ đó phát hiện hướng di chuyển của các vật thể trong ảnh. Và HOG nếu hiểu theo cách đơn giản, nó là sự kết hợp của Histogram và Gradient.



Hình 1.1: Tổng quan về Histogram of Oriented Gradient.

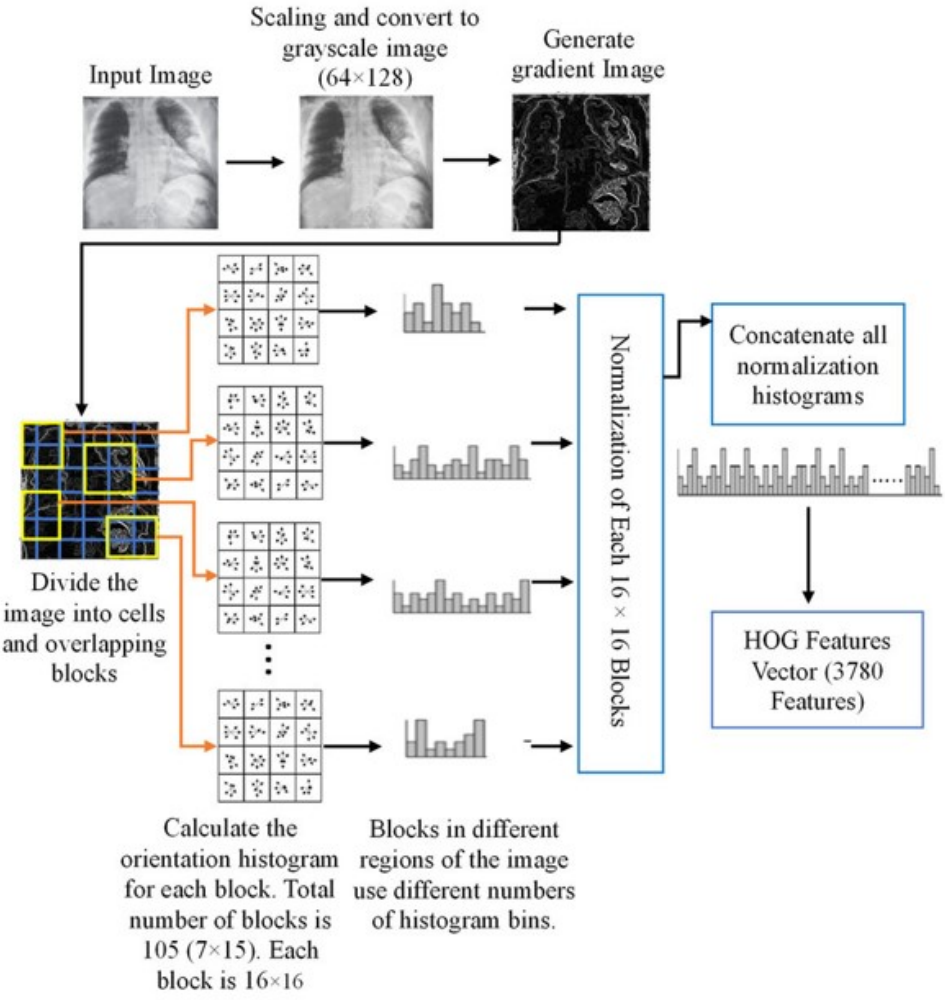
Phương pháp HOG sẽ tạo ra các *feature descriptor* với mục đích phát hiện vật thể. Từ một bức ảnh đầu vào, ta có thể phân tích nó thành 2 ma trận bao gồm *gradient magnitute* và *gradient orientation*. Sau đó kết hợp thông tin có từ 2 ma trận vào một biểu đồ phân phối histogram, trong đó gradient magnitute được biểu diễn theo các nhóm bins của gradient orientation, nói đơn giản thì gradient magnitute sẽ được biểu diễn theo trục y và gradient orientation là trục x theo một phương trình $y = f(x)$. Cuối cùng thì ta thu được *HOG feature vector* là một vector

mô tả sự phân bố của các đặc điểm cấu trúc và hình dạng của vật thể trong ảnh.

Như mô tả tại hình 1.1, Hình ảnh được chia thành dạng lưới, tại các ô sẽ cho ra biểu đồ histogram với gradient magnitute được biểu diễn theo nhóm bins của gradient orientation, cửa sổ sẽ trượt đến hết bức ảnh và cho ra một biểu đồ histogram tổng quát của cả bức ảnh, biểu đồ này đại diện cho HOG feature vector đã được đề cập ở trên.

