# Computer Vision

Feature Extraction (features, DoG, LoG, HOG)

# Contents

- Difference of Gaussians
- Histogram of Oriented Gradients

# Image features

- Features are
  - Edges
  - Color
  - Texture
  - Object boundary
  - Object shape etc
- Good features must be
  - Unique and distinctive
  - non-redundant
  - Robust
  - Global and not specific

# Gradient-based Features

- Some of the techniques are
  - DoG
  - LoG
  - HoG
  - SIFT
  - SURF…
- Advantages:
  - Invariant to small shifts and rotations of images
  - Provides localized histograms

    Which offers accurate spatial information compared to a single global histogram
  - Includes contrast normalization: reduce the impact of variable illumination of images

# Difference of Gaussian (DoG)

- Is a feature enhancement algorithm
- Use Gaussian filters to blur the image
- Find difference of Gaussian blurred version of an original image and less blurred version

$$DoG = I * G_{\sigma_1} - I * G_{\sigma_2}$$

- Where I is grey image, $G_{\sigma1}$ and $G_{\sigma2}$ are Gaussian filters with sigma, σ1 and σ2 respectively
- Gaussian kernel suppresses high-frequency spatial information
- Subtraction preserves spatial information that has frequency which is not common to filtered images
- Thus, DoG is a spatial band-pass filter

# Gaussian Filter

- Gaussian filter with mean 0 and standard deviation, σ is

3×3 filter with σ = 1

(1/16)

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

5×5 filter with σ = 1

(1/330)

| | | | | |
|---|---|---|---|---|
| 1 | 4 | 7 | 4 | 1 |
| 4 | 20 | 33 | 20 | 4 |
| 7 | 33 | 54 | 33 | 7 |
| 4 | 20 | 33 | 20 | 4 |
| 1 | 4 | 7 | 4 | 1 |

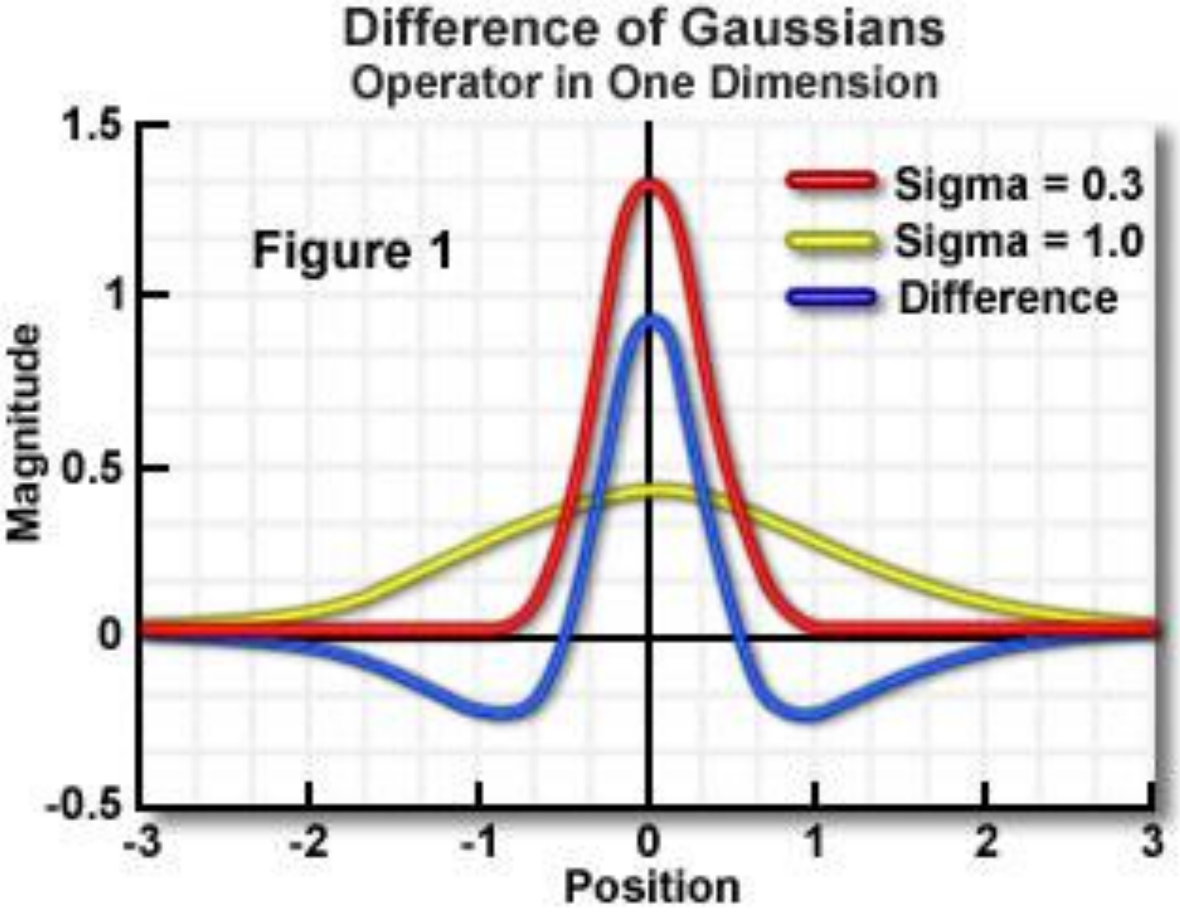
5×5 filter with σ = 2

(1/34)

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 2 | 2 | 2 | 1 |
| 1 | 2 | 2 | 2 | 1 |
| 1 | 2 | 2 | 2 | 1 |
| 1 | 1 | 1 | 1 | 1 |

- Gaussian filter is averaging filter which blurs the image
- Blurring increases with the increase in σ

# Difference of Gaussian (DoG)



# Determine DoG

Image

| | | | | | | |
|---|---|---|---|---|---|---|
| 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Difference of Gaussian for Edge Detection

Filtered Image for σ = 1

| | | | | | | |
|---|---|---|---|---|---|---|
| 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| 100 | 100 | 70 | 70 | 70 | 100 | 100 |
| 0 | 0 | 30 | 30 | 30 | 0 | 0 |
| 0 | 0 | 10 | 10 | 10 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Filtered Image for σ = 2

| | | | | | | |
|---|---|---|---|---|---|---|
| 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| 100 | 100 | 60 | 60 | 60 | 100 | 100 |
| 0 | 0 | 40 | 40 | 40 | 0 | 0 |
| 0 | 0 | 10 | 10 | 10 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- Subtract filtered image with high sigma from filtered image with low sigma
- Called difference of Gaussian

Difference of Gaussian (DoG)

| | | | | | | |
|---|---|---|---|---|---|---|
| 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| 100 | 100 | 10 | 10 | 10 | 100 | 100 |
| 0 | 0 | -10 | -10 | -10 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Difference of Gaussian (DoG)

Original Image
(Sigma 0)

# Difference of Gaussian (DoG)


- Some details appear in all three images
- Large details like eye are blurred
- Tiny feathers around the beak are not clear in rightmost image
- Some details are only visible for a specific sigma
- The sigma is a scale factor that gives clue of which features are visible at a particular scale

# Difference of Gaussian (DoG)

Sigma 0.7

Sigma 1

# Difference of Gaussian (DoG)

Sigma 0.7 Sigma 1 Difference of Sigma 0.7 and 1 (with Thresholding)

- Features that are visible in both images appear black
- Features that are only visible for one sigma, appear white
- Contains fine details in the feathers
- A pair of specific sigma controls the required details

# Difference of Gaussian (DoG)

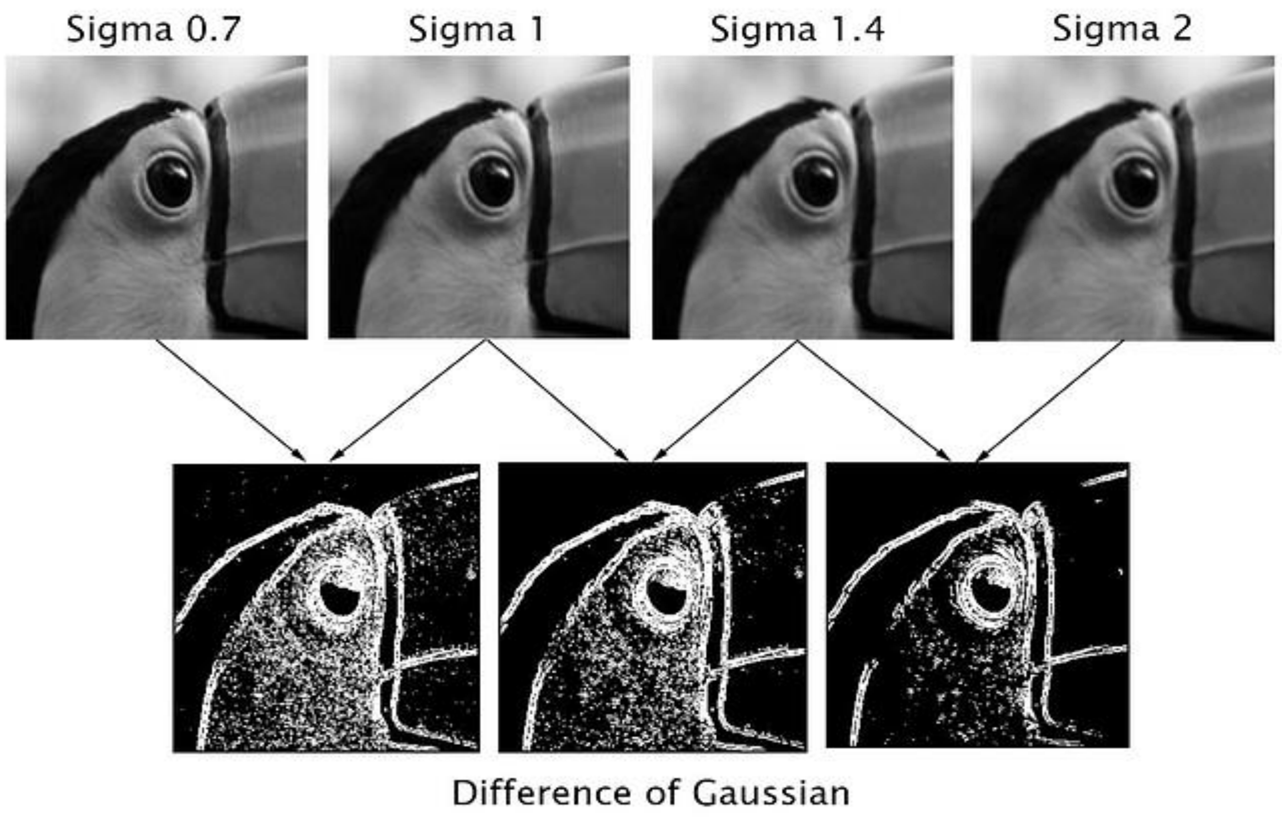
Sigma 2

Sigma 2.8

Difference of Sigma 2 and 2.8 (with Thresholding)

