Histogram of Oriented Gradients (HOG)

HOG descriptors: are mainly used to describe the structural shape and appearance of an object in an image, making them excellent descriptors for object classification. However, since HOG captures local intensity gradients and edge directions, it also makes for a good texture descriptor.

Details of the algorithm to compute HOG:

1. Load image and normalize it.
2. Compute gradient (vertical, horizontal) using Sobel filter.
3. Compute magnitude of gradient.
4. Compute direction of gradient.
5. Partition image to number of cells and Compute the histogram to each cell.
6. Visualize the histogram of each cell.
7. Partition image to number of blocks.
8. Compute block normalization.
9. Calculate the HOG feature vector.

Start processes:

1- Load image and normalize it.

(1) read image with size (128 * 64)

(2) convert image from RGB to Gray

(3) normalize image by divide on 255

Note: should the size of image is (128* 64)

2- Compute gradient (vertical, horizontal) using Sobel filter.

(1) apply a convolution operation to obtain the gradient image (X,Y) using (Sobel) filter

- **Sobel filter :**

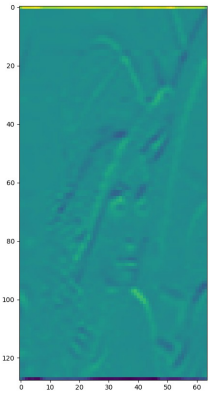
D_x

-1	0	1

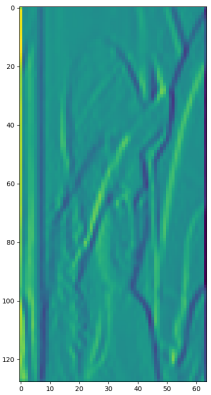
D_y

-1
0
1

$G_x = I \star D_x$



$G_y = I \star D_y$

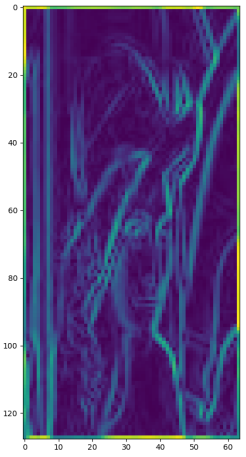


Note: I is the input image, D_x is our filter in the *x*-direction, and D_y is our filter in the *y*-direction.

3- Compute magnitude of gradient.

(1) calculate the magnitude of gradient using gradient (vertical, horizontal)

$$|G| = \sqrt{G_x^2 + G_y^2}$$

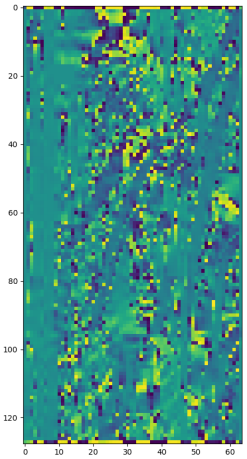


4- Compute direction of gradient.

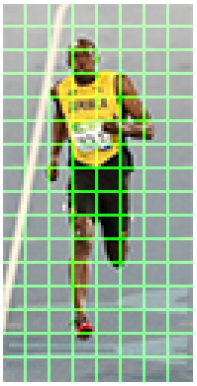
(1) calculate the orientation of the gradient for each pixel in the input image
(2) convert the angle from readies to degree
(3) map the angle to range (0-180) degree by get angle % 180

- tan^-1 (Gy / Gx) =>

$$\theta = arctan2(G_y, G_x)$$



Note: in coding don't use (arctan) because in some situation will return NAN when the magnitude_X = Zero, so using (arctan2).



5- Partition image to number of cells and Compute the histogram to each cell.

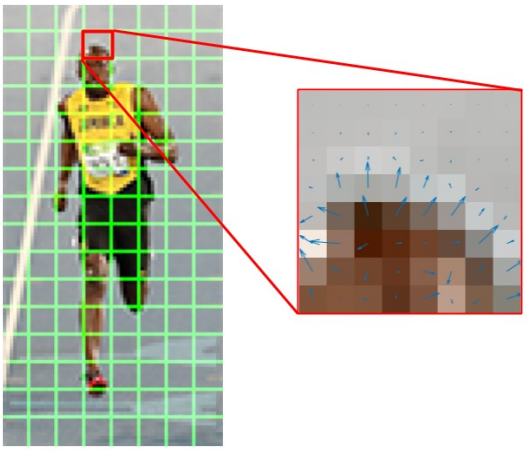
(1) partition image to cells each cell is (8*8) pixels

- **"cell" :** is a rectangular region defined by the number of pixels that belong in each cell.