2 Phân tích phương pháp

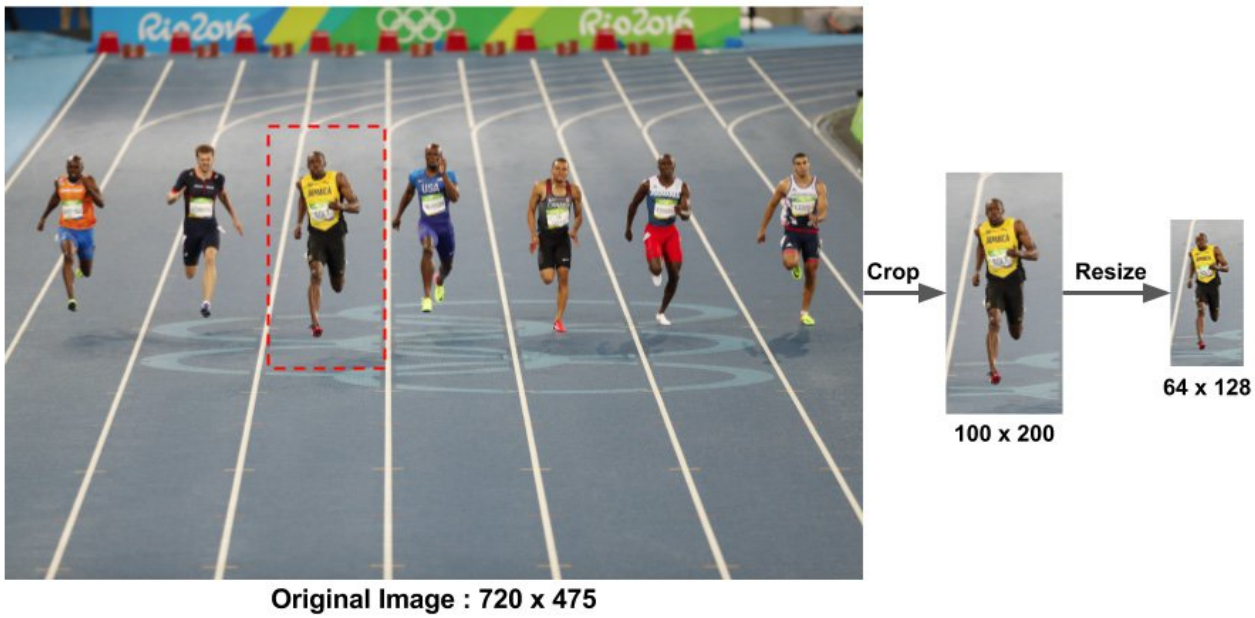
Trong phần này, chúng ta sẽ phân tích phương pháp HOG thông qua các bước thực hiện. Thứ nhất là *tiền xử lý ảnh* (pre-processing data), bước này nhằm đưa ảnh về một kích thước và kiểu ảnh cố định cung cấp cùng một loại đầu vào cho module. Thứ hai là ta chia ảnh thành các cell và block và *tính toán Gradient* cho từng cell. Bước thứ 3 là tính toán *vector đặc trưng trên từng cell* và *chuẩn hóa* giá trị thu được cho các block. Cuối cùng là kết nối các vector đã được chuẩn hóa trong các block thành một vector cho toàn bộ ảnh còn được gọi là *HOG feature vector*. Hình 2.1 mô tả tổng quan về các bước thực hiện này.



Hình 2.1: Các bước thực hiện trong phương pháp HOG.

2.1 Tiền xử lý ảnh

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. Trong bài toán này, để thuận tiện cho việc chia đều hình ảnh thành các khối, ô và tính toán đặc trưng ở các bước tiếp theo, chúng ta cần resize kích thước tất cả các hình ảnh trong tập dữ liệu về một kích thước chung và chuyển sang ảnh xám trước khi thực hiện các bước tiếp theo.



Hình 2.2: Tiền xử lý ảnh đầu vào

2.2 Tính toán Gradient

Thực hiện bằng hai phép nhân chập ảnh gốc với 2 chiều, tương ứng với các toán tử lấy đạo hàm theo hai hướng Ox và Oy. Trong đó, 2 hướng tương ứng đó là:

$$D_x = [-1 \quad 0 \; 1] \quad \text{và } D_y = [-1 \quad 0 \; 1]^T \tag{2.1}$$

Nếu chúng ta kí hiệu I là ma trận ảnh gốc và Ix và Iy là 2 ma trận ảnh mà mỗi điểm trên nó lần lượt là đạo hàm theo trục trục Ox và Oy. Chúng ta có thể tính toán được kernel như sau:

$$I_x = I * Dx \tag{2.2}$$

$$I_y = I * Dy \tag{2.3}$$

Kí hiệu * tương tự như phép tích chập giữa bộ lọc bên trái và ảnh đầu vào bên phải. Giá trị độ lớn gradient (gradient magnitude) và phương gradient (gradient direction) có thể được tạo ra từ 2 đạo hàm Ix và Iy theo công thức bên dưới:

- Độ lớn gradient:

$$G = \sqrt{Ix.^2 + Iy.^2} \tag{2.4}$$

- Phương gradient:

$$\theta = \arctan(Iy/Ix) \tag{2.5}$$

2.3 Vector đặc trưng cho từng cell

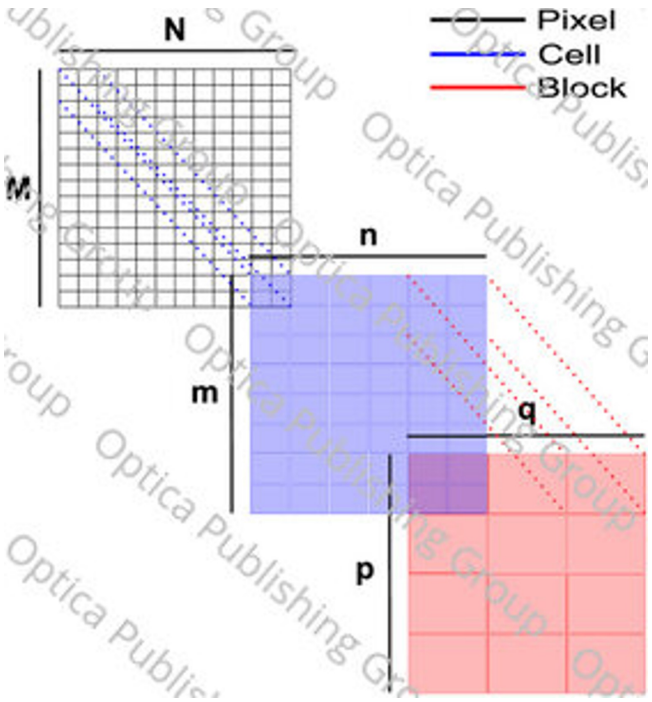
Để tính toán vector đặc trưng cho từng ô (cell), chúng ta cần chia hình ảnh thành các block, mỗi block lại chia đều thành các cell như mô tả tại hình 2.3. Để xác định được số block, chúng ta sẽ sử dụng công thức sau:

$$n_{block\_image} = \left(\frac{W_{image} - W_{block} * W_{cell}}{W_{cell}} + 1\right) * \left(\frac{H_{image} - H_{block} * H_{cell}}{H_{cell}} + 1\right)$$

trong đó:

W_{image}, W_{block}, W_{cell} : lần lượt là chiều rộng của ảnh, khối, ô

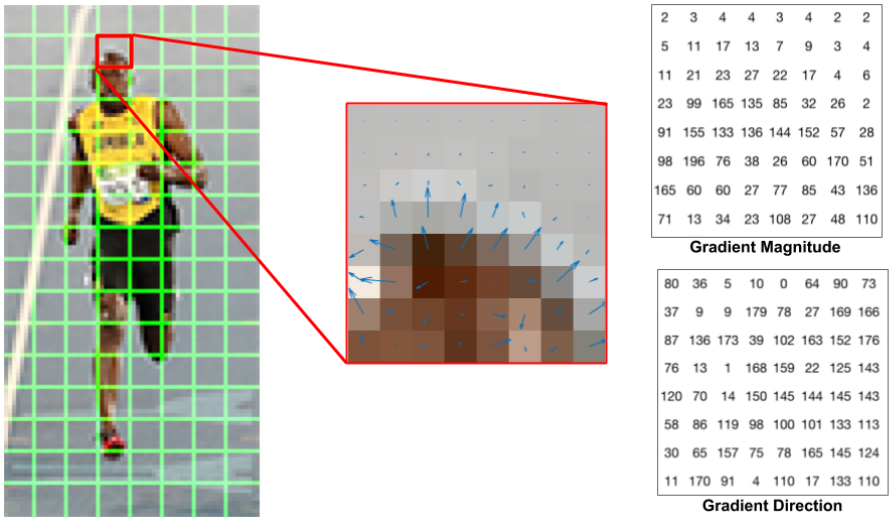
H_{image}, H_{block}, H_{cell} : lần lượt là chiều dài của ảnh, khối, ô



Hình 2.3: Phân ảnh thành các cells và blocks để tính toán vector đặc trưng.

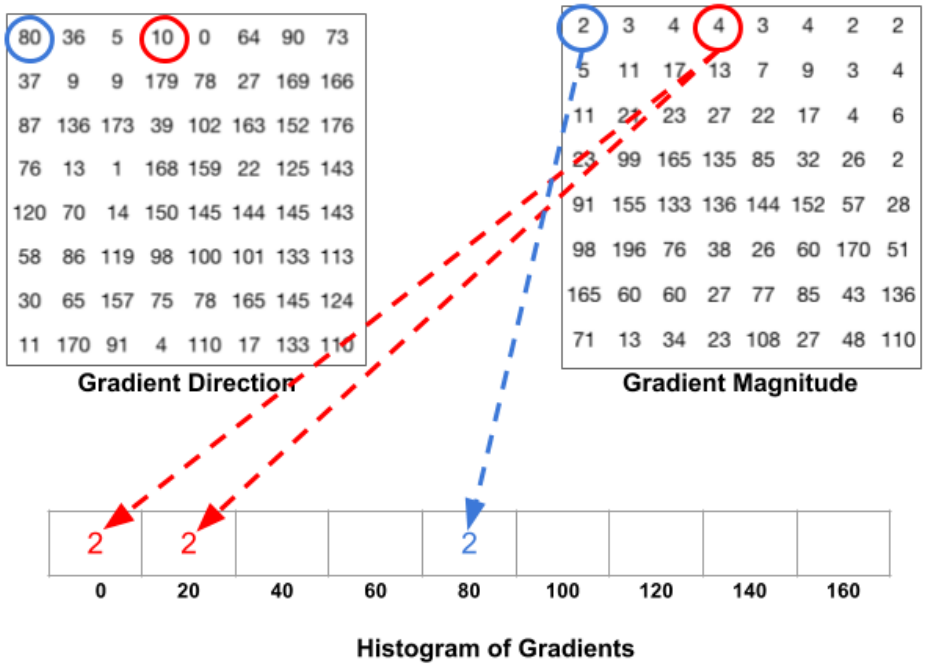
Hình ảnh được chia thành một lưới ô vuông mà mỗi một ô có kích thước 8x8 pixels. Như vậy chúng ta có tổng cộng 64 ô pixels tương ứng với mỗi ô. Trên mỗi một ô trong 64 pixels ta sẽ cần tính ra 2 tham số đó là độ lớn gradient (gradient magnitute) và phương gradient (gradient direction). Như vậy tổng cộng 8x8x2 = 128 giá trị cần tính bao gồm 64 giá trị gradient magnitute và 64 giá trị gradient direction như ma trận tại hình 2.4.