- Feather detail is not prominent
- Larger details (such as the eye) is visible
- For little change in an area, the result of the subtraction is closer to zero
- In high contrast areas (edges, blobs) the strength of the blur has a bigger impact

# Difference of Gaussian (DoG)




- Different scales show different details
- Repeatedly subtract adjacent scales to produce several DoGs

# Difference of Gaussian (DoG)

- Removes high frequency detail that often includes random noise
- Therefore, suitable for processing images with noise
- Drawback: Reduction in overall image contrast
- Used for blob detection and in SIFT descriptors

# Laplacian of Gaussian (LoG)

- Laplacian operator is also used to enhance edges of the images

$$\nabla^2 f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}$$

- It is sensitive to noise
- Before applying Laplacian operator, image is blurred using Gaussian filter
- Laplace of Gaussian is

$$\Delta(I * G) = I * \Delta G$$

I is image, G is Gaussian filter and * is convolution operator

- Laplacian of the image smoothed by a Gaussian kernel is identical to the image convolved with the Laplacian of the Gaussian kernel

# Laplacian of Gaussian for edge Detection

Apply Laplacian filter on Gaussian filtered image to detect edges

Image

| | | | | |
|---|---|---|---|---|
| 100 | 100 | 100 | 100 | 100 |
| 100 | 100 | 100 | 100 | 100 |
| 10 | 10 | 10 | 10 | 10 |
| 10 | 10 | 20 | 10 | 10 |
| 10 | 10 | 10 | 10 | 10 |

Gaussian filter

(1/16)

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

Laplacian filter

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

Gaussian filtered image

| | | | | |
|---|---|---|---|---|
| 10 | 10 | 100 | 10 | 10 |
| 10 | 77.5 | | | 10 |
| 100 | 33.5 | | | 100 |
| 10 | 11.25 | | | 10 |
| 10 | 10 | 100 | 10 | 10 |

Laplacian of Gaussian filtered image

| | | | | |
|---|---|---|---|---|
| 10 | 10 | 100 | 10 | 10 |
| 10 | | | | 10 |
| 100 | | | | 100 |
| 10 | | | | 10 |
| 10 | 10 | 100 | 10 | 10 |

# LoG for Edge Detection

- For LoG filtered image-
  - Set a threshold for zero crossings
  - Edge points have values for which zero crossings exceeds threshold
  - Strong zero crossings have a large positive maximum and the negative minimum on either size of the zero
  - Weak zero crossings are most likely noise
  - Therefore they are ignored
  - Use optimum value for threshold to detect edges

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 0 |
| 1 | 2 | 4 | 5 | 5 | 5 | 4 | 2 | 1 |
| 1 | 4 | 5 | 3 | 0 | 3 | 5 | 4 | 1 |
| 2 | 5 | 3 | -12 | -24 | -12 | 3 | 5 | 2 |
| 2 | 5 | 0 | -24 | -40 | -24 | 0 | 5 | 2 |
| 2 | 5 | 3 | -12 | -24 | -12 | 3 | 5 | 2 |
| 1 | 4 | 5 | 3 | 0 | 3 | 5 | 4 | 1 |
| 1 | 2 | 4 | 5 | 5 | 5 | 4 | 2 | 1 |
| 0 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 0 |

LoG filter for $\sigma = 1.4$

# Example of a LoG approximation

Highlight edges of LoG of the image, for threshold = 8

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| -25 | -25 | -25 | -25 | -25 | -25 | -25 | -25 | -25 |
| -25 | -25 | -25 | -25 | -25 | -25 | -25 | -25 | -25 |
| -25 | -25 | -25 | -25 | -25 | -25 | -25 | -25 | -25 |
| -10 | -10 | -10 | -10 | -10 | -10 | -10 | -10 | -10 |
| -10 | -10 | -10 | -10 | 25 | 25 | 25 | 25 | 25 |
| -10 | -10 | -10 | -10 | 25 | 60 | 60 | 60 | 60 |
| -10 | -10 | -10 | -10 | 25 | 60 | 60 | 60 | 60 |

LoG of an image

# LoG for Edge Detection

Let The resulting **LoG filtered image** is:

$$\text{LoG result} = \begin{bmatrix} 2 & 4 & 3 & 4 & 2 \\ 4 & -4 & -6 & -4 & 4 \\ 3 & -6 & -12 & -6 & 3 \\ 4 & -4 & -6 & -4 & 4 \\ 2 & 4 & 3 & 4 & 2 \end{bmatrix}$$

A **zero crossing** occurs at a pixel if the product of its value and the value of any of its **immediate neighbors** is **negative**. This means the sign of the pixel changes between itself and its neighbor, indicating a potential edge.

**Detect Zero Crossings**
We will check each pixel and its **4 neighbors** (up, down, left, right) for sign changes. Let's go through the process for a few key pixels:

# LoG for Edge Detection

**Example 1: Pixel at (1,2) = -6**

Neighbors of pixel **(1,2)**:

- Up: **3** at (0,2)
- Down: **-12** at (2,2)
- Left: **-4** at (1,1)
- Right: **-4** at (1,3)

Check the product with each neighbor:

- $(-6) \times 3 = -18 \quad \text{(Sign change: Zero crossing)}$
- $(-6) \times (-12) = 72 \quad \text{(No sign change)}$
- $(-6) \times (-4) = 24 \quad \text{(No sign change)}$
- $(-6) \times (-4) = 24 \quad \text{(No sign change)}$

Since there is a **sign change with the neighbor above** (3), a **zero crossing** is detected at **(1,2)**.

**Example 2: Pixel at (2,2) = -12**

Neighbors of pixel **(2,2)**:

- Up: **-6** at (1,2)
- Down: **-6** at (3,2)
- Left: **-6** at (2,1)
- Right: **-6** at (2,3)

Check the product with each neighbor:

- $(-12) \times (-6) = 72 \quad \text{(No sign change)}$
- $(-12) \times (-6) = 72 \quad \text{(No sign change)}$
- $(-12) \times (-6) = 72 \quad \text{(No sign change)}$
- $(-12) \times (-6) = 72 \quad \text{(No sign change)}$

Since there are **no sign changes**, there is **no zero crossing** at **(2,2)**.

**Example 3: Pixel at (1,1) = -4**

Neighbors of pixel **(1,1)**:

- Up: **4** at (0,1)
- Down: **-6** at (2,1)
- Left: **4** at (1,0)
- Right: **-6** at (1,2)

Check the product with each neighbor:

- $(-4) \times 4 = -16 \quad \text{(Sign change: Zero crossing)}$
- $(-4) \times (-6) = 24 \quad \text{(No sign change)}$
- $(-4) \times 4 = -16 \quad \text{(Sign change: Zero crossing)}$
- $(-4) \times (-6) = 24 \quad \text{(No sign change)}$

Since there are **sign changes with the neighbors up and left**, a **zero crossing** is detected at **(1,1)**.

By applying the same process to every pixel in the image, we get the following **zero crossing map**:

$$\text{Zero Crossing Map} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Here, **1** indicates the presence of a **zero crossing** (potential edge), and **0** indicates no zero crossing.

# LoG for Edge Detection

**Apply Threshold**

To remove weak zero crossings (likely caused by noise), we can apply a **threshold**. A strong zero crossing has a large positive maximum and a large negative minimum around the zero crossing point. We define a threshold of **8**:

1.For each zero crossing, find the maximum positive and negative values among its neighbors.

2.If the absolute difference between the maximum positive and the minimum negative exceeds **8**, we retain the zero crossing; otherwise, we discard it.

Example: Thresholding at (1,2)

- The neighbors of pixel (1,2) are: 3, -12, -4, -4.
- Maximum positive = 3, minimum negative = -12.
- Absolute difference = $|3-(-12)|=15$, which exceeds the threshold of 8, so we retain this zero crossing.

**After applying the threshold to all zero crossings, the final edge map retains only the significant zero crossings:**

$$\text{Final Edge Map} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

**Reason for Not Calculating Zero Crossings for Boundary Pixels**

**1.Insufficient Neighbors**:

Zero crossings are detected by comparing a pixel with its **immediate neighbors** (up, down, left, right). Boundary pixels do not have a complete set of neighbors:

1. For example, the pixel at **(0,1)** (in the first row) only has three neighbors (left, right, and below) instead of four.

**2.Edge Artifacts**:

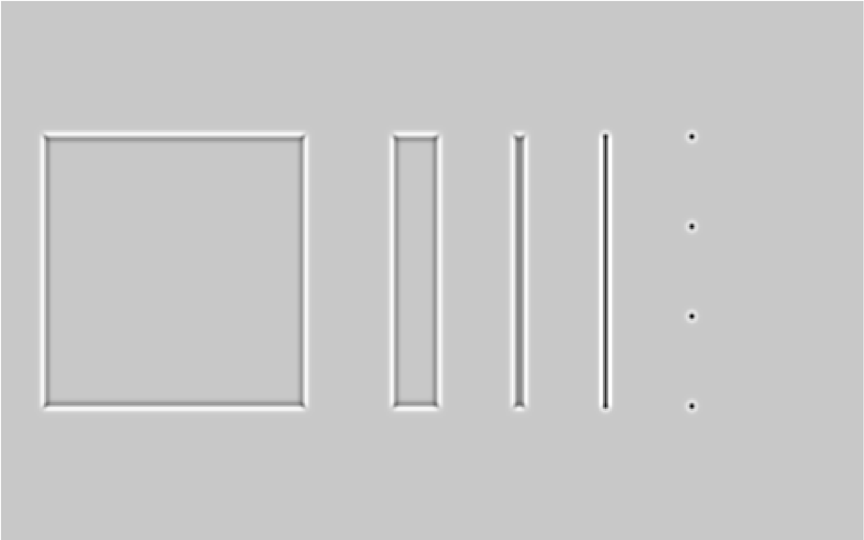
Boundary pixels often have incomplete information, which may lead to **false zero crossings** due to abrupt changes near the image edges or padding artifacts.

# Laplacian of Gaussian (LoG)

- Can be used to construct an edge detector

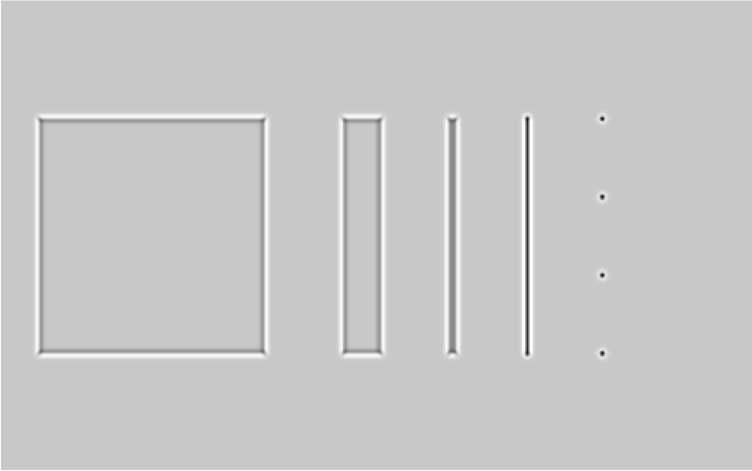
Original image

LoG of image



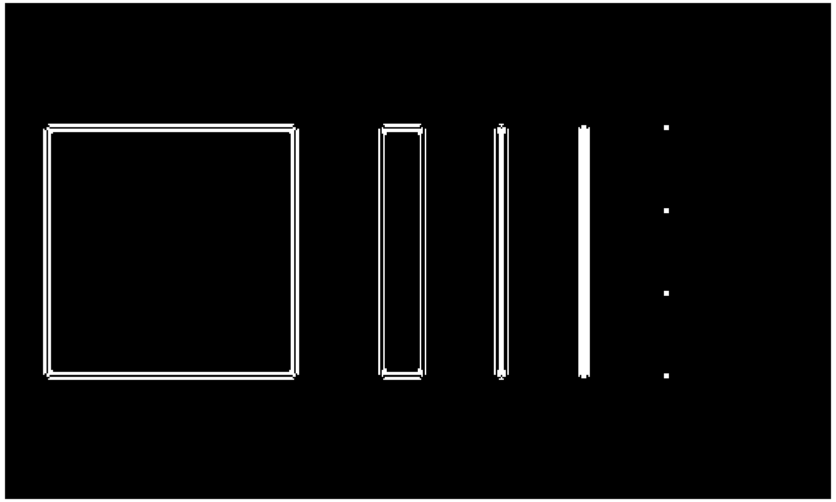
# Laplacian  of Gaussian (LoG)

LoG of image



Threshold (t1) LoG of image

Threshold (t2, t2<t1) LoG of image



- Strong (negative) response along the thin line and on the small dots
- Medium responses around the edges of the wider objects

# Laplacian of Gaussian (LoG)

- DoG closely resembles Laplacian of Gaussian (LoG)
- DoG can be used to obtain an approximation of LoG for $\sigma 2/\sigma 1 \approx 1.6$
- Computation of DoG is faster than LoG

# Histogram of Oriented Gradients

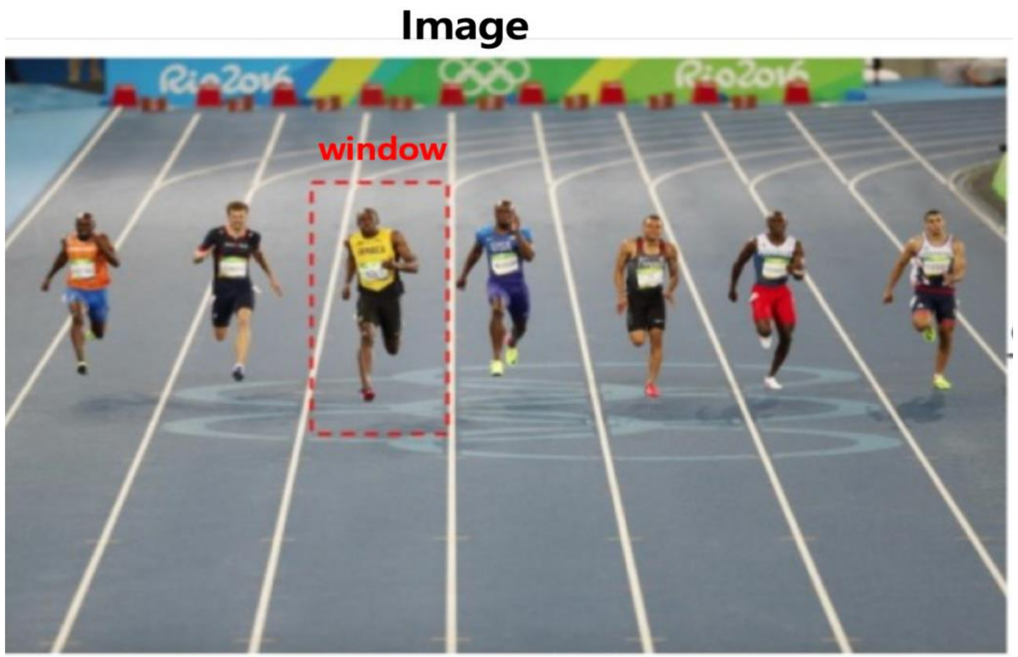
- Popular feature descriptor technique
- Widely used for object detection
- Describes shape and appearance of object
- Better than conventional edge detectors
  - Because edge detectors identify if the pixel is an edge or not
  - They do not consider gradient angle at each pixel
  - HOG Provides edge magnitude as well direction

# Histograms of Oriented Gradients (HoG)

Commonly used to extract strong features for object detection

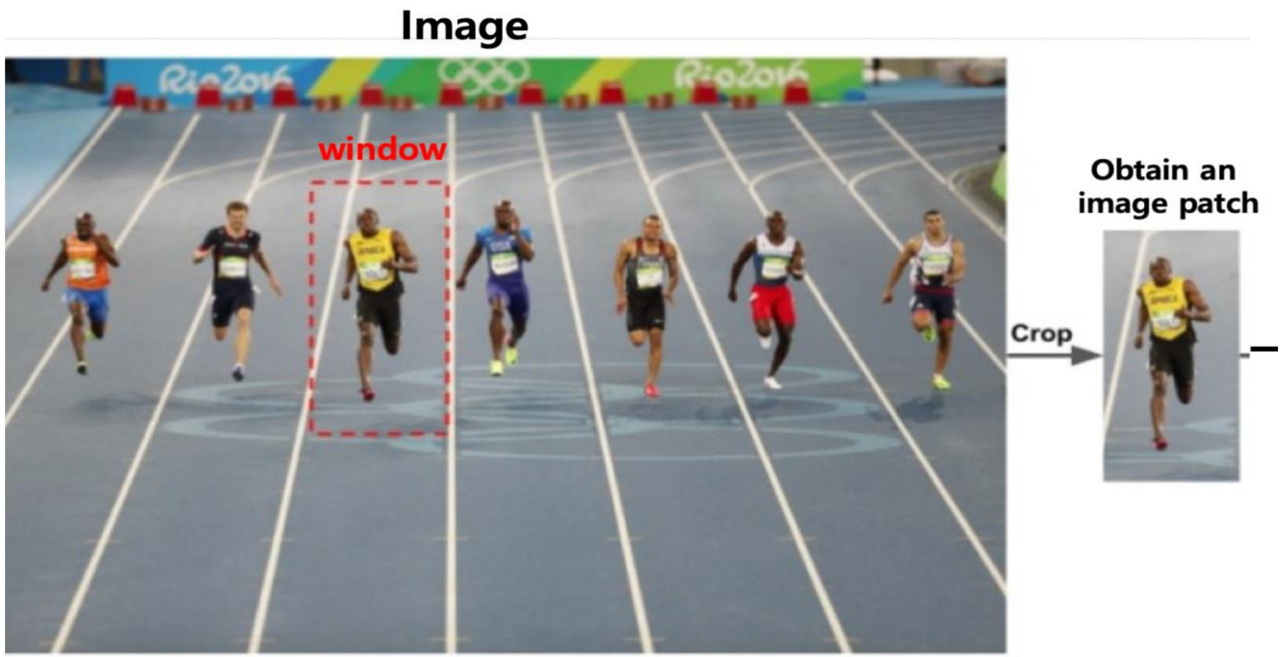
# Histograms of Oriented Gradients (HoG)