(2) the histogram of 9 bins : [0, 20, 40, 60, 80, 100, 120, 140,160]

(3) calculate histogram to each cell (8 * 8)

- each pixel in cell have the magnitude
  and direction



2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

- A bin is selected based on the **direction**

  1- find difference between direction and
  the values of histogram of bin.

  2- find the index of min difference bin is the first bin belong to it .

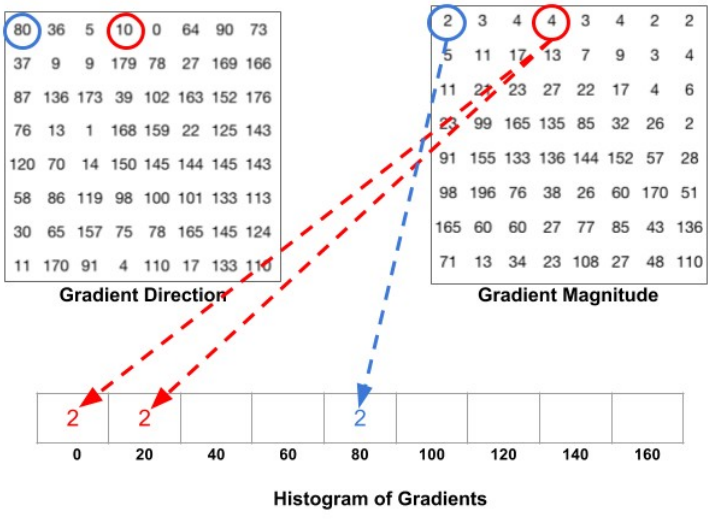
  3- find min value of (first_index -1 , first_index -1) is the second bin belong to it.

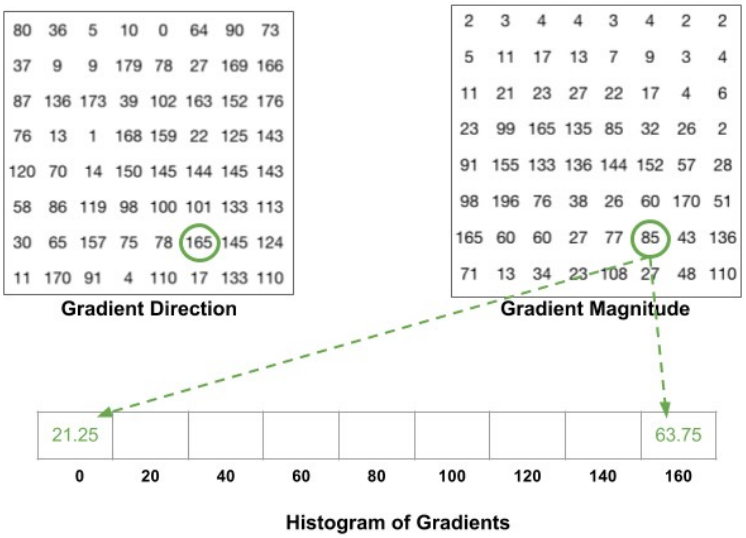
- the value that goes into the bin is selected based on the **magnitude**

  1- based on the bins which direction belong them will partition the value of magnitude

  2- distribution the magnitude based on the probability degree of near the direction or far with the
  bins which belong to them.

- **Example:**
  1- assume that direction = 45’
  2- the first bin belong it is = 40’
  3- the second may is (20 || 60)
  4- using the min difference(45-20 , 60-45) the second bin is = 60’
  5- distribution the magnitude on (40 & 60)
  6- compute the probability of first bin is: 1- [(45-40)/(180/9)] = 0.75
  7- multiply the probability in value of magnitude and save the value in bin.





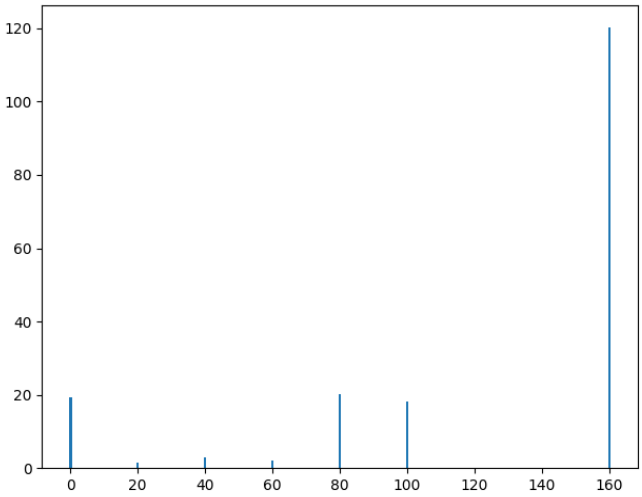
Note:

1- I use size of cell (8 * 8) but can change it based on features we are looking for.
2- in the paper distribution the magnitude on only one bin (first bit),
this solution was not a good thing.