Mapping độ lớn gradient vào các bins tương ứng của phương gradient. Sắp xếp các giá trị phương gradient theo thứ tự từ nhỏ đến lớn và chia chúng vào 9 bins. Độ lớn của phương gradient sẽ nằm trong khoảng [0, 180] nên mỗi bins sẽ có độ dài là 20 như hình bên dưới. Mỗi một phương gradient sẽ ghép cặp với một độ lớn gradient ở cùng vị trí tọa độ. Khi biết được phương gradient thuộc bins nào trong véc tơ bins, ta sẽ điền vào giá trị giá trị của độ lớn gradient vào chính bin đó.



Hình 2.4: Hình ảnh vận động viên được chia thành các lưới ô vuông, mỗi ô vuông có kích thước 8x8 pixels. Trên mỗi ô chúng ta thực hiện tính đạo hàm theo Ox, Oy để thu được 2 ma trận bên phải là gradient magnitude và gradient direction.

Chẳng hạn trong hình 2.5 ô được khoanh trong hình tròn viền xanh tương ứng với phương gradient là 80 và độ lớn gradient là 2. Khi đó tại véc tơ bins của HOG, phương gradient bằng 80 sẽ rơi vào vị trí thứ 5 nên tại ô này chúng ta điền giá trị 2 ứng với độ lớn gradient.



Hình 2.5: Mapping độ lớn gradients với các bins.

Đầu mút là các giá trị chia hết cho độ rộng của một bin (chẳng hạn 0, 20, 40,... là những đầu mút bin). Trong trường hợp độ lớn phương gradients không rơi vào các đầu mút, ta sẽ sử dụng linear interpolation để phân chia độ lớn gradient về 2 bins liền kề mà giá trị phương gradient

rơi vào. Ví dụ: giá trị phương gradient bằng x ghép cặp với độ lớn gradient bằng $y.x \in [x0, x1]$ tức là phương gradients rơi vào khoảng giữa bin thứ l và bin thứ (l-1). Khi đó tại 2 bins l và (l-1) được điền vào giá trị cường độ theo công thức interpolation:

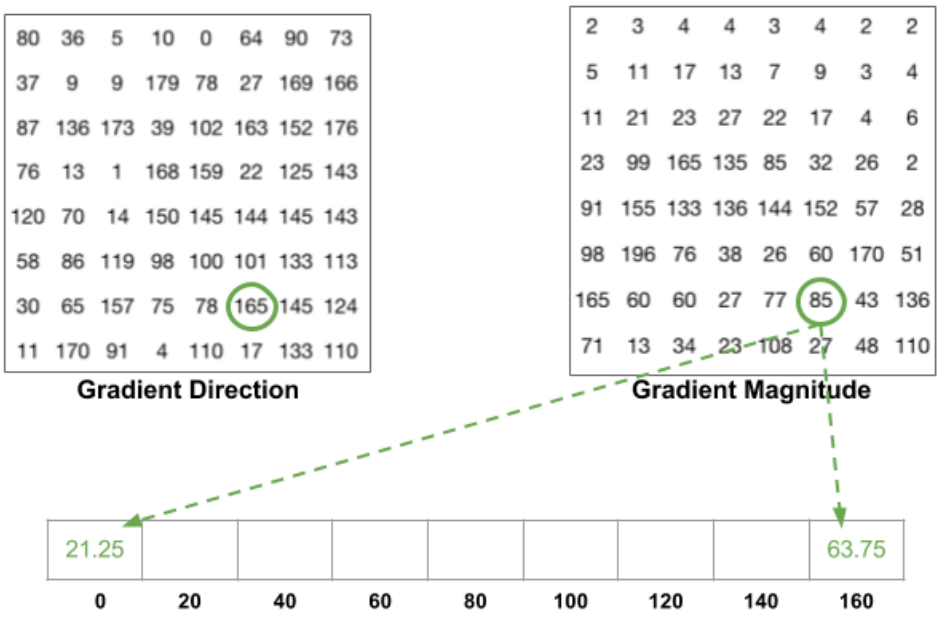
- Giá trị bin l-1:

$$x_{(l-1)} = \frac{(x_1 - x)}{x_1 - x_0} * y \tag{2.6}$$

- Giá trị bin l:

$$x_l = \frac{x - x_0}{x_1 - x_0} * y \tag{2.7}$$

Hình 2.6 sử dụng công thức (2.6) và (2.7) để tính toán biểu đồ histogram từ 2 ma trận Gradient Magnitute và Gradient Orientation.



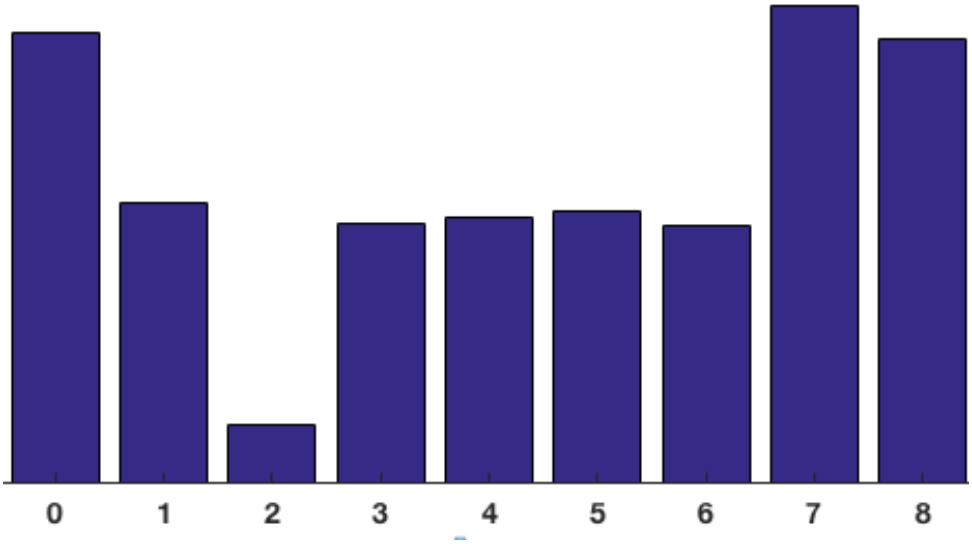
Hình 2.6: Ví dụ với điểm được khoanh tròn bởi hình tròn màu xanh có phương gradient bằng 165 và độ lớn gradient bằng 85. Ta phân chia giá trị về các bins 0 (hoặc 180) và 160 các giá trị theo công thức interpolation bên trên.

Tính tổng tất cả các độ lớn gradient thuộc cùng 1 bins của véc tơ bins ta thu được biểu đồ Histogram of Gradients như hình 2.7.

2.4 Chuẩn hóa Block

Chúng ta nhóm các cell lại thành các khối chồng lên nhau (overlapping), mỗi khối có kích thước 2x2 cell và các cell có kích thước 8x8 pixel như đã đề cập tại 2.3. Hai khối liên tiếp theo chiều ngang và chiều dọc chồng lên nhau bởi 2 cell như mô tả tại hình 2.8. Kết quả là mỗi cell sẽ được bao phủ bởi 4 block.

Sau khi có được Cell feature vector (2.3), ta sẽ ghép các đặc trưng này thành một khối duy nhất



Hình 2.7: Biểu đồ Histogram of Gradient gồm 9 bins tương ứng với một ô vuông trong lưới ô vuông.

được gọi là block feature bằng cách kết nối các cell histograms của 4 cell có trong mỗi block. Sau đó ta chuẩn hóa các block feature này (normalization block) bằng *Euclidean norm* của chính nó.

Một số phương pháp chuẩn hóa [1]:

- L2-norm:
$$v = \frac{v_2}{\sqrt{||v_2||^2 + e^2}} \tag{2.8}$$
- L1-norm:
$$v = \frac{v_1}{||v_1|| + e} \tag{2.9}$$
- L1-sqrt:
$$v = \sqrt{\frac{v_1}{||v_1|| + e}} \tag{2.10}$$

Cụ thể hơn ta sẽ phân tích chuẩn hóa theo L2-norm. Đầu tiên ta có một vector feature block với 36 features như sau:

$$V = [a_1, a_2, a_3, a_4, \cdots, a_{36}]$$

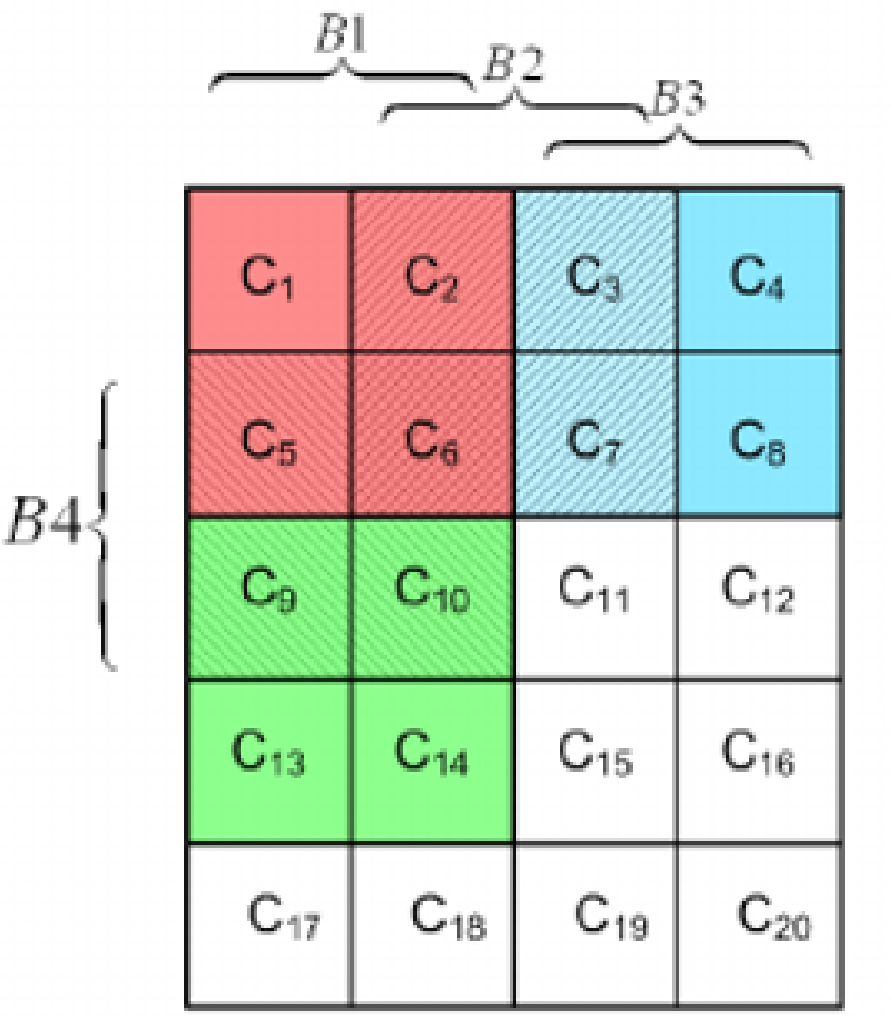
Tính căn bậc 2 của tổng bình phương các features:

$$k = \sqrt{{a_1}^2 + {a_2}^2 + {a_3}^2 + \cdots + {a_{36}}^2}$$

Sau đó lấy k chia cho các feature của vector V ta thu được:

$$\text{Normalised Vector} = \left(\frac{a_1}{k}, \frac{a_2}{k}, \frac{a_3}{k}, \ldots, \frac{a_{36}}{k}\right)$$

Ta thấy ở công thức (2.8) có hằng số e được thêm vào, hằng số này được sử dụng để đảm bảo mẫu số khác 0 khi thực hiện phép tính.



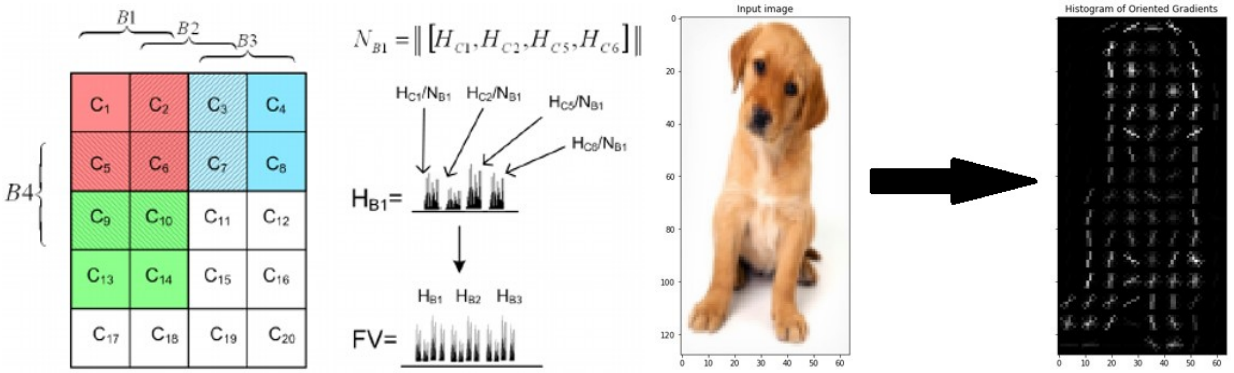
Hình 2.8: Các Khối block chồng lên nhau (overlapping)

2.5 Vector đặc trưng HOG

Sau khi chuẩn hóa các véc tơ histogram, chúng ta sẽ kết nối (*concatenate*) các véc tơ 1x36 này thành một véc tơ lớn, đây chính là véc tơ HOG đại diện cho toàn bộ hình ảnh như mô tả tại hình 2.9a.

Ví dụ: Hình ảnh của chúng ta được chia thành lưới ô vuông kích thước 16x8 (mỗi ô 8x8). Qúa trình tính toán HOG sẽ di chuyển 7 lượt theo chiều rộng và 15 lượt theo chiều cao. Như vậy sẽ có tổng cộng 7x15=105 patches, mỗi patch tương ứng với 1 véc tơ histograms 36 chiều. Do đó cuối cùng véc tơ HOG sẽ có kích thước là 105x36=3780 chiều. Đây là một véc tơ kích thước tương đối lớn nên có thể mô phỏng được đặc trưng của ảnh khá tốt.

Kết quả của phương pháp với đầu vào là một hình ảnh chứa đối tượng cần xác định và đầu ra là hình ảnh chứa các đặc điểm về cấu trúc và góc cạnh biểu diễn hình dạng của vật thể như hình 2.9b.



(a) Kết nối các normalised vector của các block thành HOG Feature Vector

(b) Ảnh HOG biểu diễn đặc trưng vật thể trong ảnh.

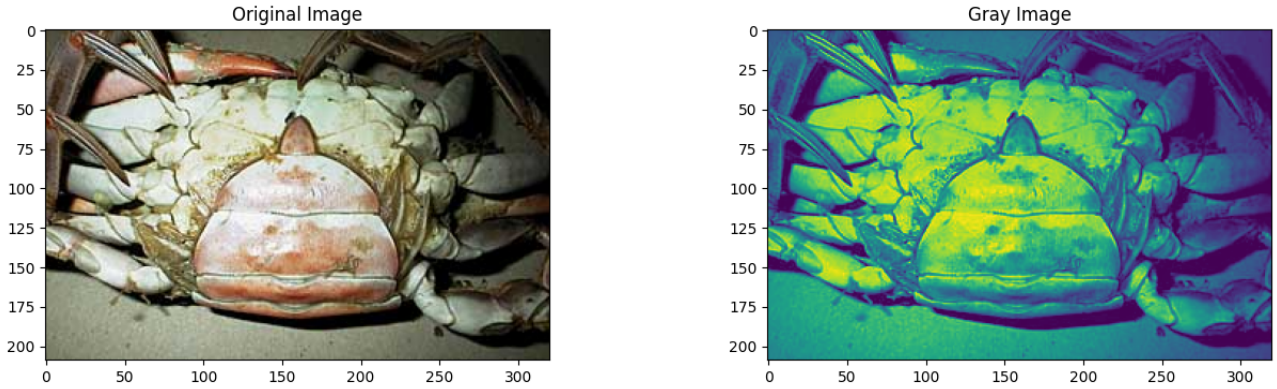
3 Ứng dụng HOG để trích xuất đặc trưng cho bộ ảnh Cua (kèm code)

```
import numpy as np
import cv2
import matplotlib.pyplot as plt

img = plt.imread('/content/CrabFemale.jpg', cv2.IMREAD_UNCHANGED)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
print('image shape:', img.shape)
print('gray shape: ', gray.shape)

plt.figure(figsize = (16, 4))
plt.subplot(1, 2, 1)
plt.imshow(img)
plt.title('Original Image')
plt.subplot(1, 2, 2)
plt.imshow(gray)
plt.title('Gray Image')
```

```
image shape: (209, 320, 3)
gray shape:  (209, 320)
Text(0.5, 1.0, 'Gray Image')
```

```
# Calculate gradient gx, gy
gx = cv2.Sobel(gray, cv2.CV_32F, dx=0, dy=1, ksize=3)
gy = cv2.Sobel(gray, cv2.CV_32F, dx=1, dy=0, ksize=3)
```

```
print('gray shape: {}'.format(gray.shape))
print('gx shape: {}'.format(gx.shape))
print('gy shape: {}'.format(gy.shape))
```



Hình 3.1: Kết quả.

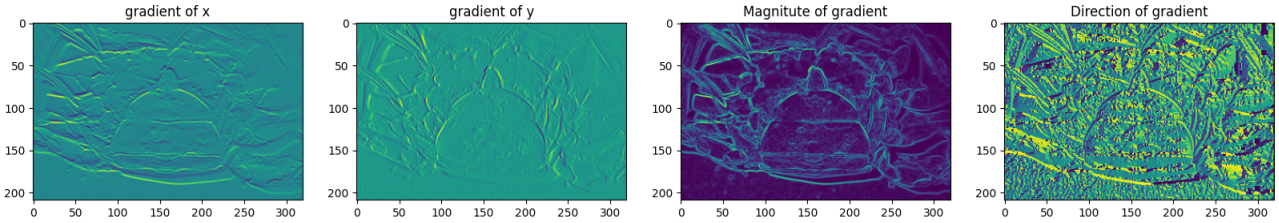
```
gray shape: (209, 320)
gx shape: (209, 320)
gy shape: (209, 320)
```

```
g, theta = cv2.cartToPolar(gx, gy, angleInDegrees=True)
print('gradient format: {}'.format(g.shape))
print('theta format: {}'.format(theta.shape))
```

```
gradient format: (209, 320)
theta format: (209, 320)
```

```
w = 20
h = 10

plt.figure(figsize=(w, h))
plt.subplot(1, 4, 1)
plt.title('gradient of x')
plt.imshow(gx)

plt.subplot(1, 4, 2)
plt.title('gradient of y')
plt.imshow(gy)

plt.subplot(1, 4, 3)
plt.title('Magnitute of gradient')
plt.imshow(g)

plt.subplot(1, 4, 4)
plt.title('Direction of gradient')
plt.imshow(theta)
```

```
<matplotlib.image.AxesImage at 0x7d484d22d570>
```



Hình 3.2: Kết quả

```
print('Original image size: ', gray.shape)

# 1. Declare parameters
cell_size = (8, 8)  # h x w in pixels
block_size = (2, 2)  # h x w in cells
nbins = 9  # number of orientation bins

# 2. Calculate the parameters passed to HOGDescriptor
# winSize: The size of the image is cropped to be divisible by the cell size.
winSize = (gray.shape[1] // cell_size[1] * cell_size[1], gray.shape[0] // cell_size[0] * cell_size[0])
# blockSize: Size of 1 block
blockSize = (block_size[1] * cell_size[1], block_size[0] * cell_size[0])
# blockStride: Number of moves of the block when performing histogram normalization step 3
blockStride = (cell_size[1], cell_size[0])
print('Size of cropped image according to winSize (pixel): ', winSize)
print('Size of a block (pixel): ', blockSize)
print('Size of block stride (pixel): ', blockStride)
# 3. Compute HOG descriptor
hog = cv2.HOGDescriptor(_winSize=winSize,
                        _blockSize=blockSize,
                        _blockStride=blockStride,
                        _cellSize=cell_size,
                        _nbins=nbins)

# Size of the square grid.
n_cells = (gray.shape[0] // cell_size[0], gray.shape[1] // cell_size[1])
print(Size of the square grid (square grid): ', n_cells)

```

```
# Reshape hog feature
hog_feats = hog.compute(gray)\
                .reshape(n_cells[1] - block_size[1] + 1,
                         n_cells[0] - block_size[0] + 1,
                         block_size[0], block_size[1], nbins) \
                .transpose((1, 0, 2, 3, 4))

print('Size hog feature (h, w, block_size_h, block_size_w, nbins) : ', hog_feats.shape)
```

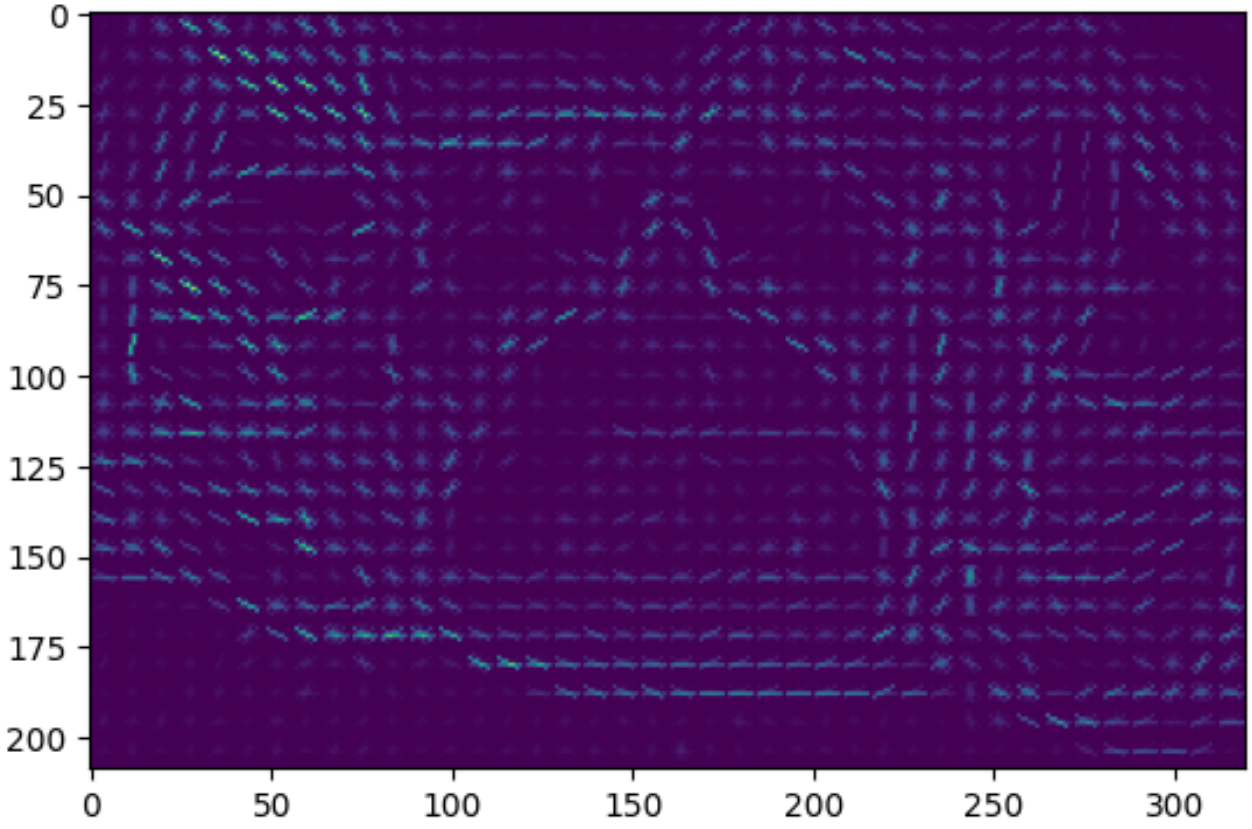
```
Original image size: (209, 320)
Size of cropped photo according to winSize (pixel): (320, 208)
Size of 1 block (pixel): (16, 16)
Size of block stride (pixel): (8, 8)
Grid size (squares): (26, 40)
Hog feature size (h, w, block_size_h, block_size_w, nbins): (25, 39, 2, 2, 9)
```

```
from skimage import feature
H = feature.hog(gray, orientations=9, pixels_per_cell=(8, 8),
                cells_per_block=(2, 2), transform_sqrt=True,
                   block_norm="L2")

print('Size hog features: ', H.shape)
```

```
Size hog features:  (35100,)
```

```
from skimage import exposure

(H, hogImage) = feature.hog(gray, orientations=9, pixels_per_cell=(8, 8),
    cells_per_block=(2, 2), transform_sqrt=True, block_norm="L2",
    visualize=True)

hogImage = exposure.rescale_intensity(hogImage, out_range=(0, 255))
hogImage = hogImage.astype("uint8")

plt.imshow(hogImage)
```

```
<matplotlib.image.AxesImage at 0x7d484d1680a0>
```



Hình 3.3: Kết quả

- Link chạy code trên google colab: Click here.
- Link code python: Click here.
- Link data với 3 ảnh test (bộ ảnh về cua): Click here.

4 Tài liệu tham khảo

[1] Tomasi, Carlo. "Histograms of oriented gradients." Computer Vision Sampler (2012): 1-6.

[2] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). Vol. 1. Ieee, 2005.

[3] Nguồn internet: phamdinhkhanh.github.io