Commonly used to extract strong features for object detection



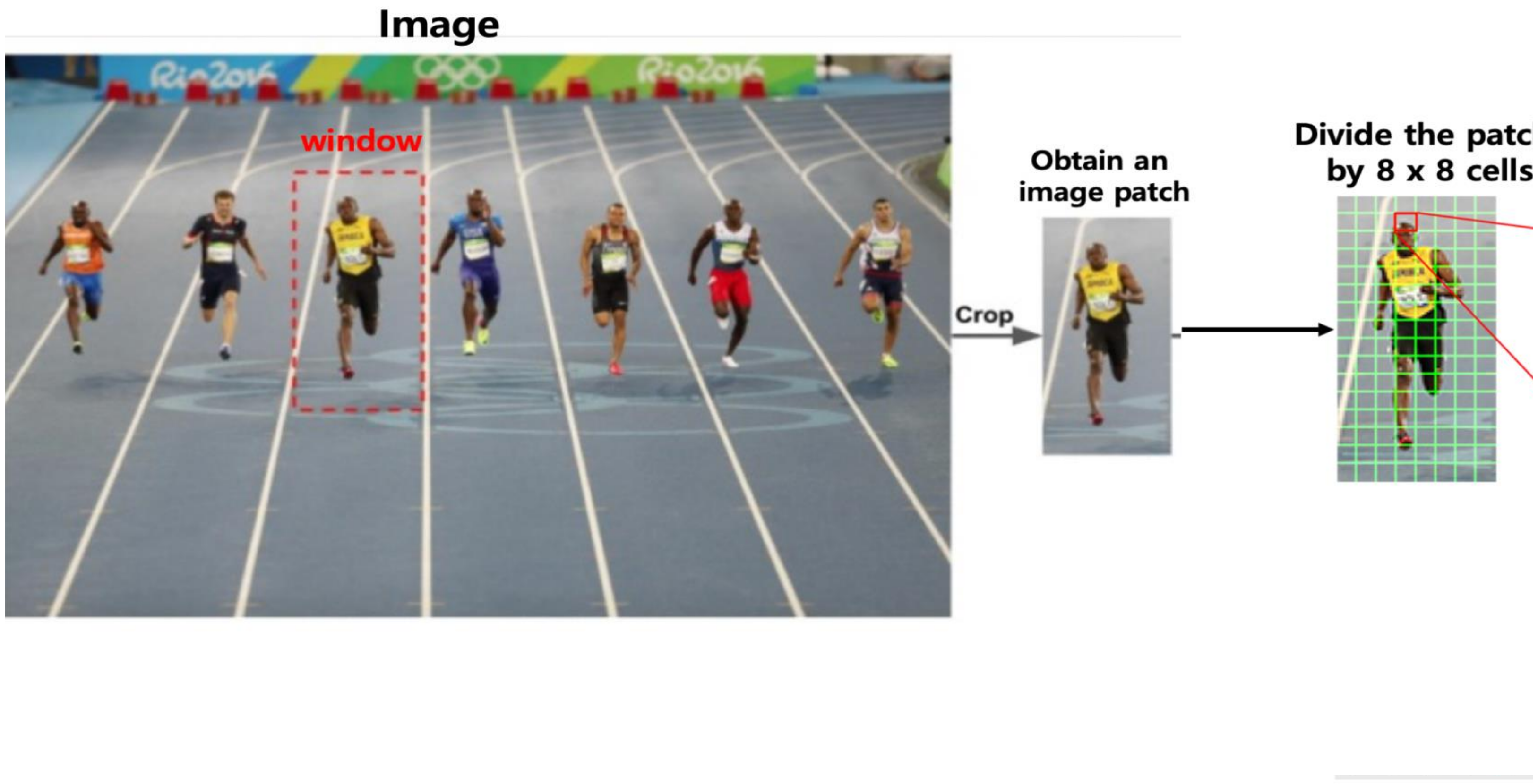
# Histograms of Oriented Gradients (HoG)

Commonly used to extract strong features for object detection



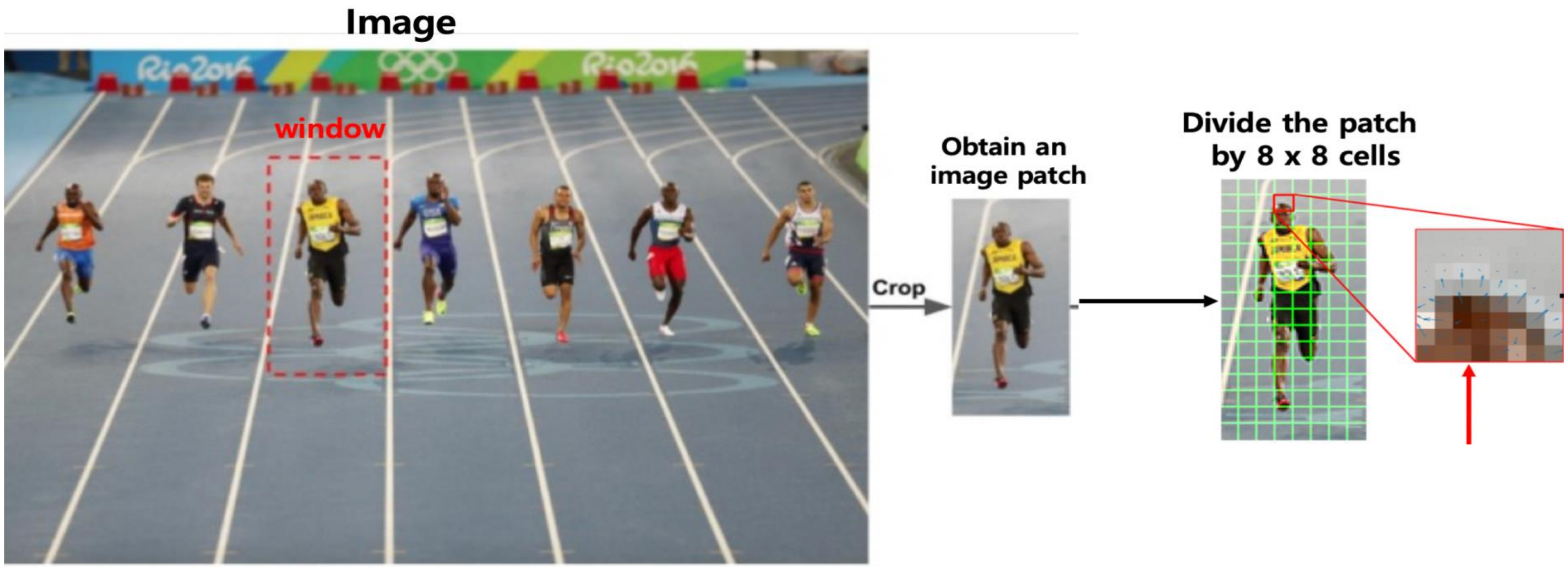
# Histograms of Oriented Gradients (HoG)

Commonly used to extract strong features for object detection



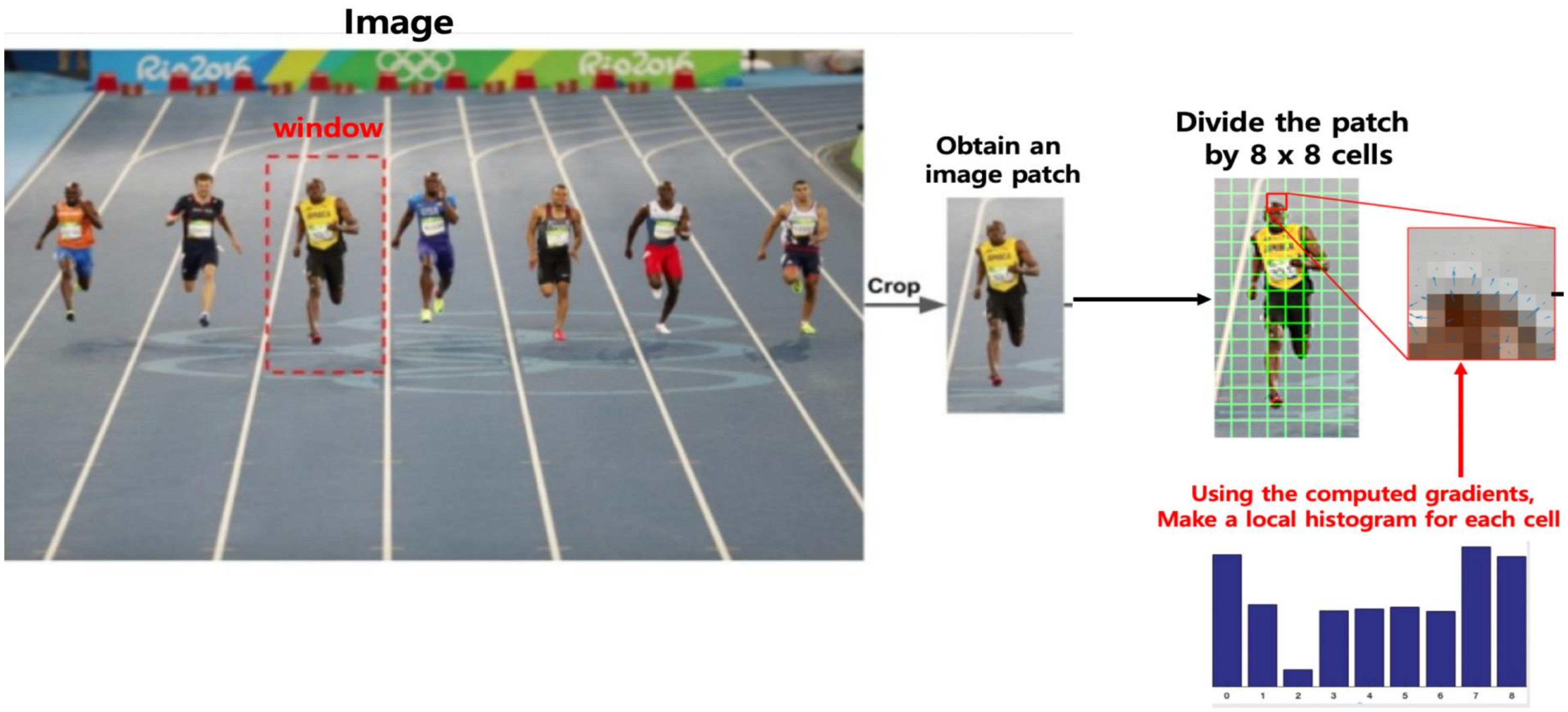
# Histograms of Oriented Gradients (HoG)

Commonly used to extract strong features for object detection



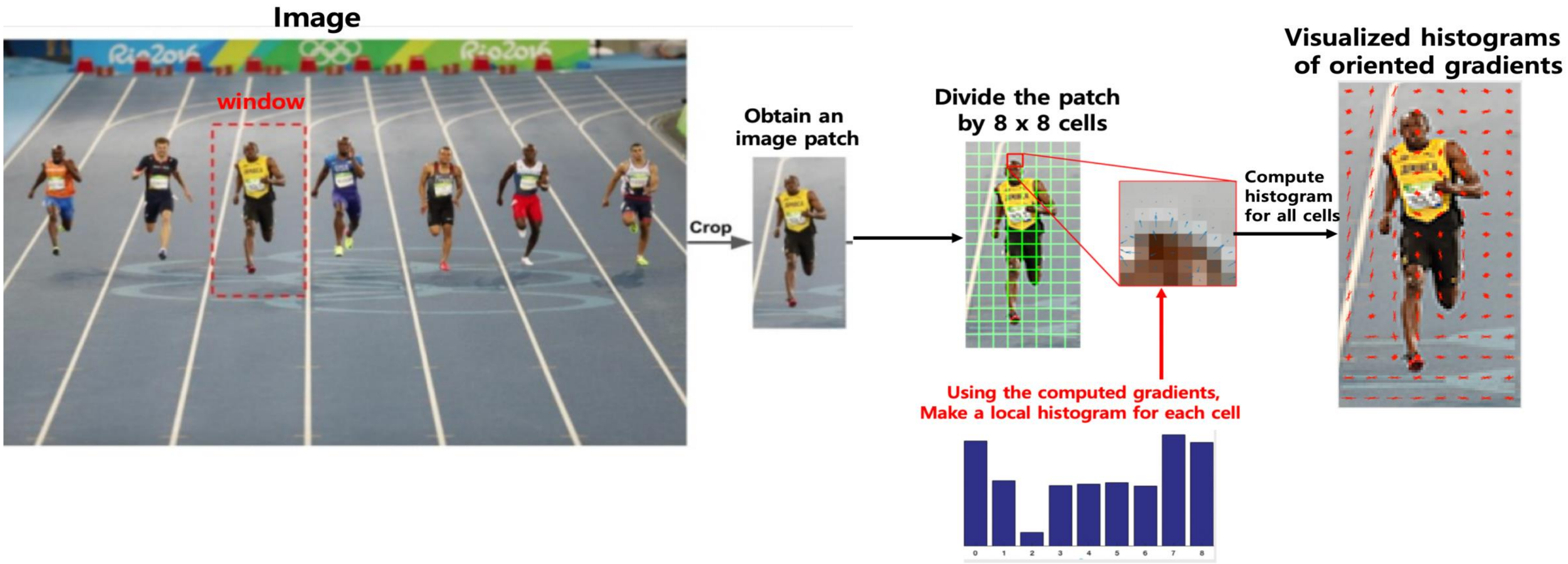
# Histograms of Oriented Gradients (HoG)

Commonly used to extract strong features for object detection



# Histograms of Oriented Gradients (HoG)

Commonly used to extract strong features for object detection



# Steps to determine HoG descriptor

1. Crop image to obtain image patch
2. Divide patch by cells of $8 \times 8$ pixels
3. Aim is to compute the gradients for pixels in the patch (localized portion)
4. Make histograms using the computed gradients (magnitude and orientation) for each cell
5. Apply normalization with neighboring histograms
6. Concatenate all histograms to form a vector
7. Vector represents feature of the image patch
8. Use features to compare objects
9. Or use features to train classifier

# Process of Calculating HOG (Step 1)

- Resize image to integer multiple of 8 (nearest to original size)


Original Image : 720 x 475

# Process of Calculating HOG (Step 2)

- To calculate gradients of patch
- Divide the image patch into cells of the same size (Ex: 8x8)
- Size can be 16x16 or larger
- Determine gradient of each pixel

# Process of Calculating HOG (Step 2)

- To calculate gradients of patch
- Divide the image patch into cells of the same size (Ex: 8x8)
- Size can be 16x16 or larger
- Determine gradient of each pixel

Example image

| | | | | |
|---|---|---|---|---|
| 121 | 10 | 78 | 96 | 125 |
| 48 | 152 | 68 | 125 | 111 |
| 145 | 78 | 85 | 89 | 65 |
| 154 | 214 | 56 | 200 | 66 |
| 214 | 87 | 45 | 102 | 45 |

Kernels for gradients

| | | |
|---|---|---|
| -1 | 0 | 1 |

| |
|---|
| -1 |
| 0 |
| 1 |

- Change in X direction, $G_x = 89 – 78 = 11$
- Change in Y direction, $G_y = 56 - 68 = -12$

# Process of Calculating HOG (Step 3)

- Step 3: Calculate Magnitude and Orientation
  - Magnitude of gradient

$$M(x, y) = \sqrt{{G_x}^2 + {G_y}^2} \quad = \sqrt{11^2 + 12^2}$$

  - Angle of gradient (in degree)

$$\emptyset = tan^{-1}\left(\frac{G_y}{G_x}\right) = 36$$

  - Magnitude would be higher when there is a sharp change in intensity (edges)
  - Direction of the vectors indicates the direction of the change of pixel intensity

# Process of Calculating HOG (Step 3)

$G_x$

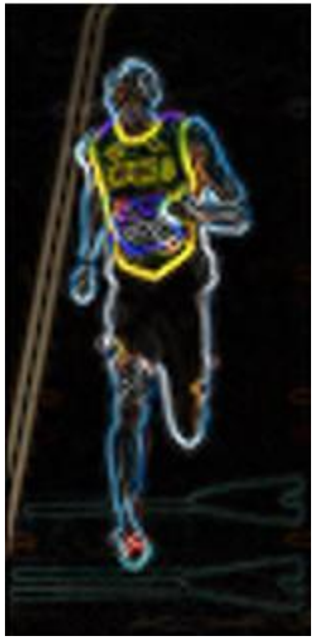
$G_y$

# Process of Calculating HOG (Step 3)

- Magnitude of gradient at a pixel crosses threshold for sharp change in intensity
- It does not cross threshold for smooth region
- Highlights only edges
- For color images, the gradient of pixel for each of the three channels are evaluated
- Magnitude of gradient at a pixel is the maximum of the magnitude of gradients of the three channels
- And the angle is the angle corresponding to the maximum magnitude

$G_x$

$G_y$



M(x,y)

$$M(x,y) = \sqrt{{G_x}^2 + {G_y}^2}$$

# Process of Calculating HOG (Step 4)

- Step 4: Calculate histogram of gradients for 8×8 pixels/cell
- For image size of 64x128,
  - 128 cells are available (8x16 = 128 cells)
  - For each cell, there are 64 pixels
  - 64 pixels have 64 magnitudes and 64 directions (64+64=128 values)
  - Map 128 values into a 9-bin histogram
  - Histogram provides a useful and compact representation of gradients
  - Bins of the histogram correspond to gradients directions 0, 20, 40 … 160 degrees
  - Every pixel votes for either one or two bins in the histogram

# Process of Calculating HOG (Step 4)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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**

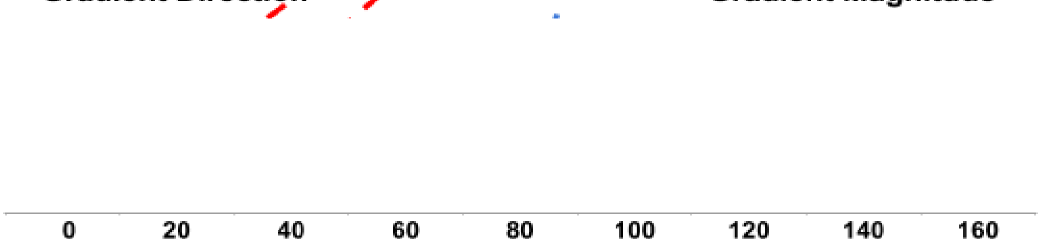
# Process of Calculating HOG (Step 4)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

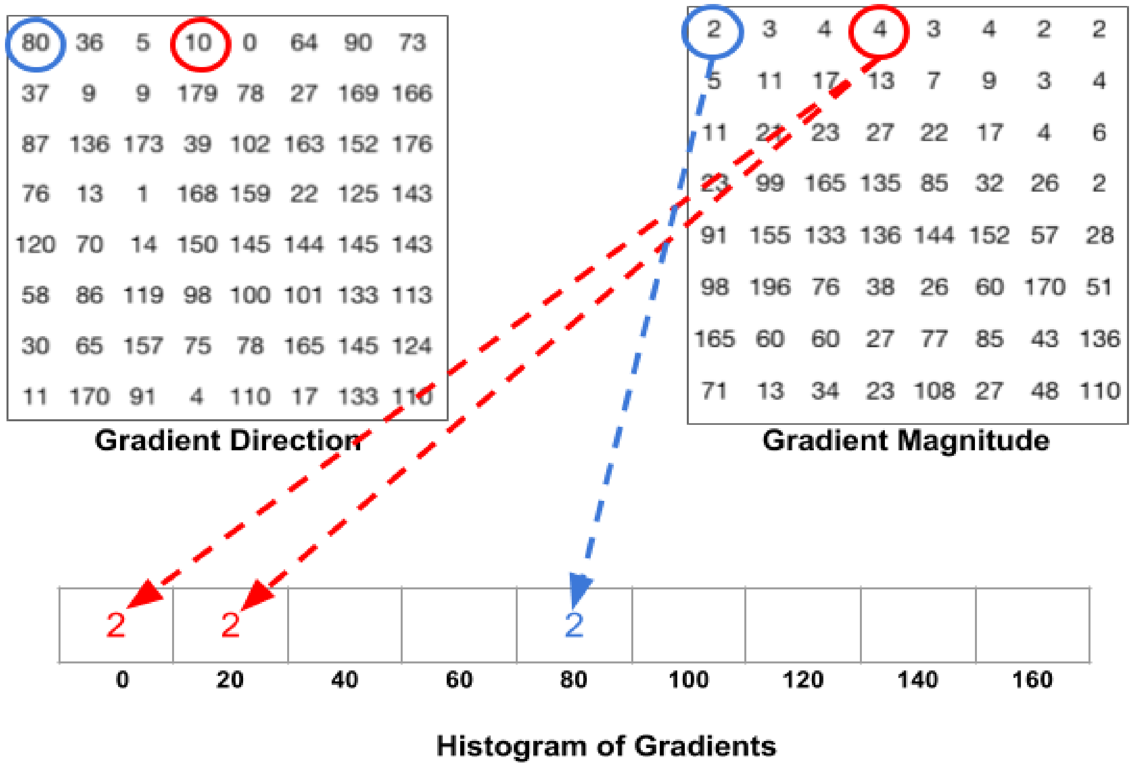
Gradient Magnitude




Histogram of Gradients

- If direction of the gradient at a pixel is exactly 0, 20, 40 … or 160 degrees
  - a vote equal to the magnitude of the gradient is cast by the pixel into the bin
- Else split its vote among two nearest bins based on the distance from the bin
- Ex: Pixel with the magnitude of gradient is 2 and angle is 80 degrees
  - vote for bin at angle 80 is 2
- Ex: Pixel with magnitude 4 and angle 10 has votes for angles 0 and 20
  - (10-0)4/20 =2
  - (20-10)4/20 = 2

# Process of Calculating HOG (Step 4)




- If direction of the gradient at a pixel is exactly 0, 20, 40 … or 160 degrees
  - a vote equal to the magnitude of the gradient is cast by the pixel into the bin
- Else split its vote among two nearest bins based on the distance from the bin
- Ex: Pixel with the magnitude of gradient is 2 and angle is 80 degrees
  - vote for bin at angle 80 is 2
- Ex: Pixel with magnitude 4 and angle 10 has votes for angles 0 and 20
  - (10-0)4/20 =2
  - (20-10)4/20 = 2

# Process of Calculating HOG (step 4)

Magnitude = 13.6
Orientation = 36

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Magnitude | | | | | | | | | |
| Bin | 0 | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 |

# Process of Calculating HOG (step 4)

Magnitude = 13.6

Orientation = 36

(40-36)/20

(36 - 20 )/20

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Bin | 0 | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 |

# Process of Calculating HOG (step 4)

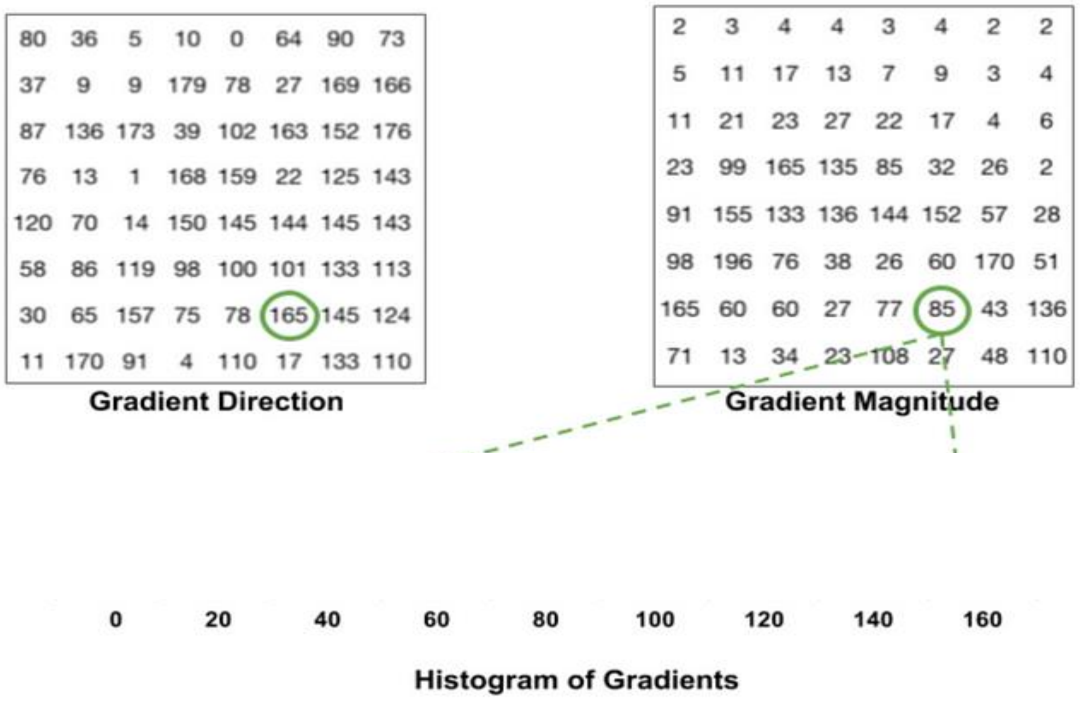
Magnitude = 13.6
Orientation = 36

(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 |

# Process of Calculating HOG (step 4)



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

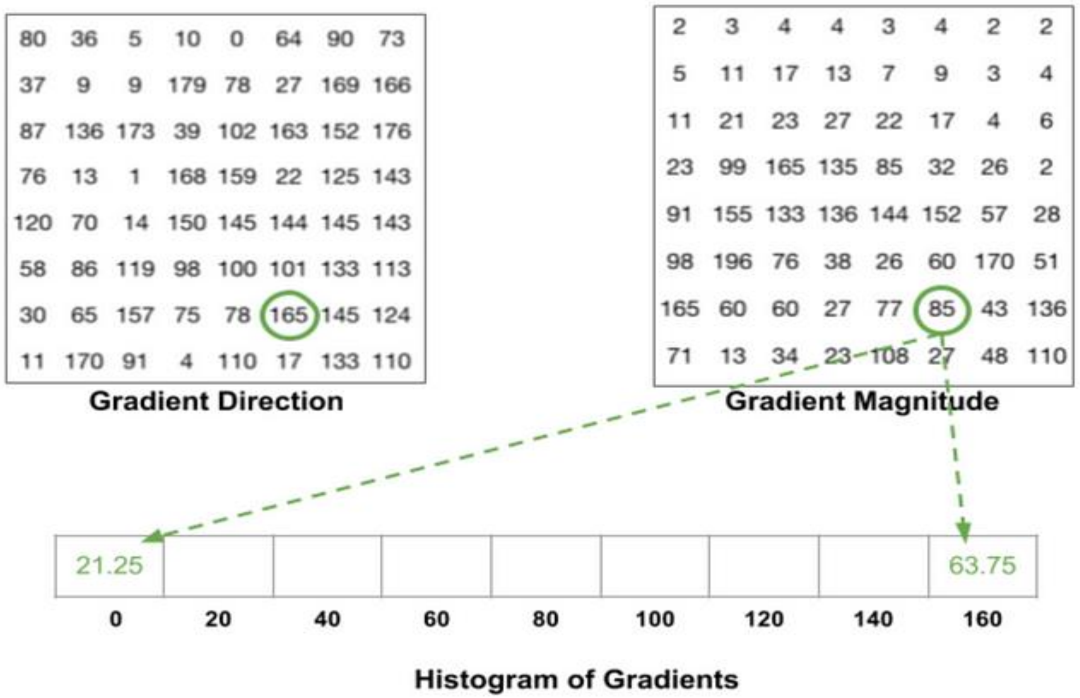
**Gradient Magnitude**



**Histogram of Gradients**

- Lies between 160 and 180, but closer to 160
- (165-160)*85/20= 21.25
- (180-165)*85/20= 63.75
- The bin with 160 degrees gets more votes than that with 0 degrees
- Angle 0 also corresponds to the 180 degrees

# Process of Calculating HOG (step 4)



# Process of Calculating HOG (step 4)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 40 | 60 | 10 | | | | | |
| 50 | 180 | 90 | | | | | |
| 180 | 100 | 40 | | | | | |
| 20 | 100 | | | | | | |
| | 0 | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

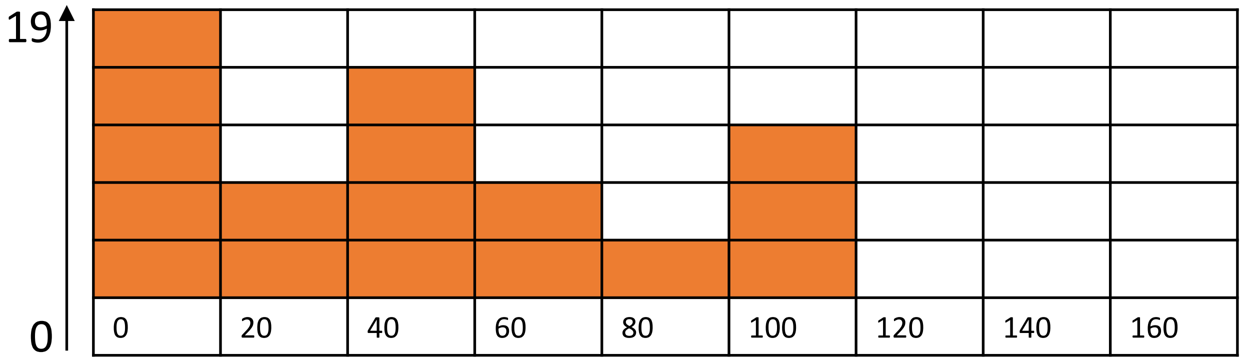
Gradient direction

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 8 | 5 | 4 | | | | | |
| 6 | 9 | 4 | | | | | |
| 7 | 3 | 1 | | | | | |
| 6 | 6 | | | | | | |
| | 1 | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

Gradient magnitude

Orientation Bins

| 0 | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 |
|---|---|---|---|---|---|---|---|---|
| 19 | 8 | 12 | 8 | 2 | 11 | | | |




Histogram for bins

# Process of Calculating HOG (step 5)

- Gradients of an image are sensitive to overall lighting
- If image is made darker by dividing all pixel values by 2, histogram values reduce to half
- HoG descriptor should be independent of lighting variations
- Normalize the histogram to make descriptor independent of lighting variations

# Process of Calculating HOG (step 5)

- Step 5: Determine Feature Vector
  - Normalize histogram over a 16×16 pixels/block
  - Each block contains 4 cells ( 1 histogram/ cell)
  - Each block has 4 histograms (9 values/ histogram)
  - Concatenate 4 histograms to form a 36 x 1 element vector for a block
  - Normalize the vector

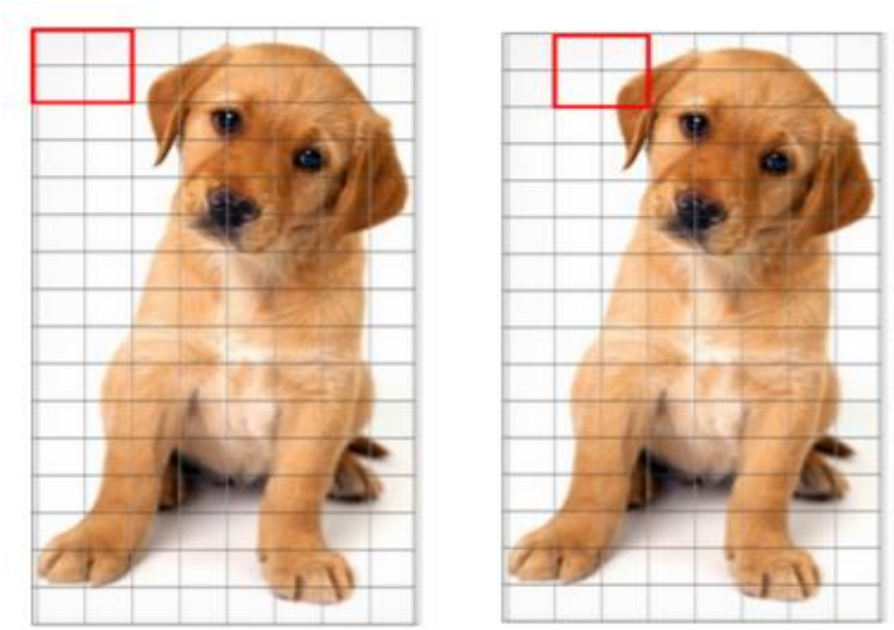
$$a_{n1} = \frac{a_1}{\sqrt{{a_1}^2 + {a_2}^2 + \cdots + {a_{36}}^2}}$$

Where values of *a* represent elements of histogram

- Normalized vector, $a_n$ has 36 element /block

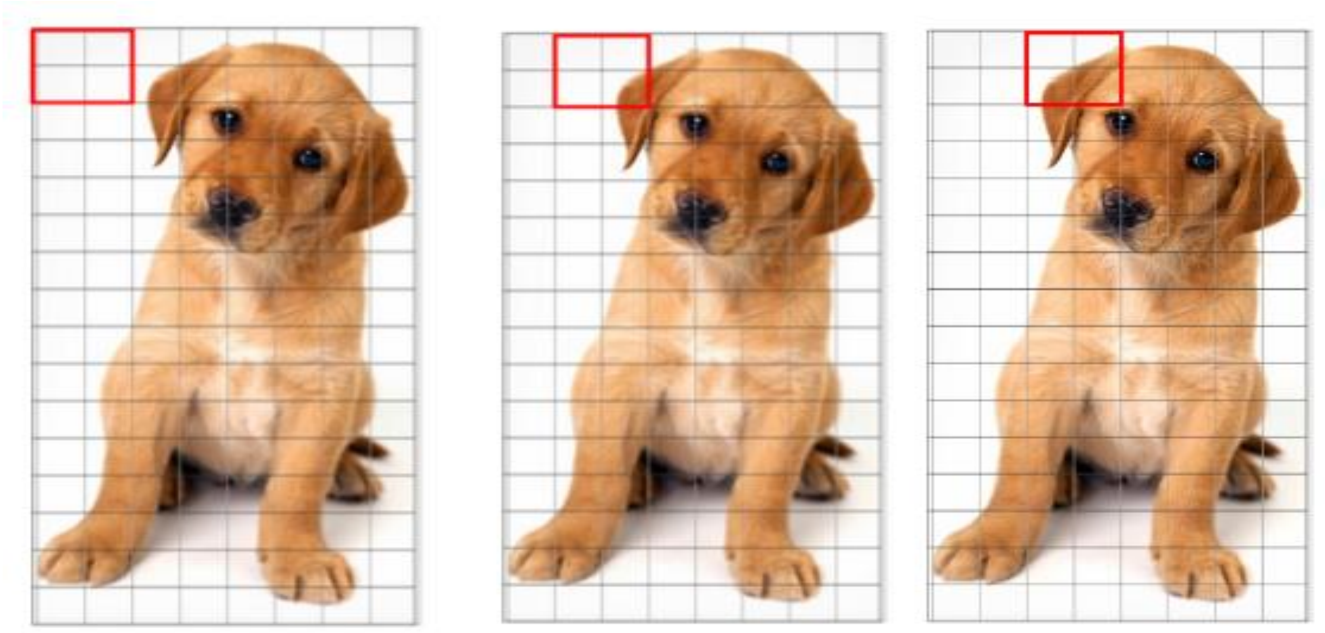
# Process of Calculating HOG (step 5)

- 16×16 block is moved in steps of 8 pixels ( i.e. 50% overlap with the previous block )



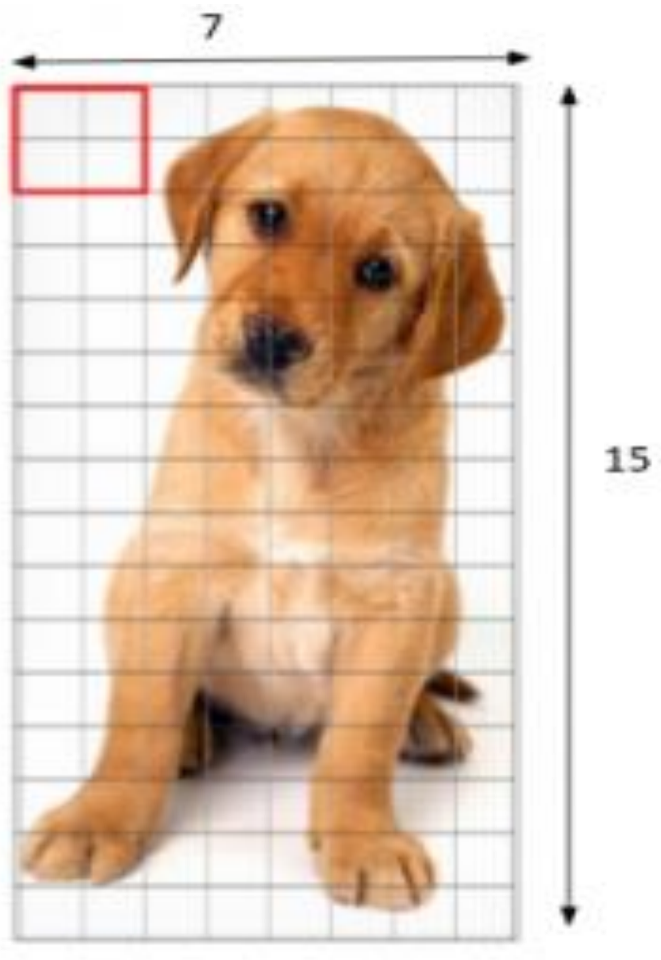
# Process of Calculating HOG (step 5)

- 16×16 block is moved in steps of 8 ( i.e. 50% overlap with the previous block )



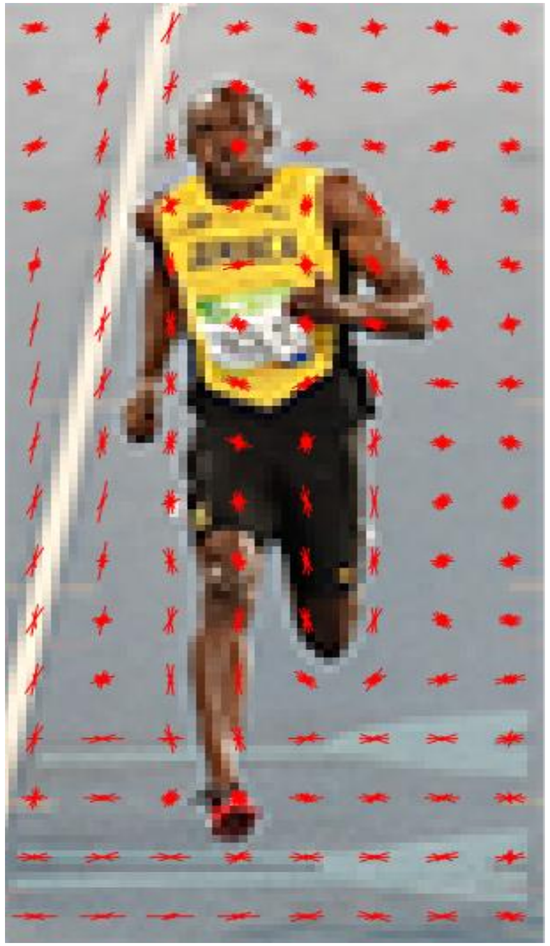
# Process of Calculating HOG (step 5)

- 16×16 block is moved in steps of 8 ( i.e. 50% overlap with the previous block )
- Number of blocks = 7 x 15 = 105
- For each block, there are 36 values
- Length of the final vector 105 x 36 = 3780

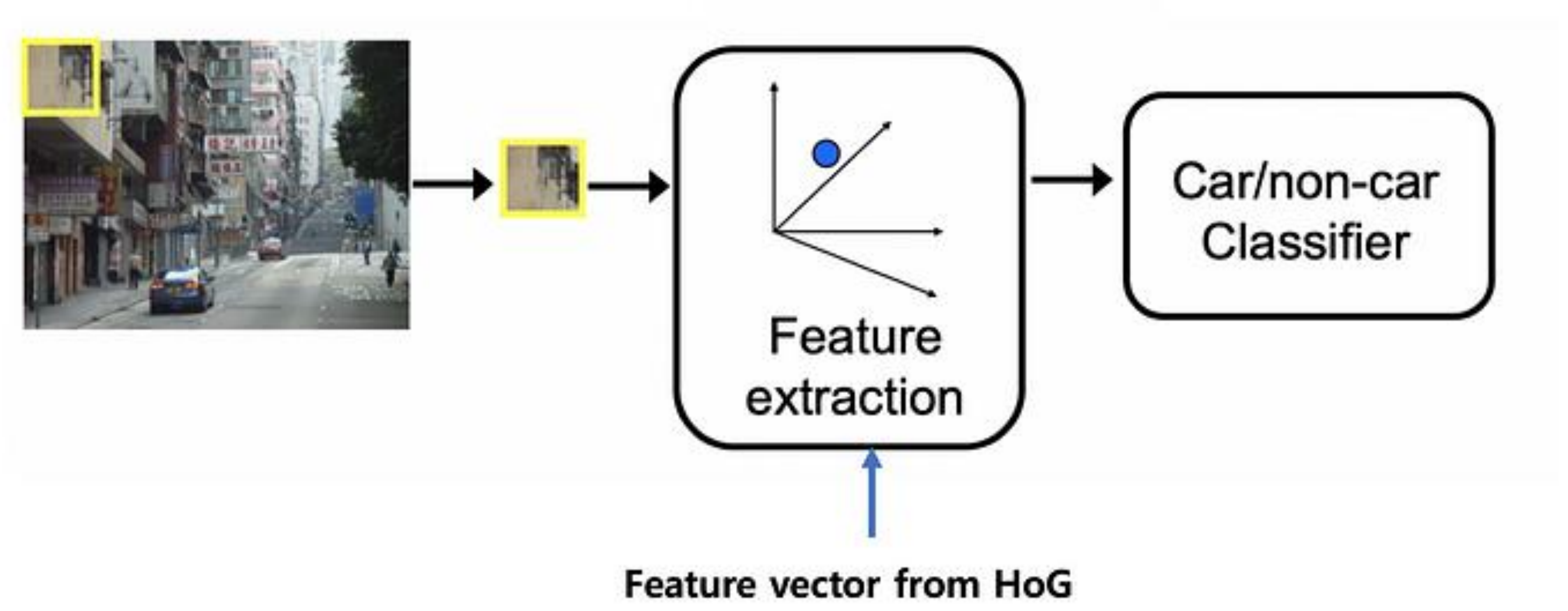


# Process of Calculating HOG (step 6)

- Step 6: Visualize HOG
  - Plot the 9×1 normalized histograms in the 8×8 cells
  - Dominant direction of the histogram captures the shape of the person
- Step 7: Classify images
  - Each image is represented by a descriptor (feature vector) of length,3780
  - Train classifier (ex: SVM) using descriptors of images



# Example of HoG descriptor for object detection



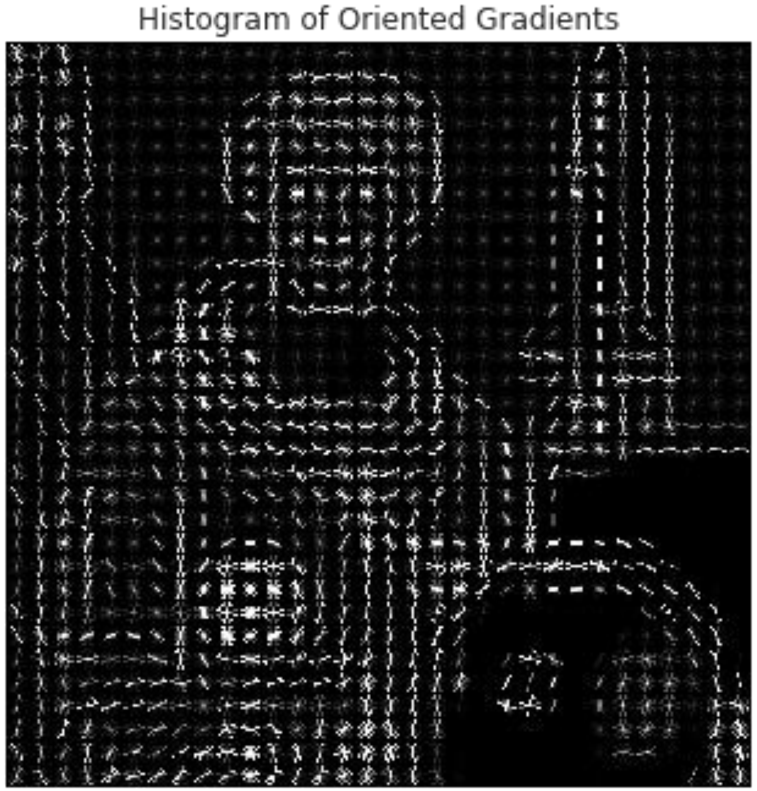

- Common choice of classifier for conventional object detection (without deep learning) is the SVM (support vector machine)

# HOG for face detection

Input image

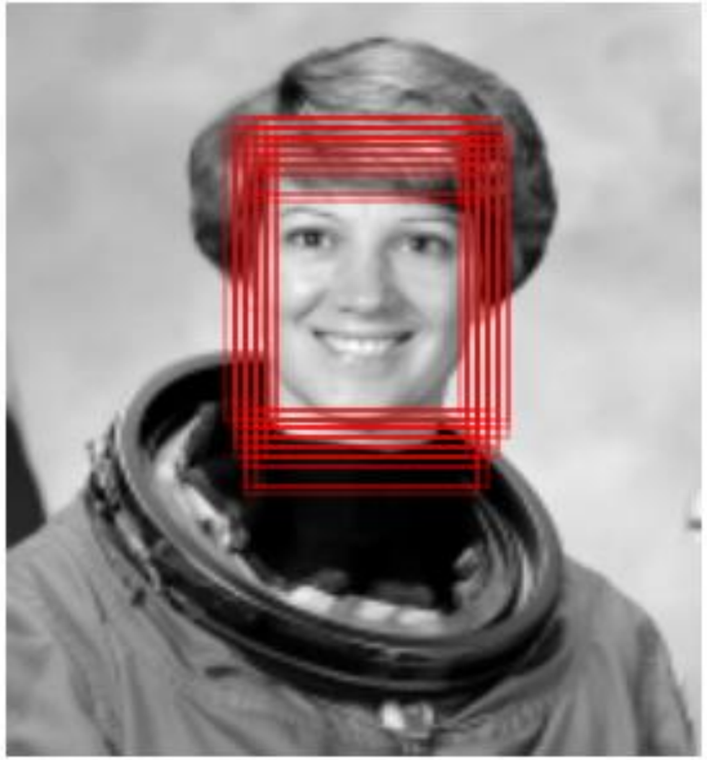
Determine HoG of images with faces and non faces

# HOG for face detection


Determine HoG of images with faces and non faces

# HOG for face detection

- Determine HoG of images with faces and non faces
- Train classifier for labelled images (face and non face)
- Test classifier for the given astronaut image

# References


- https://nptel.ac.in/courses/108103174
- https://www.oreilly.com/library/view/programming-computer-vision/9781449341916/ch02.html
- https://www.baeldung.com/cs/image-processing-feature-descriptors
- 
- https://sbme-tutorials.github.io/2018/cv/notes/9_week9.html
- https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/
- 
- https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f
- https://medium.com/@dnemutlu/hog-feature-descriptor-263313c3b40d
- https://learnopencv.com/histogram-of-oriented-gradients/
- https://debuggercafe.com/image-recognition-using-histogram-of-oriented-gradients-hog-descriptor/
- https://www.google.com/amp/s/iq.opengenus.org/object-detection-with-histogram-of-oriented-gradients-hog/amp/
- 
- https://www.google.com/url?sa=t&source=web&rct=j&url=https://m.youtube.com/watch%3Fv%3D5nZGnYPyKLU&ved=2ahUKEwjn2ISl1sz-AhUERmwGHe-aB3EQo7QBegQIDBAF&usg=AOvVaw3j2q__19MNeHimMyT1lewO
- https://nptel.ac.in/courses/108103174

- https://sbme-tutorials.github.io/2018/cv/notes/6_week6.html
- https://medium.com/data-breach/introduction-to-harris-corner-detector-32a88850b3f6
- https://www.baeldung.com/cs/harris-corner-detection
- https://www.codingninjas.com/codestudio/library/harris-corner-detection
-
- https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.cs.umd.edu/class/fall2019/cmsc426-0201/files/12_HarrisCornerDetection.pdf&ved=2ahUKEwj_q4fY5br-AhWu-jgGHeTBCJAQFnoECD8QAQ&usg=AOvVaw0WjY5eRFeu-vCUFu-g6o90
- https://fiveko.com/feature-points-using-harris-corner-detector/