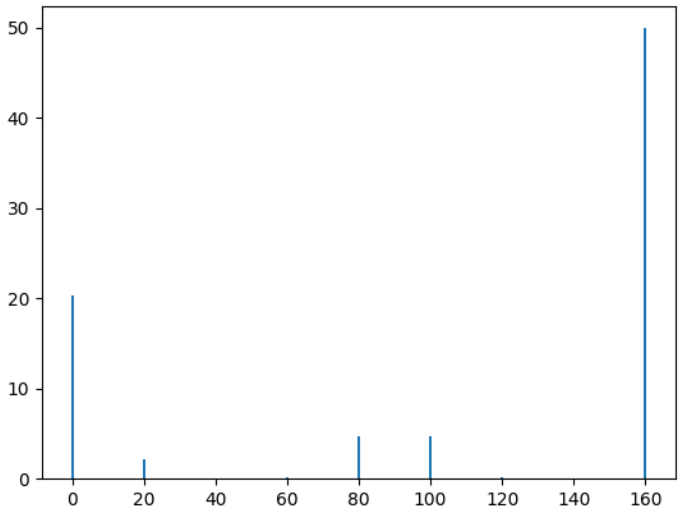
6- Visualize the histogram of each cell.

(1) show the histogram of each cell after map direction to histogram
of 9 bins : [0, 20, 40, 60, 80, 100, 120, 140,160] .

histogram first cell



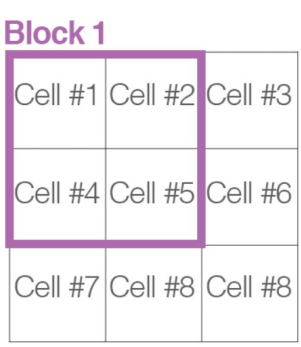
histogram second cell

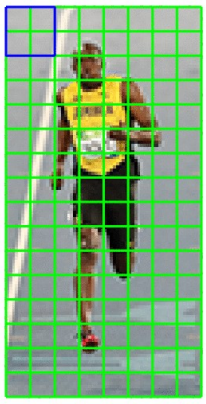


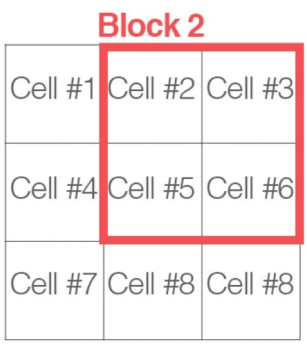
7- Partition image to number of blocks.

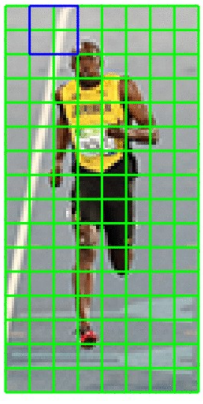
- Because Gradients of an image are sensitive to overall lighting.
- Partition image to blocks and normalize it .

(1) **Block :** grouping the “cells” together , is (2 * 2) cell
(2) Blocks overlap with together , each cell contributes to the final feature vector more than once.









Note: Dalal and Triggs report that using either (2 x 2) or (3 x 3) cells_per_block obtains reasonable accuracy in most cases.

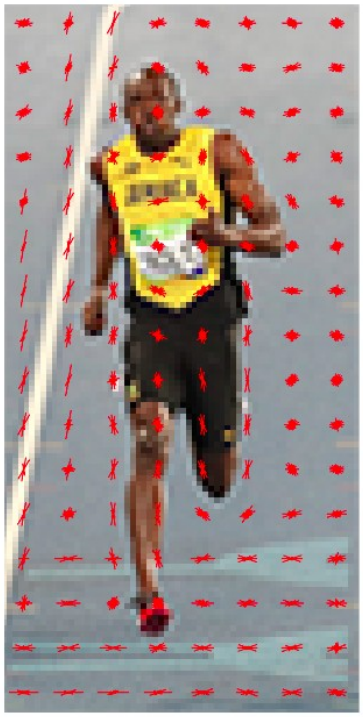
8- Compute block normalization.

