VIT-AP UNIVERSITY

Computer Vision

(Course Code: 4047)

Module-2:Lecture-6: Histograms of Oriented Gradients (HoG)

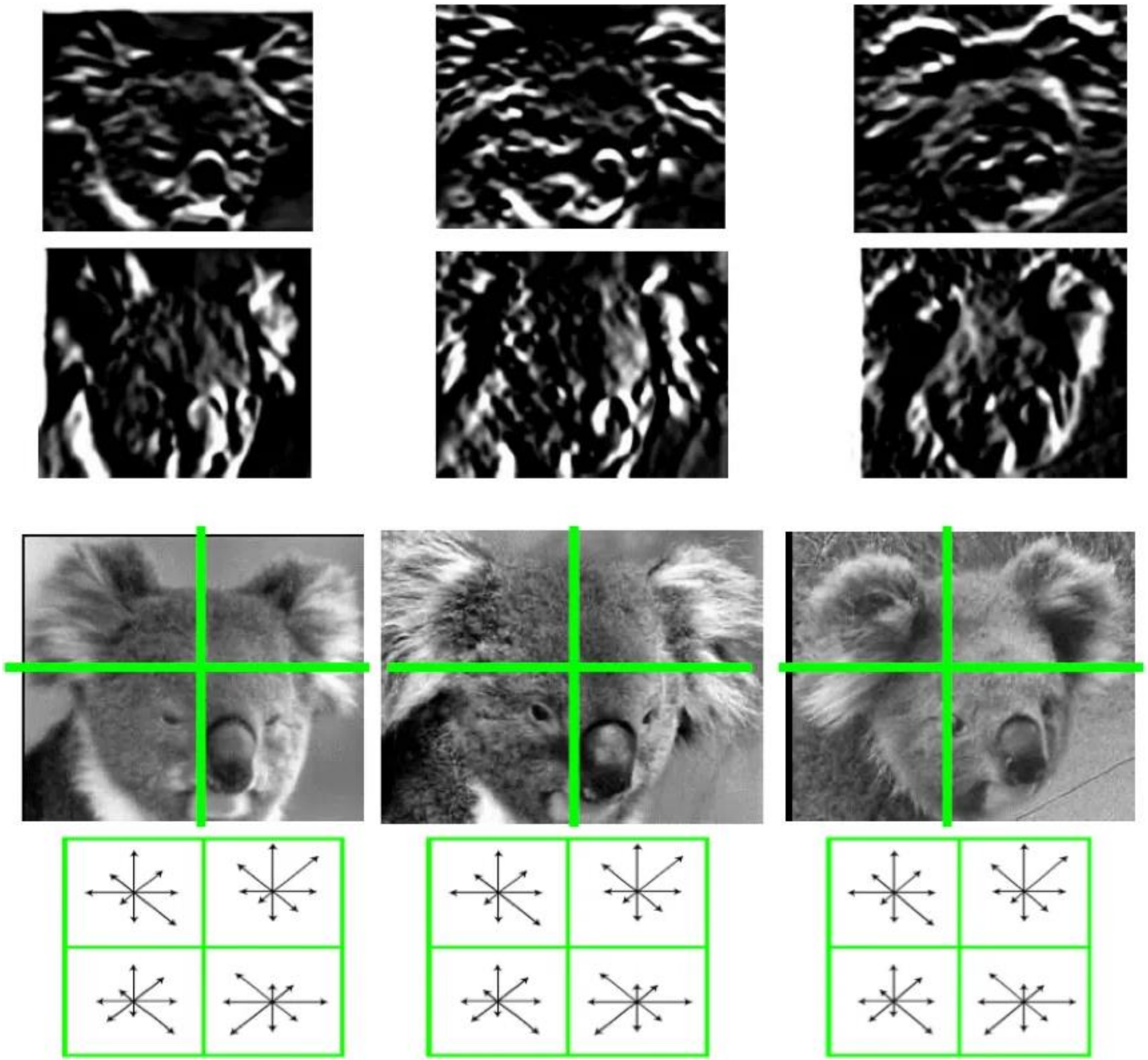
Gundimeda Venugopal, Professor of Practice, SCOPE

Gradient-based Representations

- ❖ Gradient-based representations are vector representations composed of edge, contour, and intensity gradient information
- ❖ The gradient-based approaches are decent because they summarize the local distribution of gradients with histograms, where the term ‘local’ refers to the equally-divided image patches (regions divided by green line in figure).

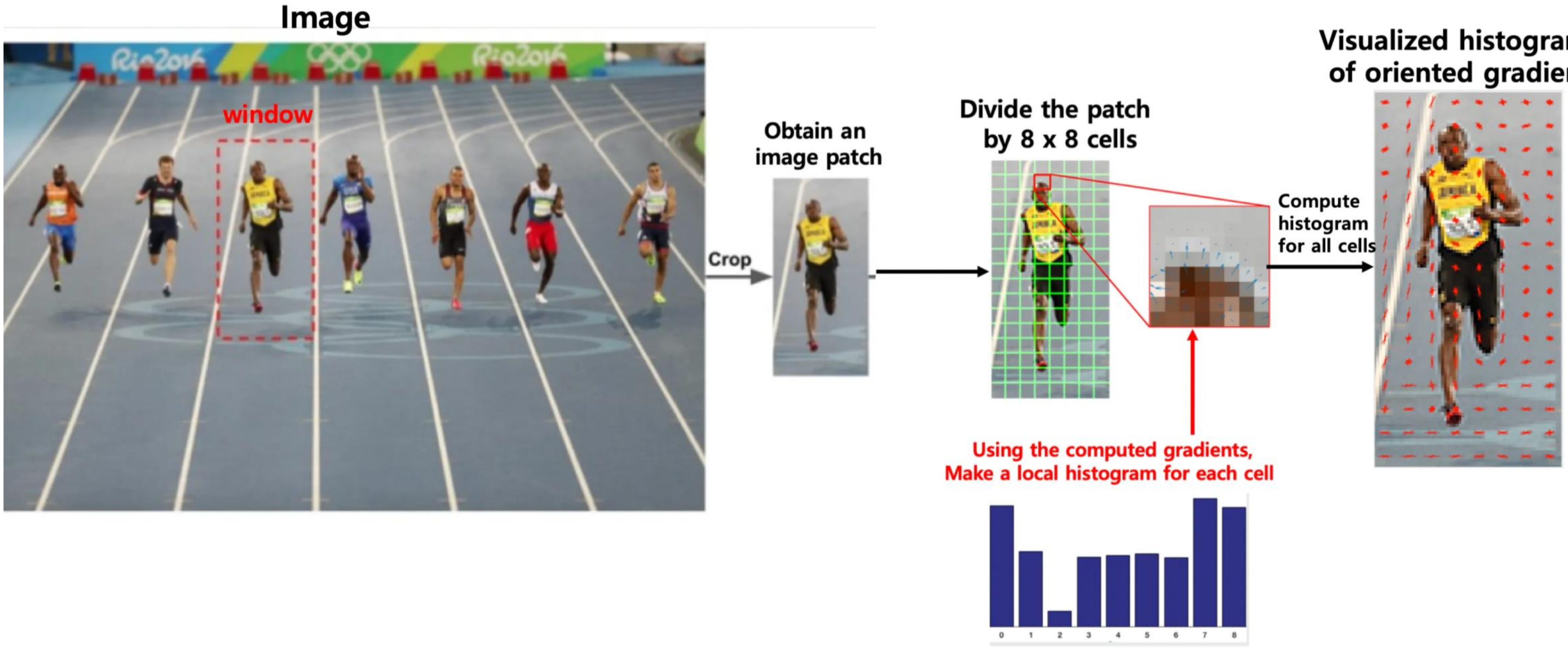
❖ Advantages of gradient-based representations:

- ➢ Locally orderless: invariance to small shifts and rotations
- ➢ Localized histograms from local distributions (distributions from each grid in green line in Figure) offers more spatial information compared to a single global histogram (distribution of complete image)
- ➢ Includes contrast normalization: reduce the impact of variable illumination (color)

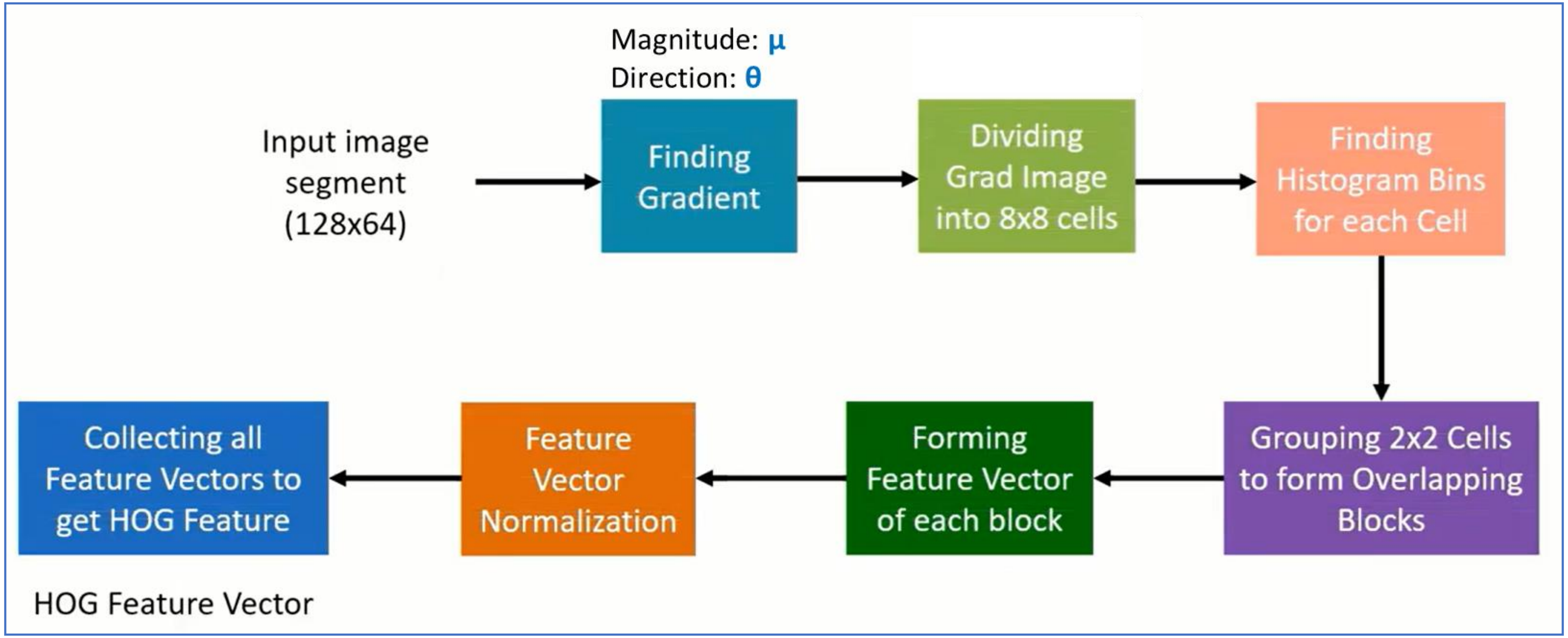


Histograms of Oriented Gradients (HoG)

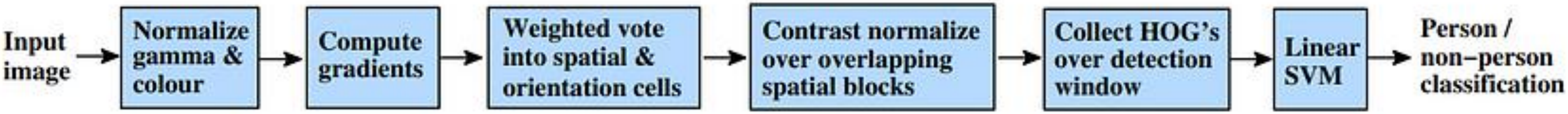
- Idea behind the Histogram of oriented gradients descriptor is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions.
- The '*Histograms of Oriented Gradients (HoG)*' is one of the representative feature descriptors with gradient-based representations and it is commonly used to extract strong features for object detection.
- How HoG works to extract local features from a given image patch, which is a part of the image overlapped with the sliding window.



HOG: How it works?



Histograms of Oriented Gradients for Human Detection



HOG key ideas

- Gradients

  [-1 0 1] and $[-1\ 0\ 1]^T$ were good enough.

- Cell Histograms

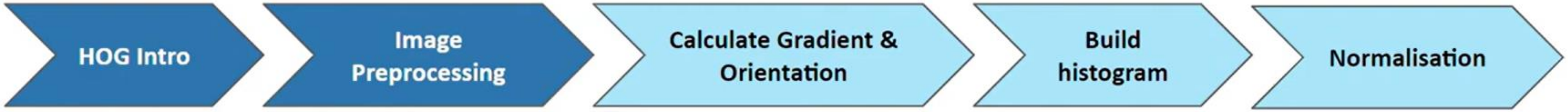
  Each pixel within the cell casts a weighted vote for an orientation-based histogram channel based on the values found in the gradient computation. (9 channels worked)

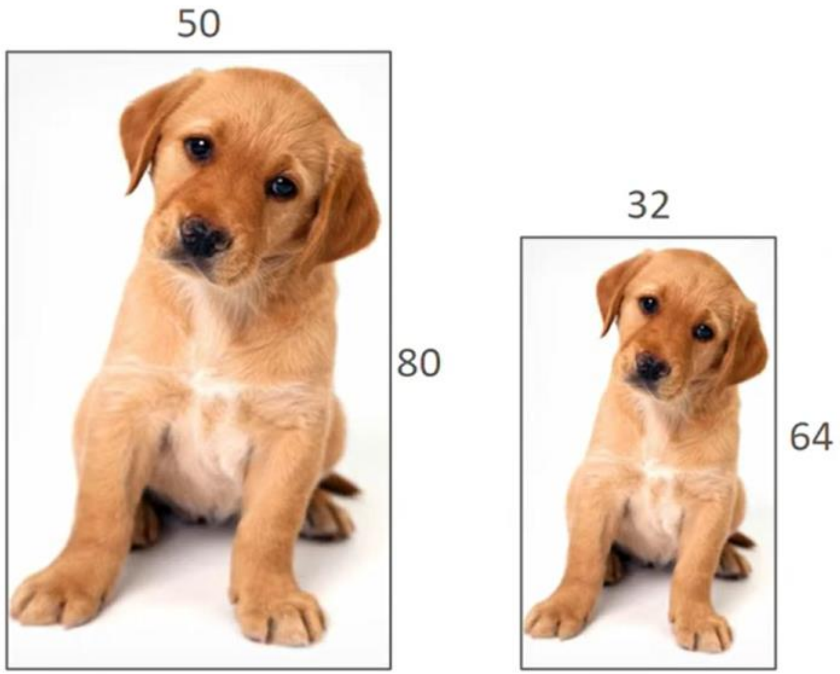
- Blocks

  Group the cells together into larger blocks, either R-HOG blocks (rectangular) or C-HOG blocks (circular).

HOG Intuition

Intuition: HOG Feature Descriptor





Original Image   Resized Image

- Resized image size: 32x64
- HOG Cell size: 4x4
- Recommended:
  - Image size: 64x128
  - cell size: 8x8 / 16x16

Getting the Image Segment + Calculate Gradients

Intuition: HOG Feature Descriptor

HOG Intro → Image Preprocessing → Calculate Gradient & Orientation → Build histogram → Normalisation

P32 | C8

P64
C16

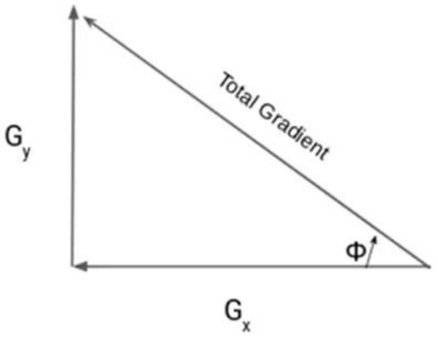
Select a 4 x 4 cell

Pixel values for selected cell

121	10	78	96
48	152	64	125
145	78	85	89
154	214	56	200

For edge pixels, HOG uses padding

From Pythagoras theorem



Gradient in x dir. (Gx)
= 89 - 78
= 11

Gradient in y dir. (Gy)
= 64 - 56
= 8

Total Gradient Magnitude
$= \sqrt{[(Gx)^2+(Gy)^2]}$
$= \sqrt{[(11)^2+(8)^2]}$
= 13.6

Orientation (Φ)
= atan(Gy / Gx)
= 36 deg.

$[-1\ 0\ 1]$ and $[-1\ 0\ 1]^T$
Central Difference

Forward Difference: $f'(x) = \dfrac{f(x+h) - f(x)}{h}$

Backward Difference: $f'(x) = \dfrac{f(x) - f(x-h)}{h}$

Central Difference: $f'(x) = \dfrac{f(x+h) - f(x-h)}{2h}$

First Order Derivative Kernel for detecting edges

Approximate the two components I_x and I_y of the gradient of I by central differences:

$I_x(r, c) = I(r, c + 1) - I(r, c - 1)$ and $I_y(r, c) = I(r - 1, c) - I(r + 1, c)$.

Central difference requires division by 2, but this constant can be ignored because of subsequent normalizations.

The gradient is then transformed to polar coordinates, with the angle constrained to be between 0 and 180 degrees, so that gradients that point in opposite directions are identified:

$$\mu = \sqrt{I_x^2 + I_y^2} \quad \text{and} \quad \theta = \frac{180}{\pi}\left(\tan_2^{-1}(I_y, I_x) \mod \pi\right)$$

where $\tan_2^{-1}$ is the four-quadrant inverse tangent, which yields values between $-\pi$ and π.

Intuition: HOG Feature Descriptor

HOG Intro → Image Preprocessing → Calculate Gradient & Orientation → Build histogram → **Normalisation**

Total Gradient Magnitude = 13.6

Orientation (Φ) = 36 deg.

(40-36)/20

(36 - 20)/20

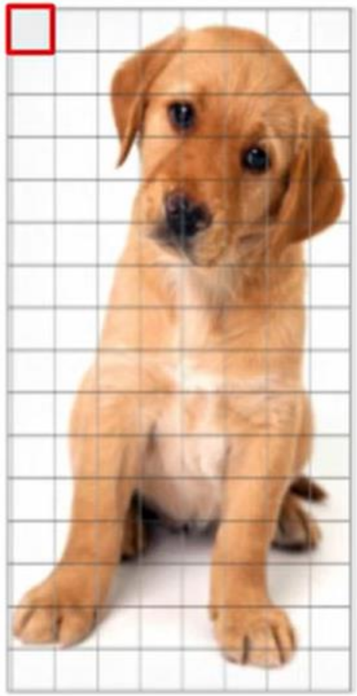
Magnitude		(4/20)*13.6	(16/20)*13.6						
Bin	0	20	40	60	80	100	120	140	160
Features	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

Weight 16 4

20 36 40

P32 | C8

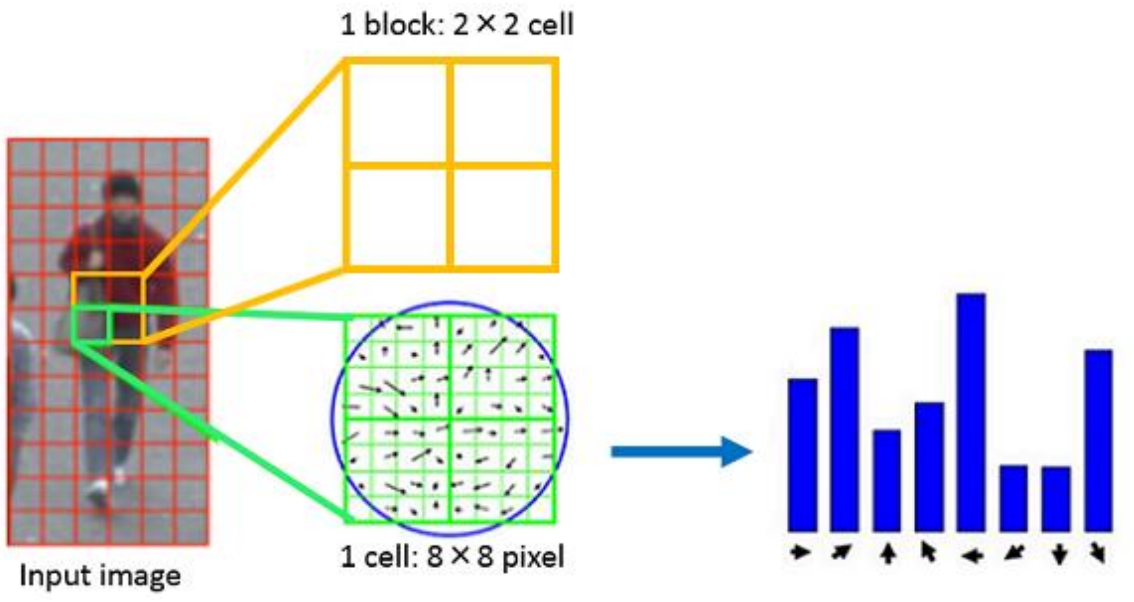
Matrix 1 x 9

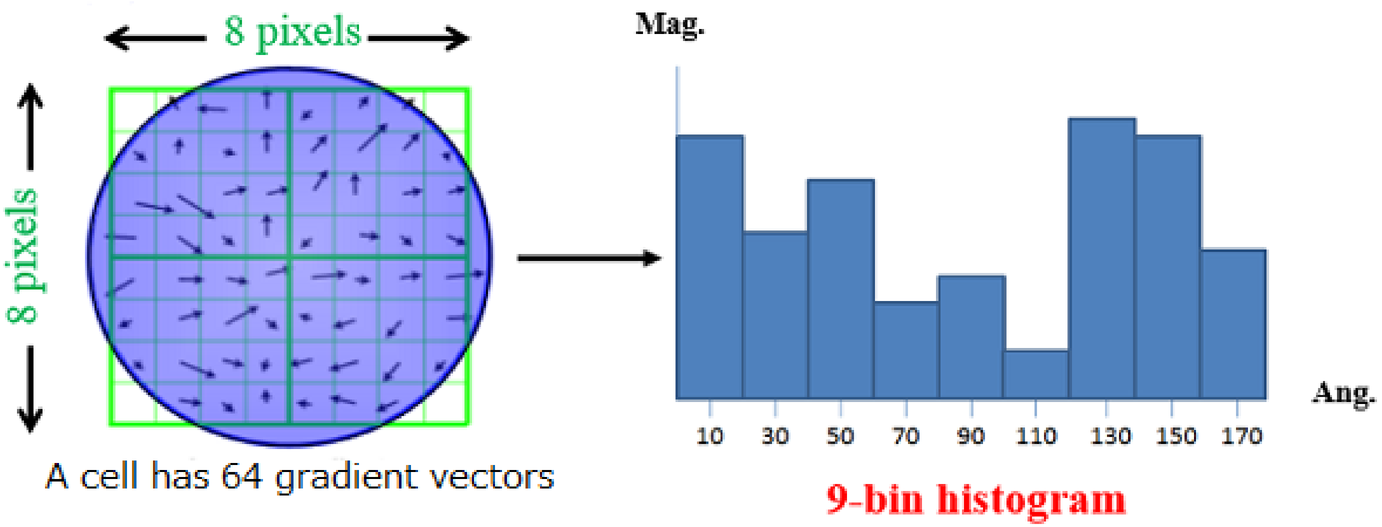


P64

C16

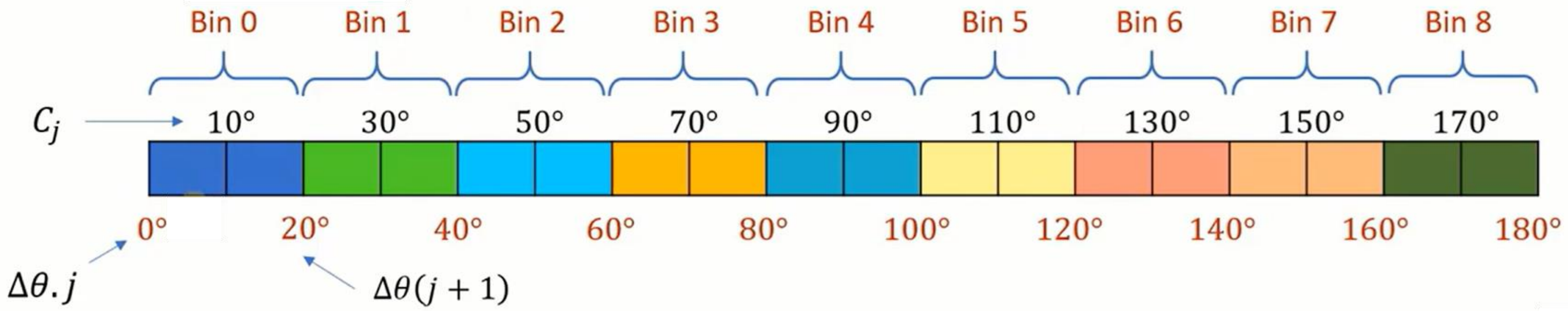
Histogram for gradient magnitude and orientation





Getting 9 point Histogram (Bins) for each cell (using Bi-linear Interpolation)

- Number of Bins = 9 (0 to 8) in range = 0° to 180°
- Step Size, $\Delta\theta = \frac{180°}{9} = 20°$
- Each j^{th} Bin will have boundaries = $[\Delta\theta . j, \Delta\theta(j+1)]$.
- and centre of j^{th} Bin, $C_j = \left[\frac{\Delta\theta . j + \Delta\theta(j+1)}{2}\right] = \Delta\theta\left(j + \frac{1}{2}\right)$



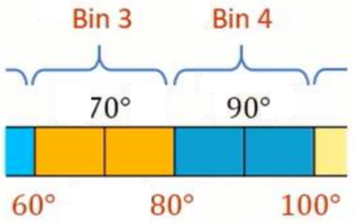
Getting 9 point Histogram (Bins) for each cell (using Bi-linear Interpolation)(2)

Steps:

- For a given value of angle θ (Obtained from gradient angle matrix), first we find value of j and $j+1$.
- Value of j is calculated by $j = floor\left(\left[\frac{\theta}{\Delta\theta} - \frac{1}{2}\right]\right)$, where $\Delta\theta = 20°$.
- Values assigned to j^{th}Bin is, $V_j = \mu\left[\frac{C_{j+1}-\theta}{\Delta\theta}\right]$
- Value assigned to $(j+1)^{th}$Bin is, $V_{j+1} = \mu\left[\frac{\theta - C_j}{\Delta\theta}\right]$
- The values assignment to two Bins is actually by Bi-linear interpolation.
- Sum of values assigned to two Bins will always be 1 i.e. $V_j + V_{j+1} = 1$.

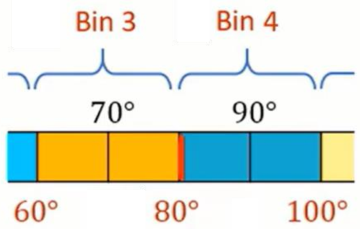
Getting 9 point Histogram (Bins) for each cell - Examples

Example 1

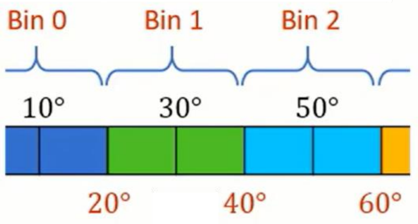


- Let $\theta = 77°$ (closer to Bin 3) and $\mu = 1$.
- Value of $j = floor\left(\left[\frac{\theta}{\Delta\theta} - \frac{1}{2}\right]\right) = floor\left(\left[\frac{77}{20} - \frac{1}{2}\right]\right) = floor(3.35) = 3$.
- Values assigned to 3rd Bin is, $V_3 = \mu\left[\frac{C_{j+1}-\theta}{\Delta\theta}\right] = 1\left[\frac{C_4-77}{20}\right] = 1\left[\frac{90-77}{20}\right] = 0.65$.
- Values assigned to 4th Bin is, $V_4 = \mu\left[\frac{\theta-C_j}{\Delta\theta}\right] = 1\left[\frac{77-C_3}{20}\right] = 1\left[\frac{77-70}{20}\right] = 0.35$.

Example 2 (θ is at Boundary of Bin)



- Let $\theta = 80°$ (on boundary of Bin 3 & 4) and $\mu = 1$.
- Value of $j = floor\left(\left[\frac{\theta}{\Delta\theta} - \frac{1}{2}\right]\right) = floor\left(\left[\frac{80}{20} - \frac{1}{2}\right]\right) = floor(3.5) = 3$.
- Values assigned to 3rd Bin is, $V_3 = \mu\left[\frac{C_{3+1}-\theta}{\Delta\theta}\right] = 1\left[\frac{C_4-80}{20}\right] = 1\left[\frac{90-80}{20}\right] = 0.5$.
- Values assigned to 4th Bin is, $V_4 = \mu\left[\frac{\theta-C_3}{\Delta\theta}\right] = 1\left[\frac{80-70}{20}\right] = 0.5$.
- Here also, $V_3 + V_4 = 0.5 + 0.5 = 1$.
- **This shows that if θ falls on any Bin boundary, then both the left and right Bins will get equal parts of μ.**



Example 3 (θ at Bin's Centre)

- Let $\theta = 30°$ (centre of Bin 1) and $\mu = 1$.
- Value of $j = floor\left(\left[\frac{\theta}{\Delta\theta} - \frac{1}{2}\right]\right) = floor\left(\left[\frac{30}{20} - \frac{1}{2}\right]\right) = floor(1) = 1$.
- Values assigned to 1st Bin is, $V_1 = \mu\left[\frac{C_{j+1}-\theta}{\Delta\theta}\right] = 1\left[\frac{C_2-30}{20}\right] = 1\left[\frac{50-30}{20}\right] = 1$.

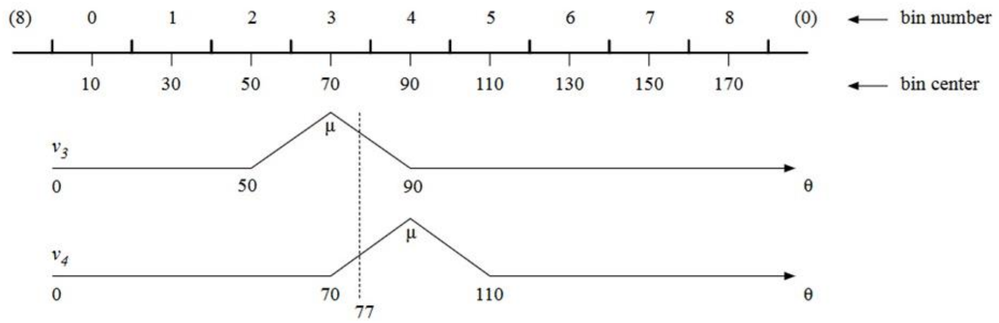
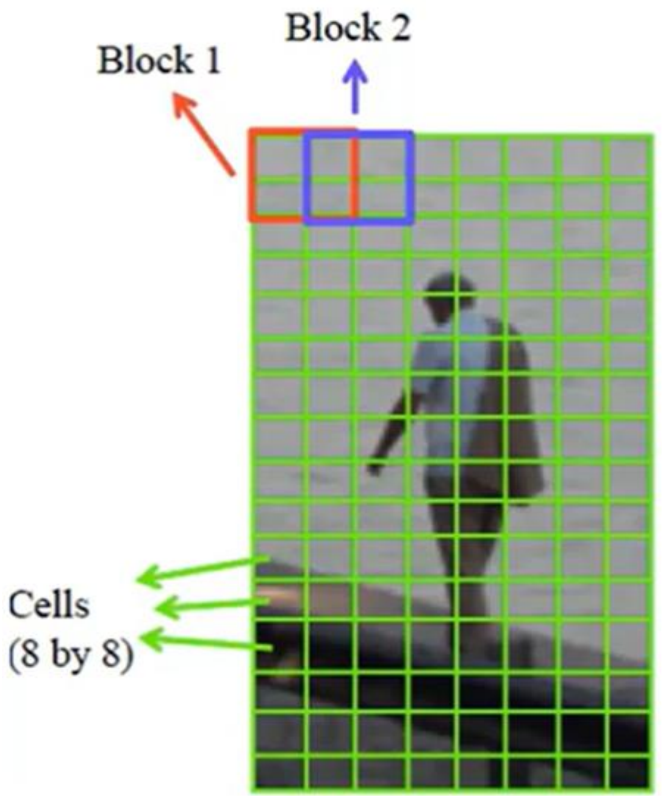


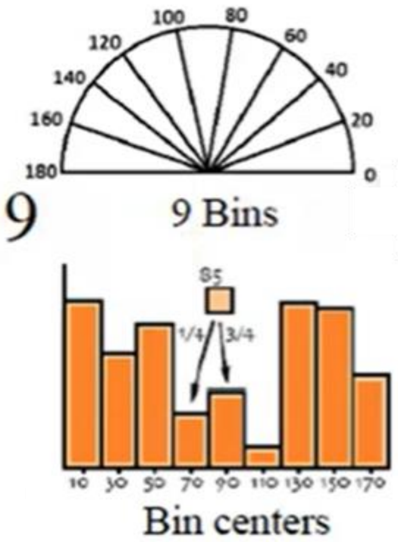
Figure : Voting by bilinear interpolation with $B = 9$ orientation bins. The two plots at the bottom show the votes v_3 and v_4 as the gradient orientation varies between 0 and 180 degrees. For instance, a gradient with orientation $\theta = 77$ degrees (dashed line) and magnitude μ contributes 0.65μ to bin 3 and 0.35μ to bin 4. The sum of the two contributions is always μ.

https://courses.cs.duke.edu/fall15/cps274/notes/hog.pdf

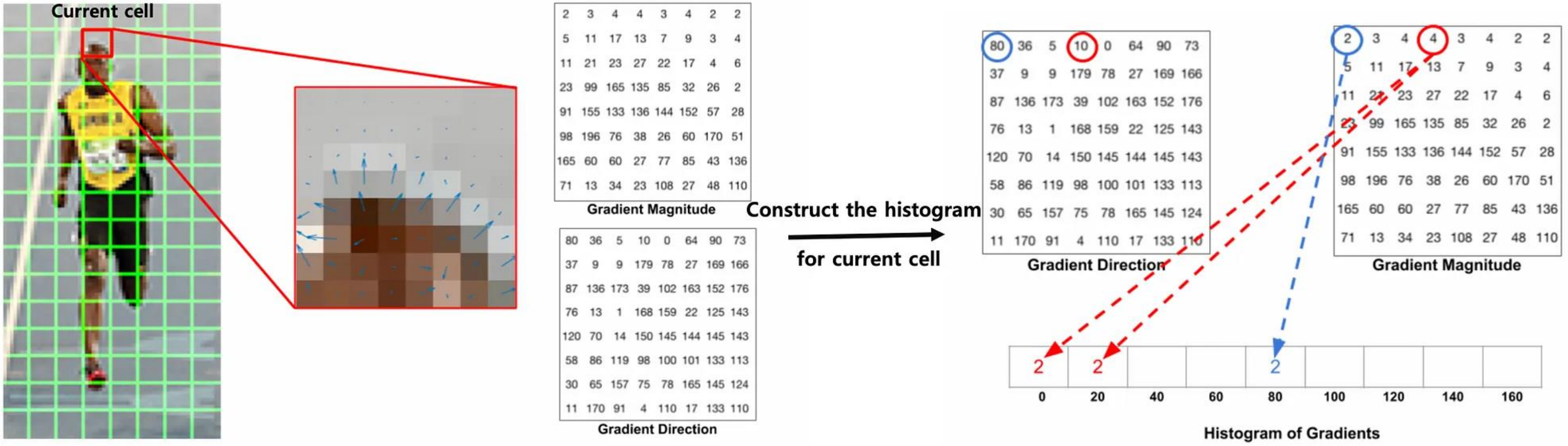
HOG Feature Descriptor: Voting



- Each block consists of 2x2 cells with size 8x8
- Quantize the gradient orientation into 9 bins (0-180)
  - The vote is the gradient magnitude



Weighted voting (with 8X8 Cell)



Intuition: HOG Feature Descriptor

HOG Intro → Image Preprocessing → Calculate Gradient & Orientation → Build histogram → Normalisation

- Gradients for highlighted Block (having 36 features):

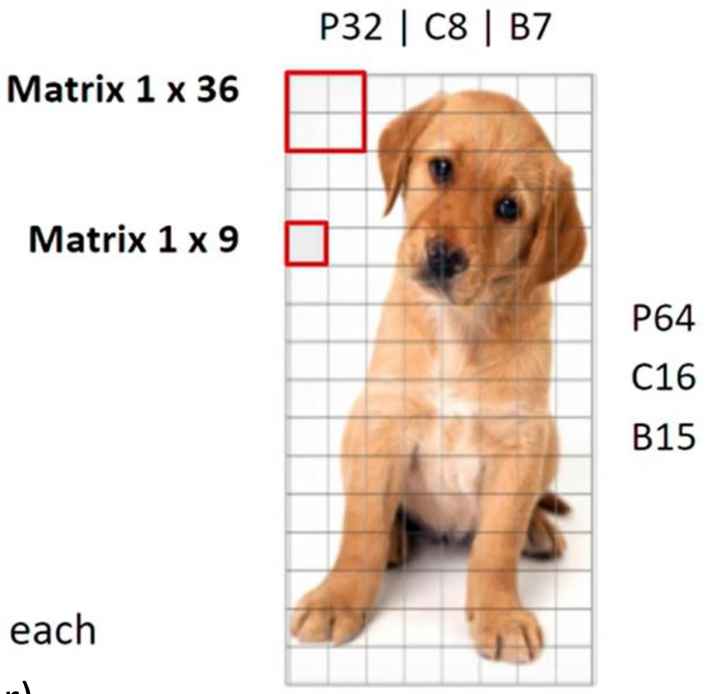
  $V = [a_1, a_2, a_3, \dots a_{36}]$

- Root of the sum of squares:

  $k = \sqrt{(a_1)^2 + (a_2)^2 + (a_3)^2 + \dots (a_{36})^2}$

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

- Normalised vectors created for all 105 blocks, having 36 features each
- Image **Feature Descriptor: 1 x 3780 matrix** (HOG Descriptor)



The HOG algorithm works by creating histograms of the distribution of gradient orientations in an image and then normalizing them in a very special way. This special normalization is what makes HOG so effective at detecting the edges of objects even in cases where the contrast is very low.

These normalized histograms are put together into a feature vector, known as the HOG descriptor, that can be used to train a machine learning algorithm, such as a Support Vector Machine (SVM), to detect objects in images based on their boundaries (edges). Due to its great success and reliability, HOG has become one of the most widely used algorithms in computer vison for object detection.

Making Overlapping Blocks from Cells

- Once Histogram (9 Bins) computation is over for all cells, then 4 cells (in 2×2) are clubbed together to form a Block.

- This clubbing is done in overlapping manner with stride of 8 pixels.

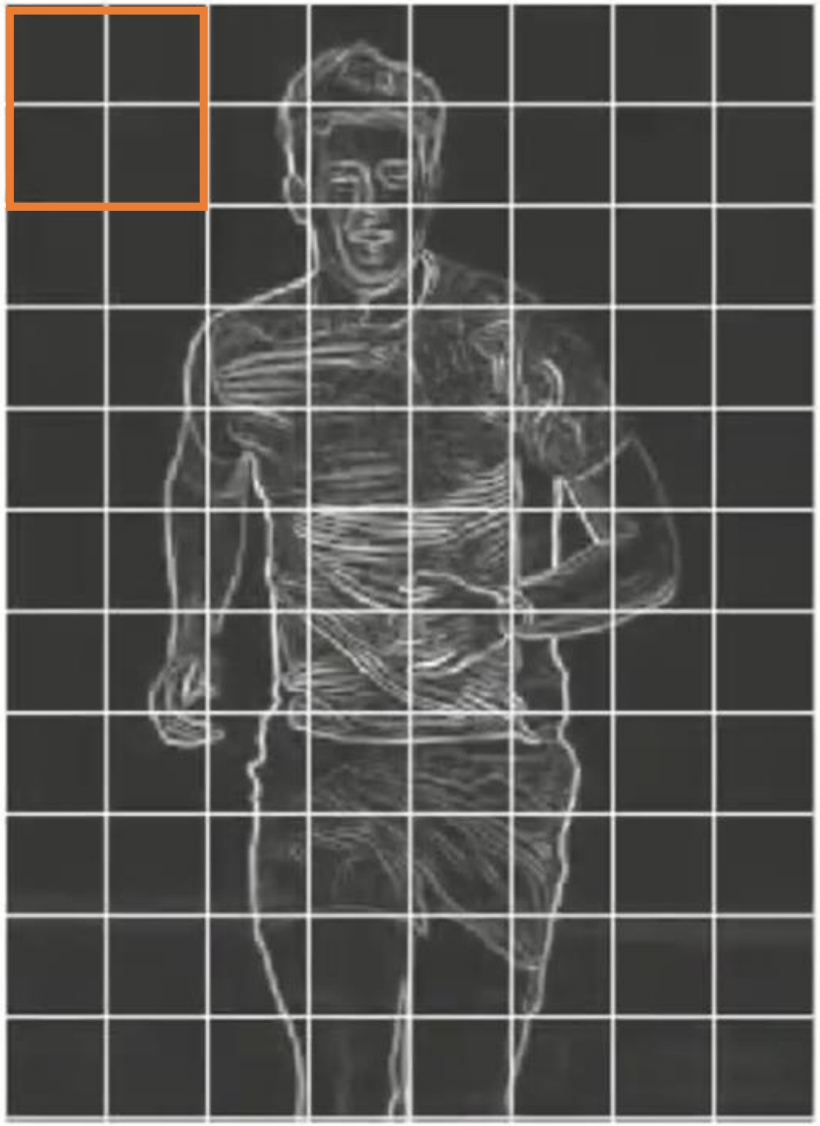
- For all 4 cells in a block, concatenate all 9-point histograms of each cell to form a 36-point feature vector, let represented as,

$$fb_i = [b_1, b_2, b_3, \dots \dots \dots \dots, b_{36}]$$

- Values of fb_i are normalized by L_2 norm as,

$$fb_i \leftarrow \frac{fb_i}{\sqrt{\|fb_i^2\| + \epsilon}}$$

(ϵ to deal with zero gradient regions)



- To achieve this normalization, first find,

$$k = \sqrt{b_1^2 + b_2^2 + b_3^2 + \cdots \dots \dots \dots \dots \dots + b_{36}^2}$$

then,

$$fb_i = \left[\frac{b_1}{k}, \frac{b_2}{k}, \frac{b_3}{k}, \dots \dots \dots \dots, \frac{b_{36}}{k}\right]$$

- This normalization is done to reduce the effect of changes in contrast between images of the same object.

Block Normalization Options

Dalal and Triggs tried 4 different kinds of normalization.
Let υ be the block to be normalized and e be a small constant.

L2-norm: $$f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}$$

L2-hys: L2-norm followed by clipping (limiting the maximum values of v to 0.2) and renormalizing,

L1-norm: $$f = \frac{v}{(\|v\|_1 + e)}$$

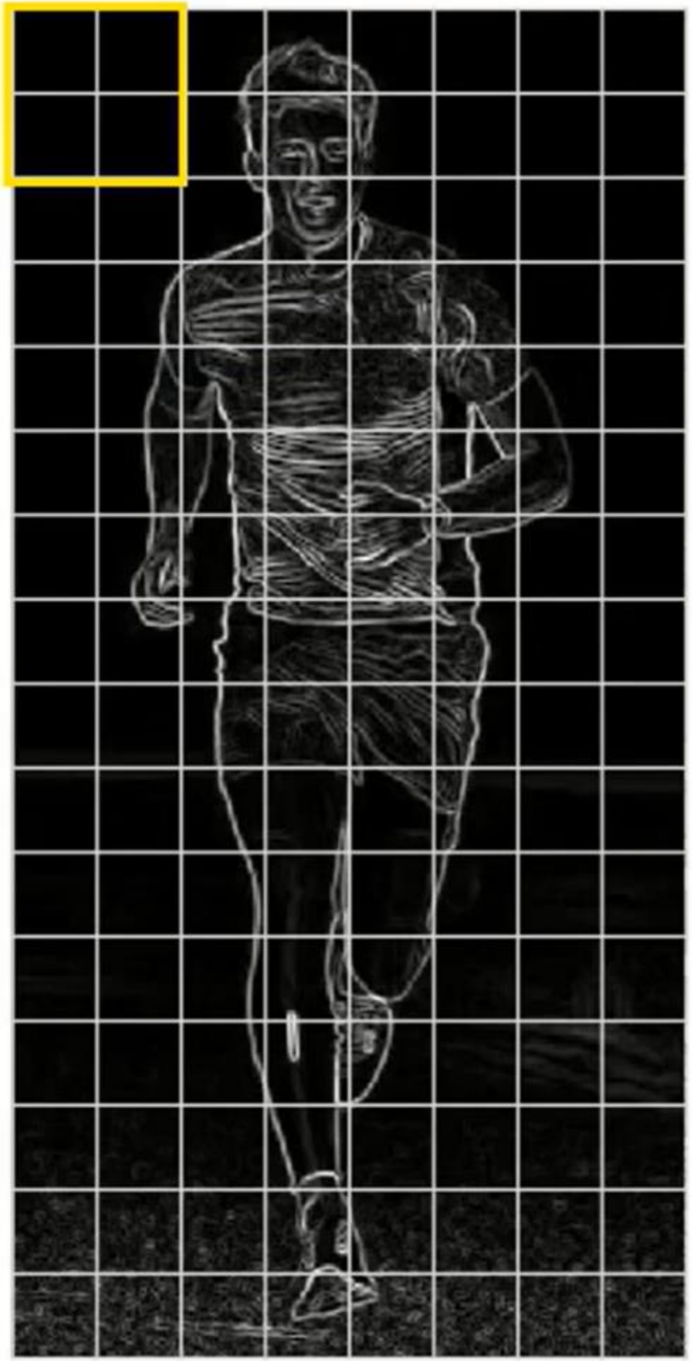
L1-sqrt: $$f = \sqrt{\frac{v}{(\|v\|_1 + e)}}$$

In their experiments, Dalal and Triggs found the L2-hys, L2-norm, and L1-sqrt schemes provide similar performance, while the L1-norm provides slightly less reliable performance.
However, all four methods showed very significant improvement over the non-normalized data.

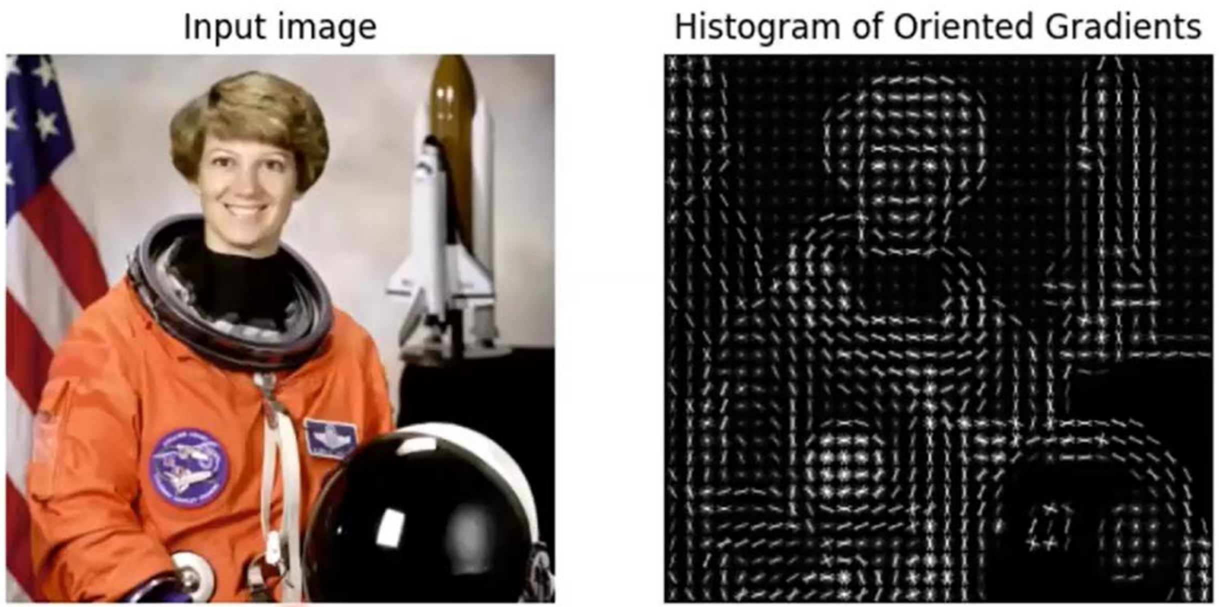
Getting HOG Feature Vector

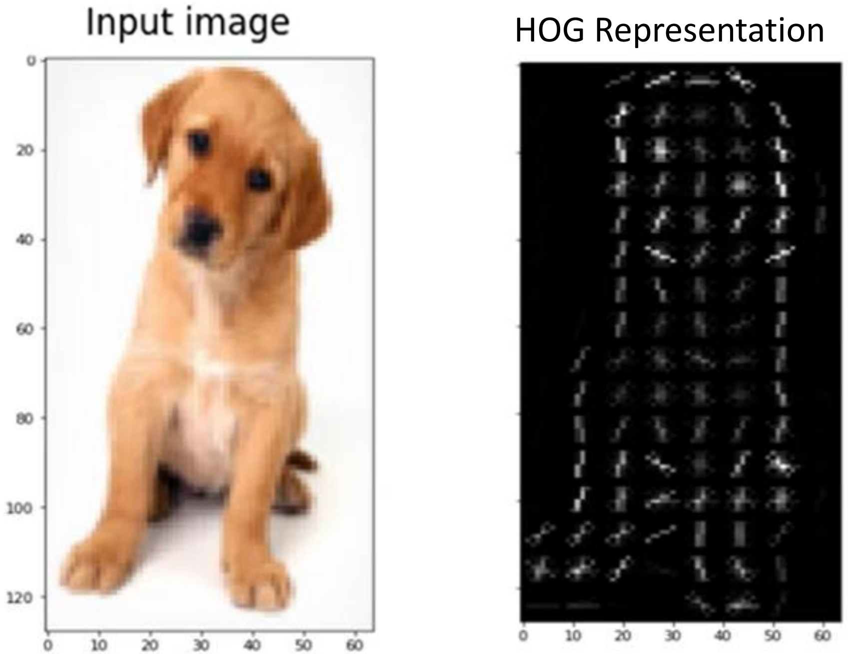
- From each block, a 36-point feature vector is collected.
- In horizontal direction, there are total 7 blocks.
- And in vertical direction, there will be total 15 blocks.
- Therefore, total length of HOG feature is,

$$7 \times 15 \times 36 = 3780$$



HOG Feature Examples





HOG Design Options

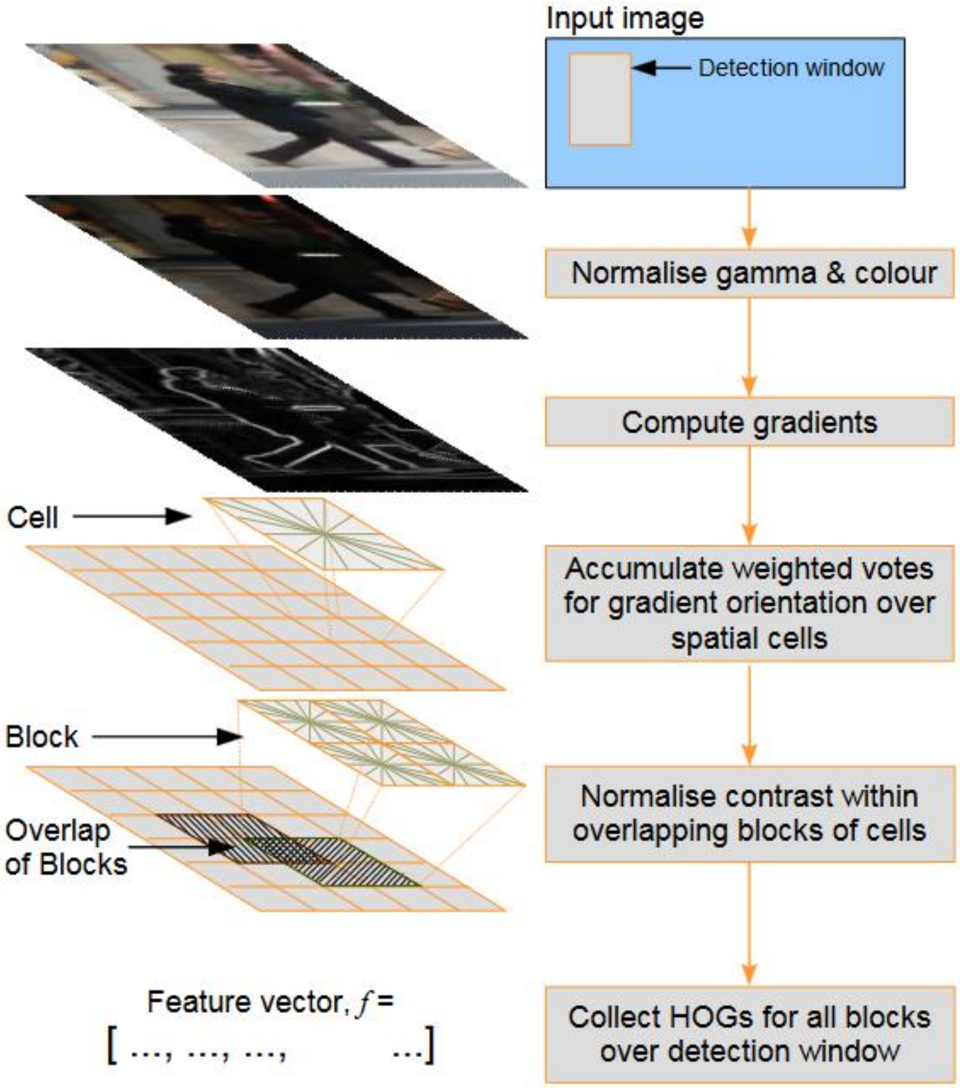
- Parameters and design options:
    - Angles range from 0 to 180 or from 0 to 360 degrees?
        - In the Dalal & Triggs paper, a range of 0 to 180 degrees is used,
        - and HOGs are used for detection of pedestrians.
    - Number of orientation bins.
        - Usually 9 bins, each bin covering 20 degrees.
    - Cell size.
        - Cells of size 8x8 pixels are often used.
    - Block size.
        - Blocks of size 2x2 cells (16x16 pixels) are often used.
- Usually a HOG feature has 36 dimensions.
    - 4 cells * 9 orientation bins.

HoG Descriptor processing chain

Brief *HoG* descriptor processing chain

- By cropping, we obtain an image patch.
- Compute the gradients for pixels in the patch.
- Divide the image patch by cells where each cell is composed of 8×8 pixels (local regions in image patch).
- Make histograms using the computed gradients (magnitude, orientation) for each cell.
- L2 normalization with neighboring histograms
- Concatenate all histograms to a vector, and it is the vector representation produced by *HoG* for the given image patch

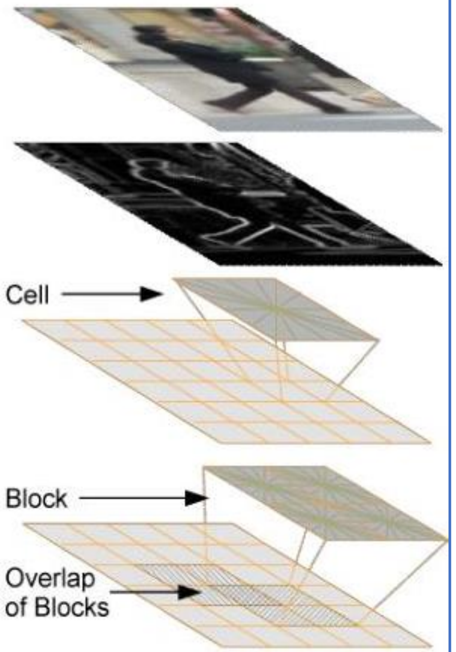
HOG Descriptor



HOG Descriptor for a 64x128 pixels image region

Descriptor

1. Compute gradients on an image region of 64x128 pixels
2. Compute histograms on 'cells' of typically 8x8 pixels (i.e. 8x16 cells)
3. Normalize histograms within overlapping blocks of cells (typically 2x2 cells, i.e. 7x15 blocks)
4. Concatenate histograms



HOG vs SIFT Comparison

#	Topic	HOG	SIFT
1	Feature Descriptors	The unit size of the HOG feature is small, so a certain spatial resolution can be preserved, and the normalization operation makes the feature insensitive to local contrast changes.	SIFT features for interest points are usually associated with a definite orientation and scale. SIFT feature of the square area in an image is calculated after the corresponding direction and scale transformation are performed.
2	Scale and Illumination Changes	HOG descriptors are sensitive to scale and illumination changes, which can affect the accuracy of object recognition	SIFT features are invariant to scale and illumination changes.
3	Feature Extraction	HOG is good for feature extraction of rigid objects like people	SIFT is better for feature extraction of objects in complex environments
4	Natural scene classification	Less accurate compared to SIFT	Accuracy of SIFT is higher than that of HOG
5	Performance	HOG is computationally efficient compared to other feature extraction methods like SIFT and SURF, making it suitable for real-time applications.	Less Efficient
6	Feature representation Compared to Deep Learning	Deep learning is a self-learning feature representation method, which has a better representation than SIFT/HOG.	The local features of the last layer of the neural network were similar to SIFT, but the representative ability was much stronger. CNN can do everything that SIFT can do.

Histograms of Oriented Gradients Algorithm for Object Detection

❖ Histograms of Orientated Gradients (HOG) algorithm is implemented in a series of steps:

1. Given the image of particular object, set a detection window (region of interest) that covers the entire object in the image.
2. Calculate the magnitude and direction of the gradient for each individual pixel in the detection window.
3. Divide the detection window into connected *cells* of pixels, with all cells being of the same size (see Fig. 3). The size of the cells is a free parameter and it is usually chosen so as to match the scale of the features that want to be detected. For example, in a 64 x 128 pixel detection window, square cells 6 to 8 pixels wide are suitable for detecting human limbs.
4. Create a Histogram for each cell, by first grouping the gradient directions of all pixels in each cell into a particular number of orientation (angular) bins; and then adding up the gradient magnitudes of the gradients in each angular bin (see fig. in next slide). The number of bins in the histogram is a free parameter and it is usually set to 9 angular bins.
5. Group adjacent cells into *blocks* (see fig. in next slide). The number of cells in each block is a free parameter and all blocks must be of the same size. The distance between each block (known as the stride) is a free parameter but it is usually set to half the block size, in which case you will get overlapping blocks. The HOG algorithm has been shown empirically to work better with overlapping blocks.
6. Use the cells contained within each block to normalize the cell histograms in that block (see fig. in next slide). If you have overlapping blocks this means that most cells will be normalized with respect to different blocks. Therefore, the same cell may have several different normalizations.
7. Collect all the normalized histograms from all the blocks into a single feature vector called the HOG descriptor.
8. Use the resulting HOG descriptors from many images of the same type of object to train a machine learning algorithm, such as an SVM, to detect those type of objects in images. For example, you could use the HOG descriptors from many images of pedestrians to train an SVM to detect pedestrians in images. The training is done with both positive a negative examples of the object you want detect in the image.
9. Once the SVM has been trained, a sliding window approach is used to try to detect and locate objects in images. Detecting an object in the image entails finding the part of the image that looks similar to the HOG pattern learned by the SVM.

References

- HOG Intuition | Simple Explanation | Feature Descriptor & Engineering
- Histogram of Gradients: UCF Center for Research in Computer Vision
- HOG Features (Theory and Implementation using MATLAB and Python)
- https://courses.cs.duke.edu/fall15/cps274/notes/hog.pdf (good technical overview of HOG)
- http://vision.stanford.edu/teaching/cs231b_spring1213/papers/CVPR05_DalalTriggs.pdf (Original Paper)
- https://medium.com/jun94-devpblog/cv-9-object-detection-with-sliding-window-and-feature-extraction-hog-cf1820c86b46
- https://theses.hal.science/tel-00390303/file/NavneetDalalThesis.pdf
- https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients
- https://medium.com/@danyang95luck/comparison-of-hog-histogram-of-oriented-gradients-and-sift-scale-invariant-feature-transform-e2b17f61c9a3
- https://builtin.com/articles/histogram-of-oriented-gradients (Good example)