(1) after partition image to blocks become (15 *7) = 105 blocks
(2) each block contains on 4 cells = 36 value
(3) each cell contains 1 histogram bin
(4) each histogram contains 9 value
(5) normalization each block using (L2 Norm)

- apply **(L2 Norm)** on block vector (36*1) :

$$Norm_{L2} = \sqrt{\sum_{n=0}^{len-1} |pSrc[n]|^2}$$

- **normalize vector =** (vector / L2norm)



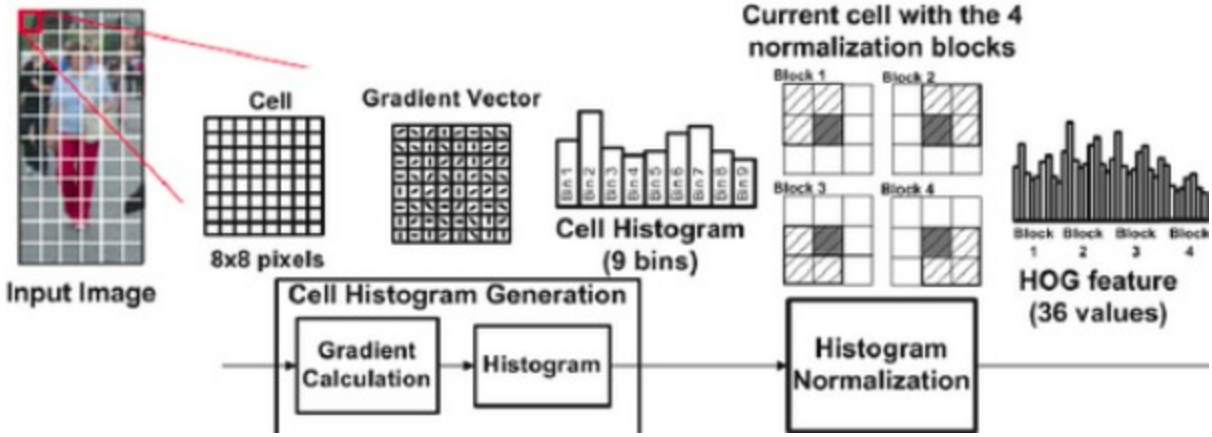
9- Calculate the HOG feature vector.

(1) in final image have (15 * 7) blocks

(2) each block have (36*1) normalized value

(3) concatenate them all into one giant vector
we obtain a 36×105 = **3780** dimensional vector.

(4) this vector contains on the HOG features (3780).



important information

(1) Time

- **Using Function timeit.default_timer().**

  - Returns the current time instant, a floating-point number of seconds since the epoch.

type	Average time (seconds)
Read image	0.0106
Calculate HOG features	0.333 (with visualize image in function = 6.325)
calculate_cells_histogram	0.315
calculate_block_normalization	0.004

(2) Convert code from sequential to parallel by using multi-threading

- By using machine have this information:

Information Computer	
Processor	Up to 8th Gen Intel® Core™ i7 Processor
Graphics	- Intel Integrated Graphics - NVIDIA® GeForce® 940MX(4GB)
Memory Ram	16 GB
Thread(s) per core	2
Core(s) per socket	4
Socket(s)	1
CPU(s) = Thread(s) per core X Core(s) per socket X Socket(s)	8

- **Can do the convert program to parallel and using threading model by Number_Of_Thread = 8**

  (1) partition the image to 8 part , distribute each part to thread, concatenate all HOG features in one vector of features.

  (2) can change part of calculate histogram of cells to work on many thread.

(3) Parameters tuning (effect of parameters on memory/speed and performance)

effect of parameters	memory	Time	performance
Size of image	√	√	?
Number of bins	√	√	√
Number of cells	√	Depend on **Size of image**	√

"?" → should try doing to know if effect or not.

(4) programing language

Python 3	
Library	version
numpy	1.13.3
scipy	1.1.0
matplotlib	3.0.2

(5) OS: Ubuntu 16.04

(6) Reference

- **https://www.learnopencv.com/histogram-of-oriented-gradients/**
- **https://gurus.pyimagesearch.com/lesson-sample-histogram-of-oriented-gradients-and-car-logo-recognition/#**
- **http://cs.brown.edu/people/pfelzens/papers/lsvm-pami.pdf**
- **https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf**