Advanced Pattern Recognition

Template Matching
Viola-Jones Algorithm
Histogram of Oriented Gradients (HOG)
Deep Learning

Pattern Recognition

- **Objective**: to extract relevant information from images using image processing

Feature similarity	Histogram similarity	Model similarity	Advanced Classifiers
• area, perimeter, mean intensity, etc.	• local and / or global	• geometrical models (Hough) • Template Matching	• Viola-Jones algorithm • Histogram of Oriented Gradients (HOG) • Convolution Neural Networks (*Deep Learning*)

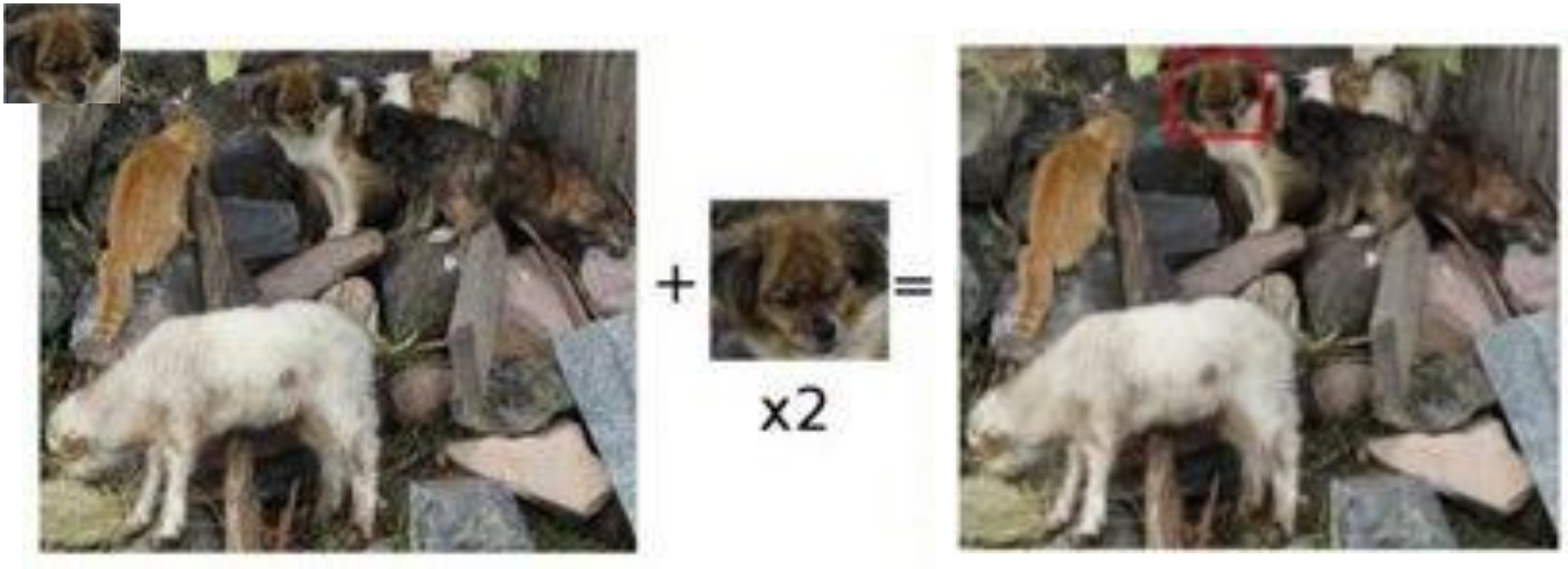
… and others

Template matching

Template Matching

- Search for matches between two or more images
- Can be performed by comparison of:
  - features
  - subparts of the image that contain information more discriminative ← More efficient
- When this is not possible, is done by assessing the similarity between the original image and one or more template images by convoluting both images
  - More precise
  - Less efficient

Example



Measures of similarity

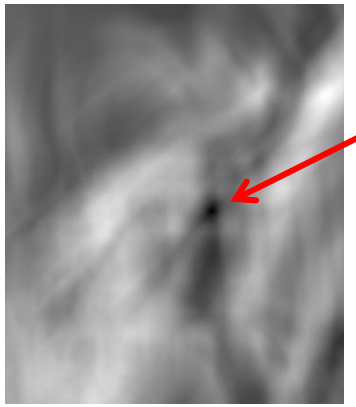
- Sum square difference

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

512 x 512 pixels

129 x 60 pixels

Template Matching



384 x 453 pixels

Min
2.0

(x,y) = (234,234)

(W – tw + 1) x (H – th + 1)

Measures of similarity

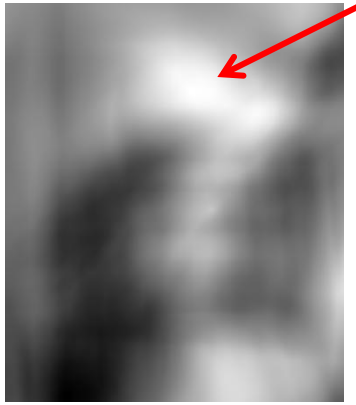
- Correlation

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x', y+y'))$$

512 x 512 pixels

129 x 60 pixels

Template Matching



384 x 453 pixels

Max
613557376.0

(x,y) = (213,100)

Measures of similarity

- Correlation Coefficient (correlation with mean = 0)

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))$$

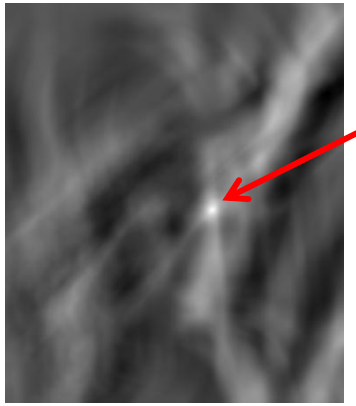
With

$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$$
$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'')$$

512 x 512 pixels

129 x 60 pixels

Template Matching



384 x 453 pixels

Max
55113824.0

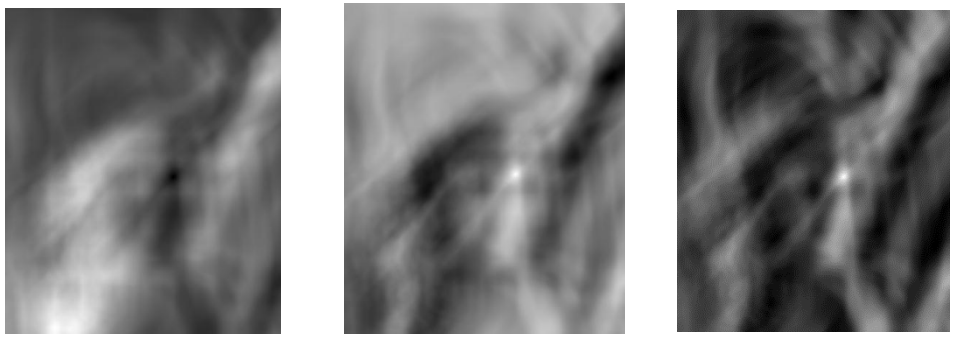
(x,y) = (234,234)

Normalized methods

- The previous methods produce images in the range [- ∞ , ∞], which can be difficult to work with
- **Normalization** generates images in the range [-1, 1], being 1 a total correspondence for correlation and coefficient of correlation, and 0 for the sum of squared differences

$$R_{cc\_norm}(x,y) = \frac{R_{cc}(x,y)}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

- Careful, because with normalization more similar regions might be detected



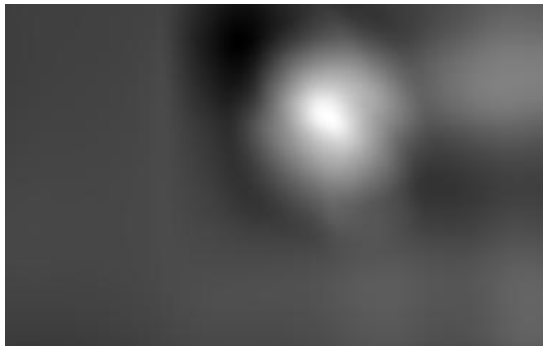
Results

Template

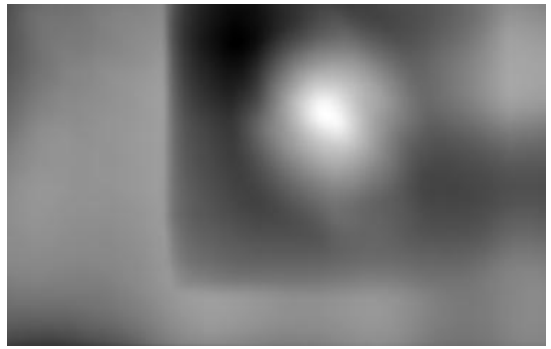
Correlation Coef

Correlation Coef Norm

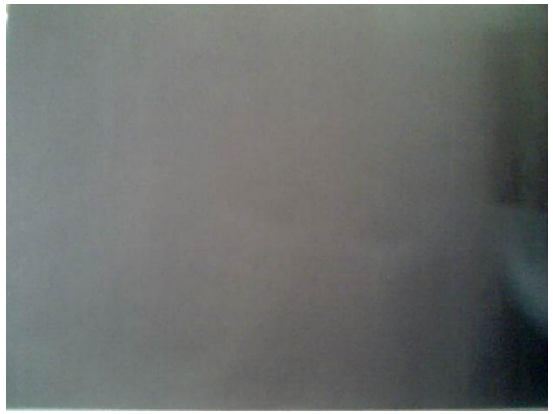
With object



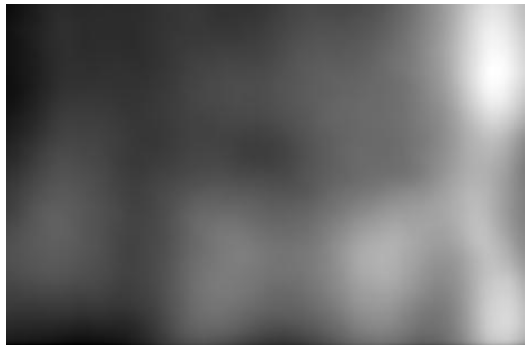
Max = 159 488 592.0



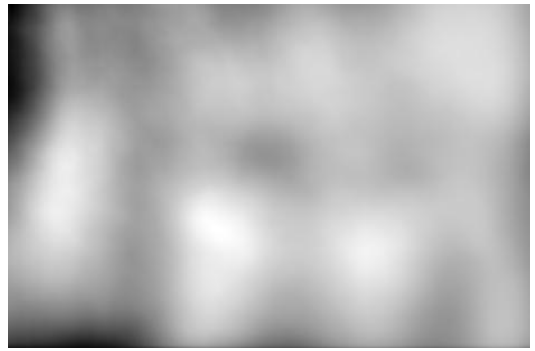
Max = 0.88



Without objects



Max = 33 790 024.0



Max = 0.47

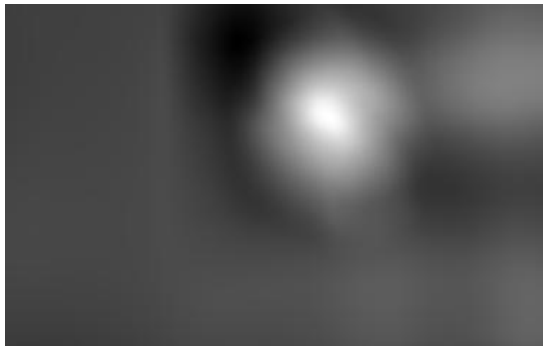
Results

Template

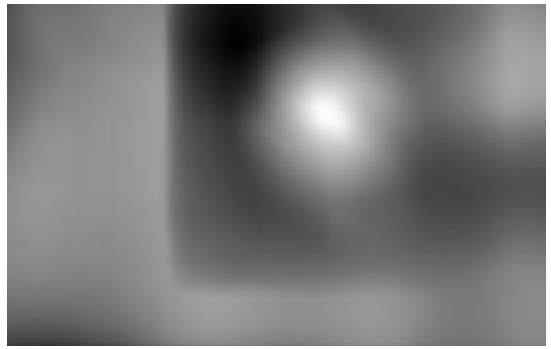
Correlation Coef

Correlation Coef Norm

With object

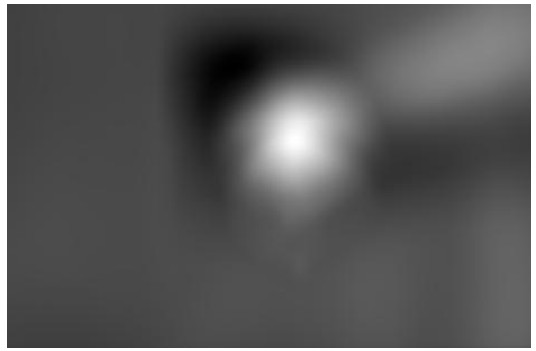


Max = 159 488 592.0

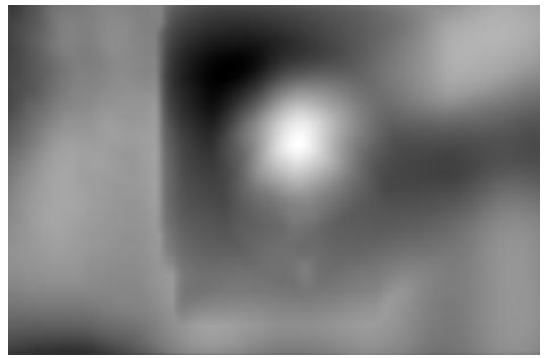


Max = 0.88

Without objects

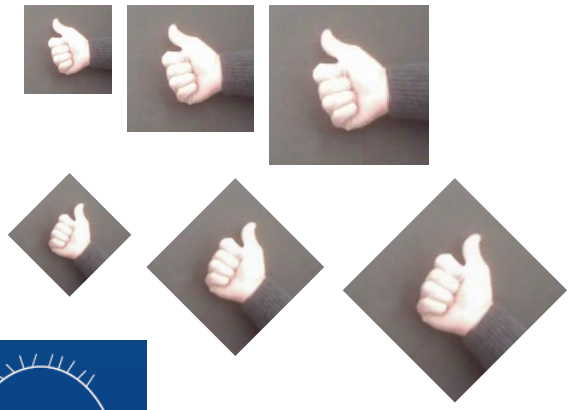


Max = 185 876 736.0

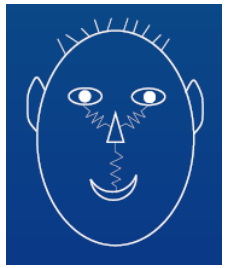


Max = 0.82

Problems



- High sensitivity to size, rotation and distortion changes
  - Solutions
    - Multi-level analysis: creating multiple templates with different configurations
    - Use flexible templates (several interconnected templates)

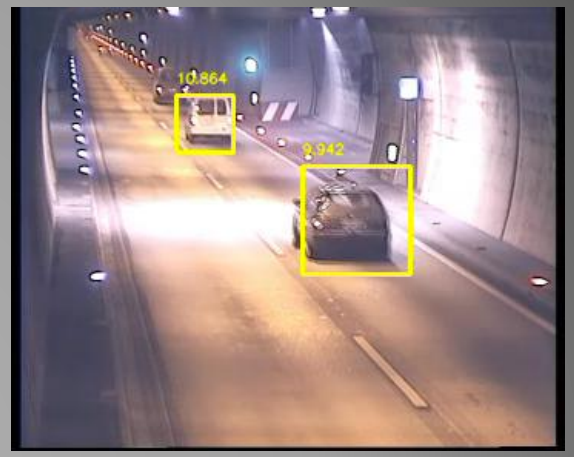


- Results with noise: Different image settings may generate false positives.
- High computation
  - Given that this is a convolution it is recommended to use FFTs
    - For an NxN image in space domain this method has a complexity of O(N4), while with FTTs it has O (N2 Log N)
  - Or start the algorithm with a low resolution and increase only in the most promising areas

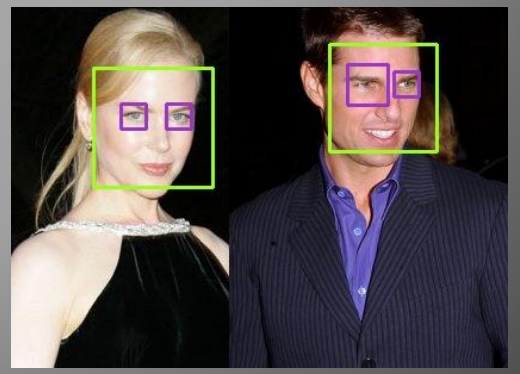
In OpenCV…

```
result = cv.MatchTemplate(image, template,  type)
```

TM_SQDIFF,
TM_SQDIFF_NORMED,
TM_CCORR,
TM_CCORR_NORMED,
TM_CCOEFF,
TM_CCOEFF_NORMED



Cascade Classifiers
Viola-Jones Algorithm



Introduction – Viola-Jones

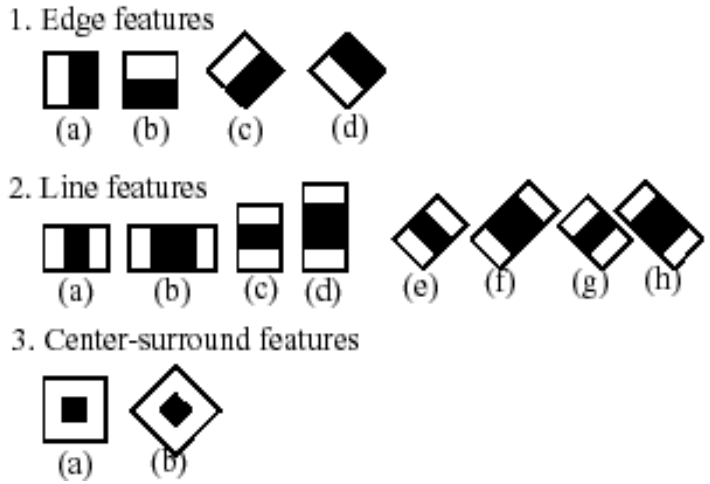
- Is a very effective algorithm for detection of patterns that can be used in real-time applications
- It was proposed in 2001 by Paul Viola and Michael Jones

> Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." *Proceedings of the CVPR 2001* Vol. 1. IEEE, 2001.

- Runs on 3 phases:
  - Feature extraction (rectangular Haar features)
  - Classification using cascaded classifiers with boosting
  - Repeat the process with different scale factors
- The training of the classifier uses an improved version of the AdaBoost algorithm (Yoav Freund e Robert Schapire)

Feature Extraction

- Are used the Haar-like Features:
  - Rectangular
  - You can use all or only subsets
- The feature value is obtained by subtracting the sum of all pixels of white region to the same value from the black region
- The implementation is very efficient if you use the integral image to do the calculations



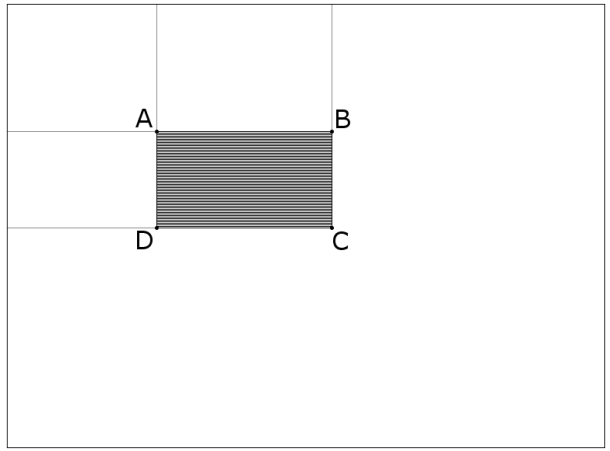
$$haar = \sum white - \sum black$$

Integral Image

- For each pixel (x,y) is determined the integral of the área between (0,0) and (x,y)

$$I(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} i(x',y')$$

- The integral of a rectangle (A, B, C, D) can be determined by analyzing only 4 points regardless of the size of the rectangle



$$\sum_{\substack{x0 < x \le x1 \\ y0 < y \le y1}} i(x,y) = I(C) + I(A) - I(B) - I(D).$$

Original				
5	**2**	**3**	4	1
1	**5**	**4**	2	3
2	2	1	3	4
3	5	6	4	5
4	1	3	2	6

Integral				
5	7	10	14	15
6	13	**20**	26	30
8	17	25	34	42
11	25	39	52	65
15	30	47	62	81

$5 + 2 + 3 + 1 + 5 + 4 = 20$

Original				
5	2	3	4	1
1	**5**	**4**	**2**	3
2	**2**	**1**	**3**	4
3	5	6	4	5
4	1	3	2	6

$5 + 4 + 2 + 2 + 1 + 3 = 17$

Integral				
5	7	10	**14**	15
6	13	20	26	30
8	17	25	**34**	42
11	25	39	52	65
15	30	47	62	81

$34 - 14 - 8 + 5 = 17$

Learning in the Viola-Jones

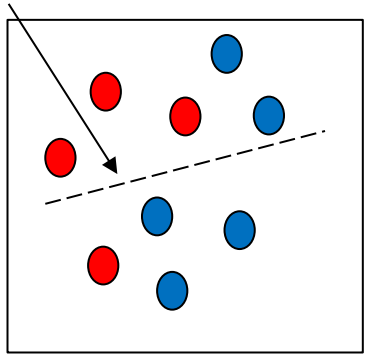
- Strong classifiers are created with a better selectivity than a random process

- Boosting:
  1. All examples initially have identical weights
  2. Choose the classifier that best separates the current DataSet
  3. Misclassified examples have an increase in their weight
  4. Repeat the process from point 2
  5. Strong classifier is a linear combination of all the weak classifiers

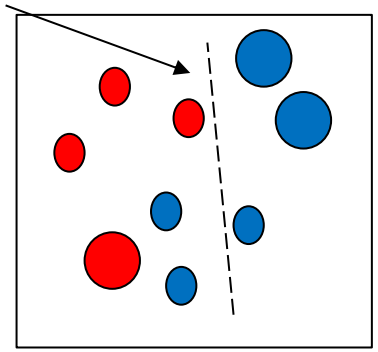
- The regions that are rejected will not be reviewed again

Boosting

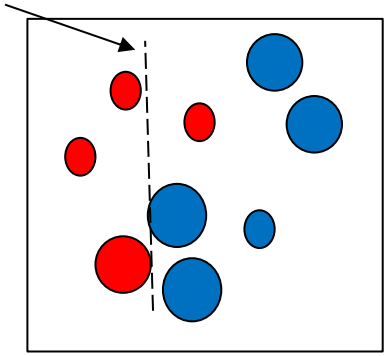
Weak Classifier 1



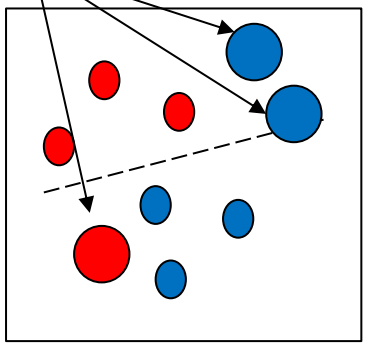
Weak Classifier 2



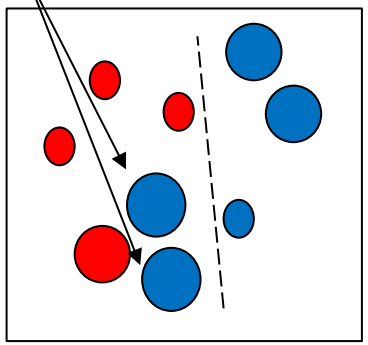
Weak Classifier 3



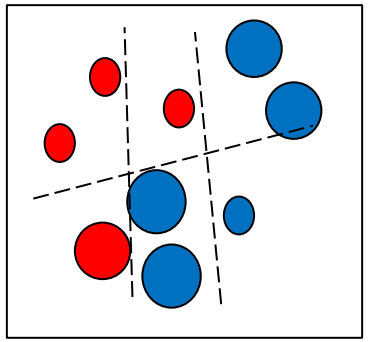
Increase weights



Increase weights

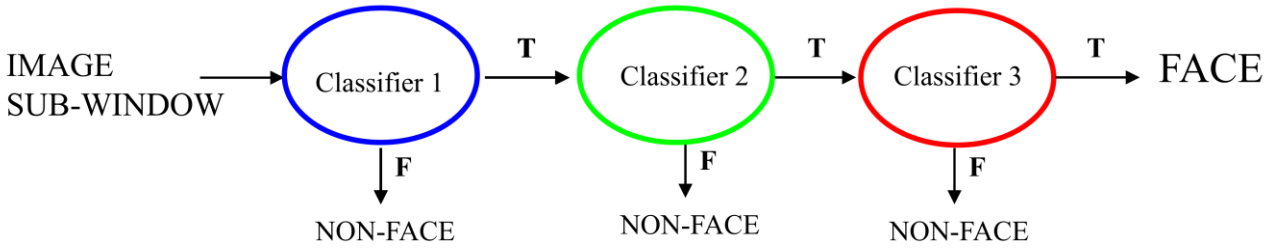


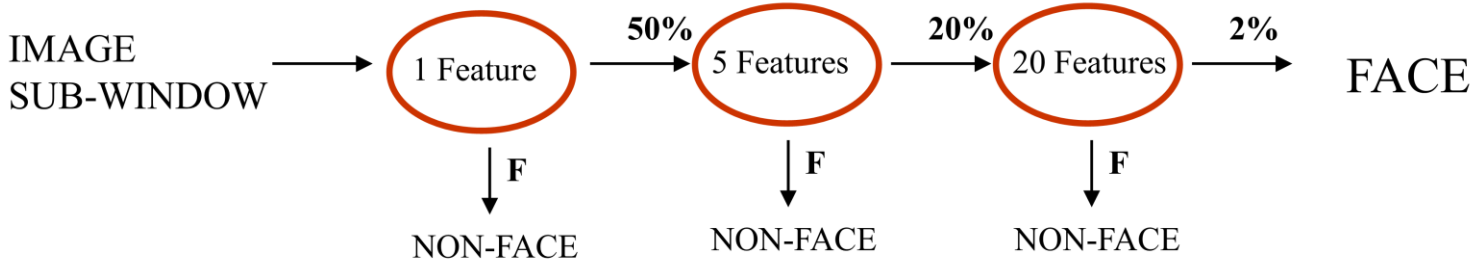
Strong Classifier



Cascade architecture

- Each strong classifier has a low performance with false acceptance rates in the order of 50%

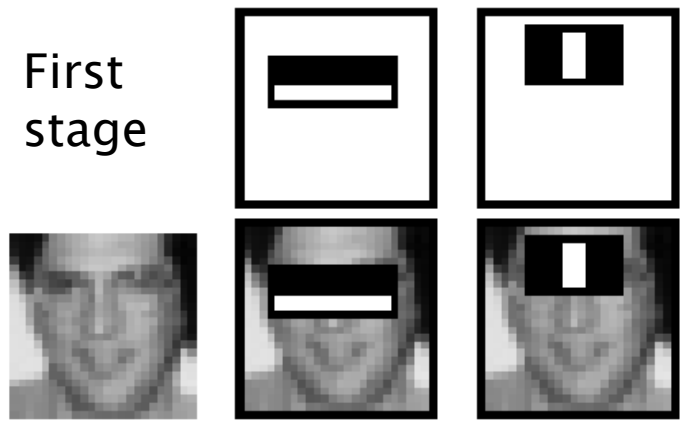




Training

- The most popular Face Detector was trained with:
  - 5000 faces rescaled to 24x24 pixels
  - 300 Million non-faces (8500 images)

- **Duration** of training: weeks…
- 23 classification stages
- First stage uses only 2 features
- Last stage with 192 nodes
- 0.067s in microprocessor of 700Mhz



In OpenCV…

```
detectorFilename = "haarcascade_frontalface_default.xml"
haar = cv.CascadeClassifier(pathname + detectorFilename)
faces = haar.detectMultiScale(
        imgOriginal,
        scaleFactor = 1.4,
        minSize = (20, 20),
        maxSize = (100,100) )

for (x,y,w,h) in faces:
        imgOriginal = cv.rectangle(imgOriginal, (x,y), (x+w,y+h), (0, 0, 255), 4)
```

Pretrained classifier

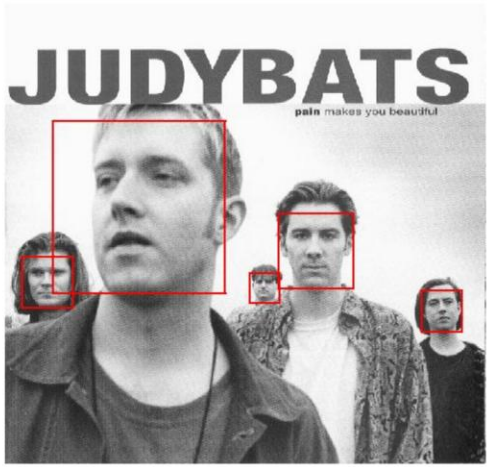
Gray Image

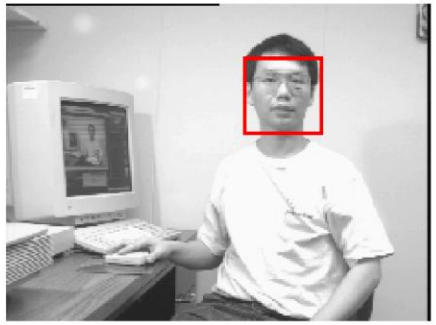
Scale Factor (between iterations)

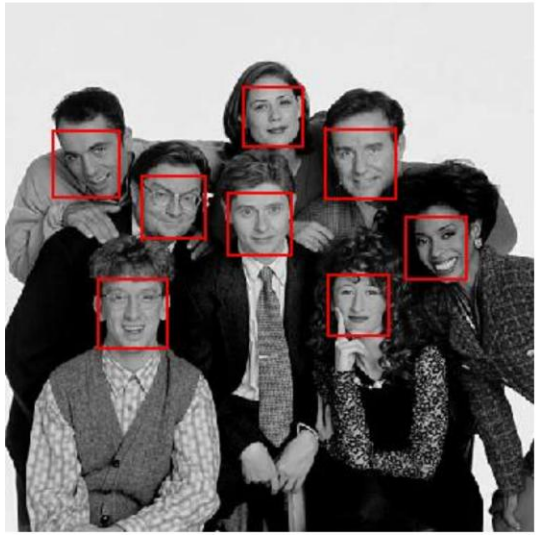
Minimum Size

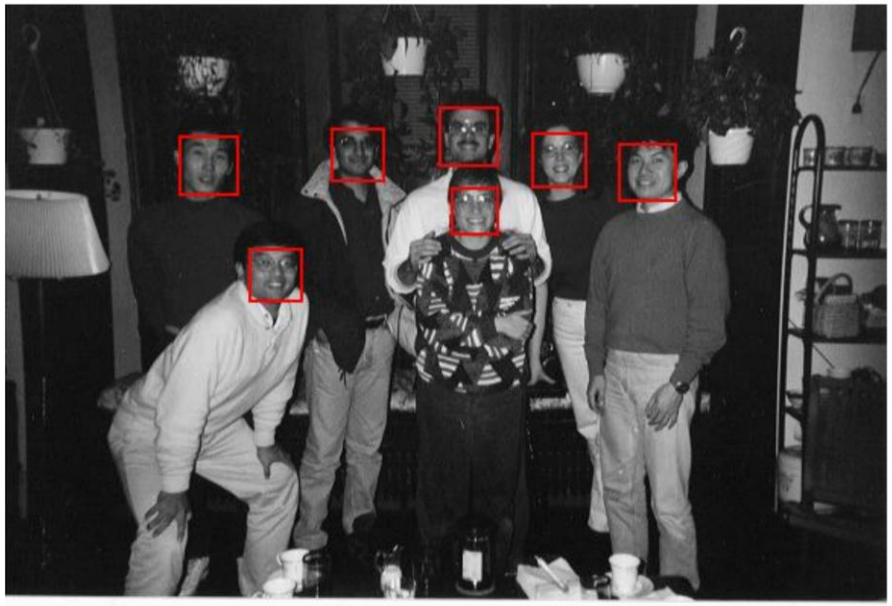
Maximum Size

Examples



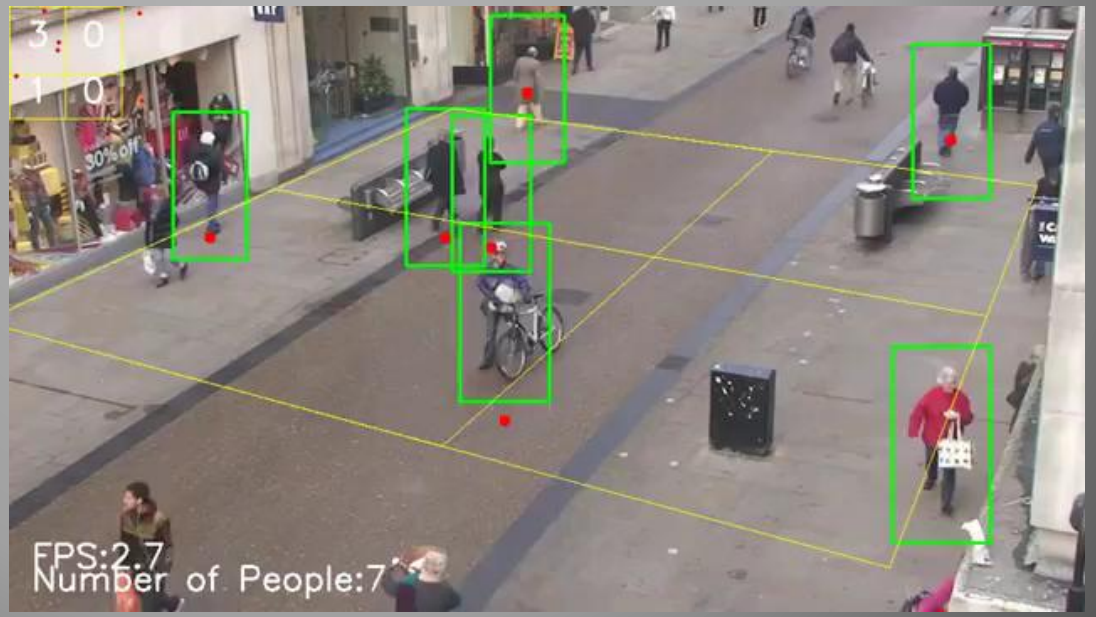






Other examples

Histogram of Oriented Gradients (HOG)

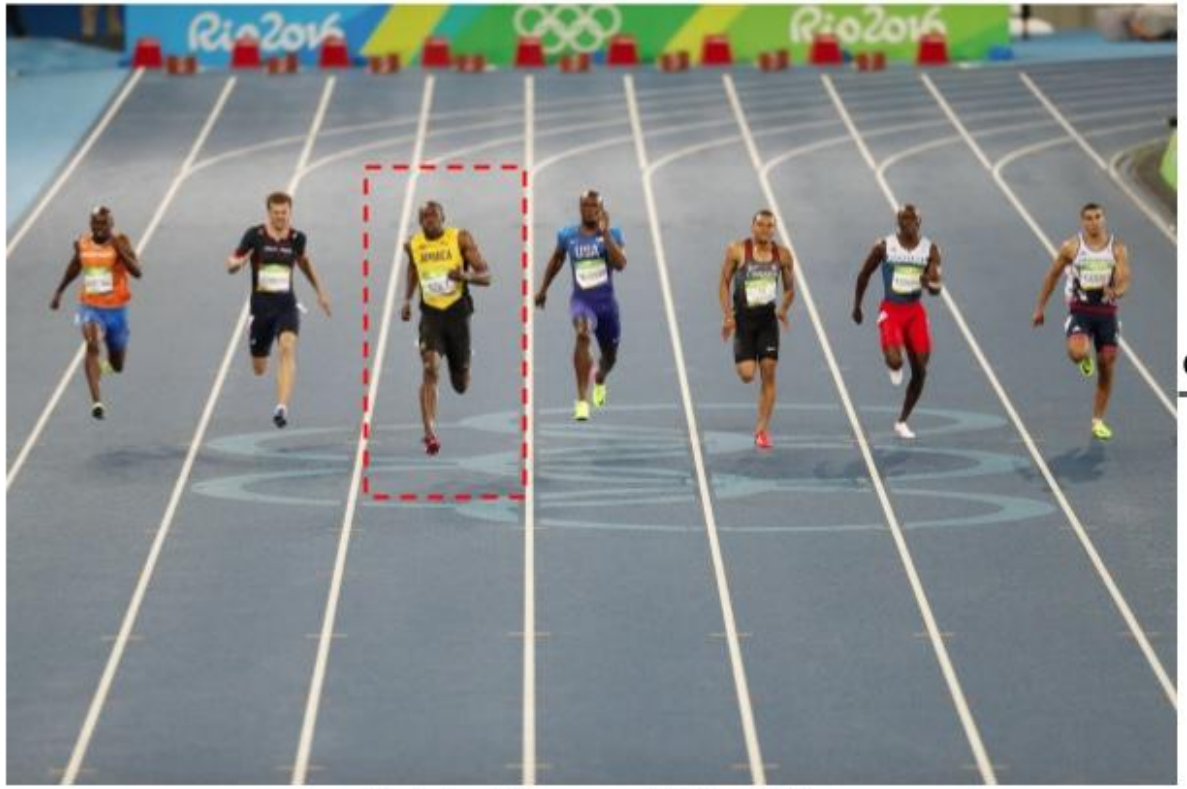


Histogram of oriented gradients

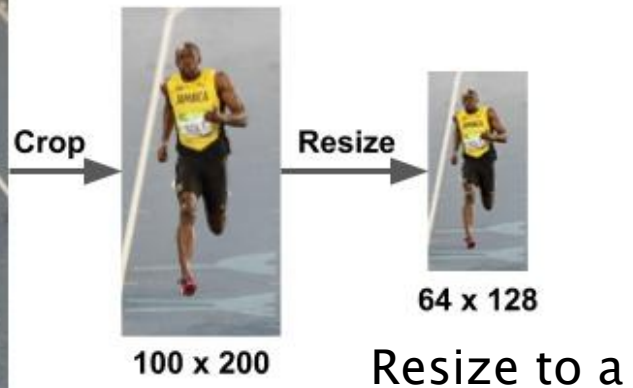
- It is a feature descriptor used for object detection
- Extracts information about the image in the form of a vector of features
- The vector is later used by machine learning algorithms for object recognition
- It extracts information about the image gradient

Source: Learning OpenCV
https://www.learnopencv.com/histogram-of-oriented-gradients/

Step 1 – Cut out a sliding window



Original Image : 720 x 475



Resize to a preset size

Step 2 – gradient calculation

- Calculation of X and Y gradients

-1	0	1

-1
0
1

- Obtain orientation and magnitude

$$g = \sqrt{g_x^2 + g_y^2}$$

$$\theta = \arctan \frac{g_y}{g_x}$$

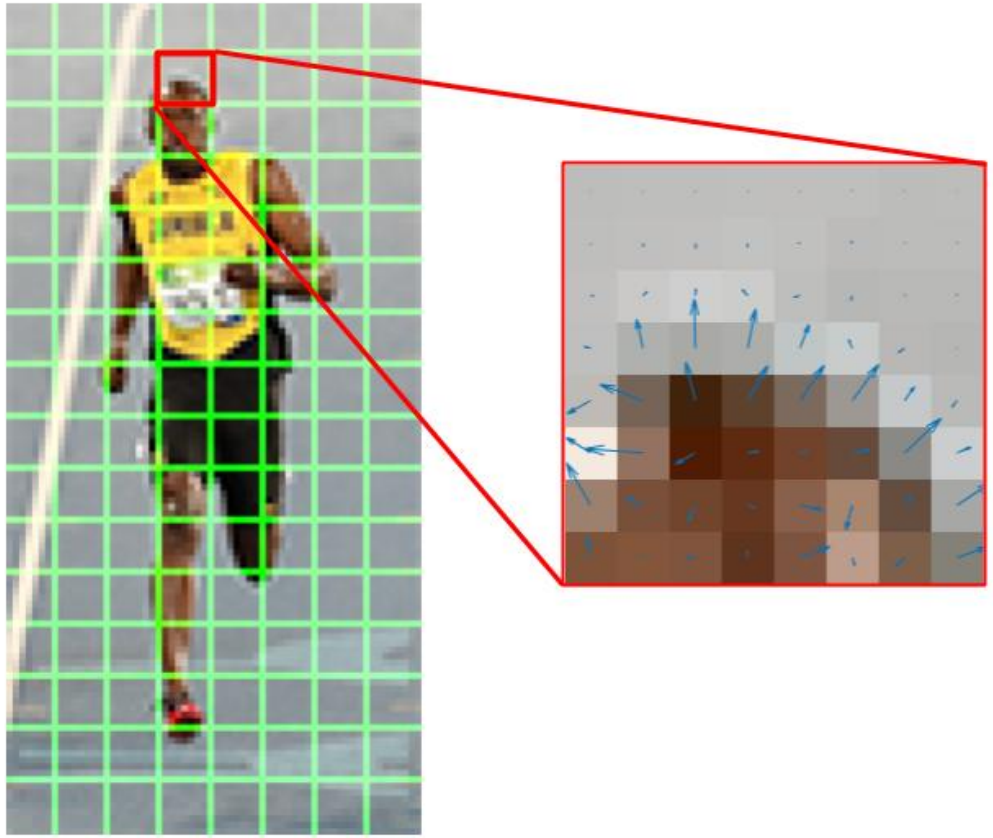
X

Y

X + Y
(sobel)

Step 3 – Calculation of histograms of oriented gradients

- Applied on each 8x8 pixel window



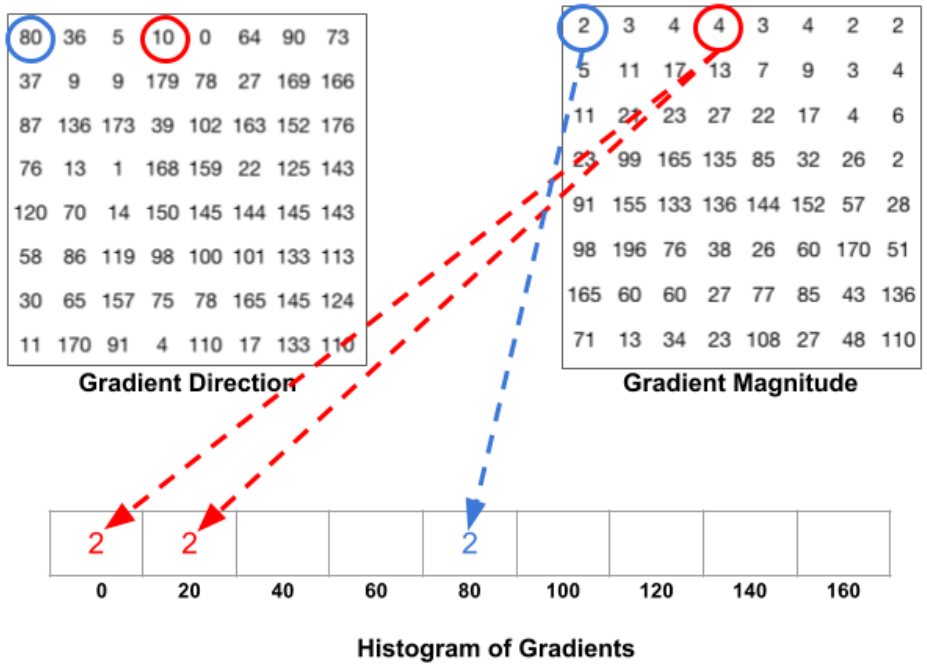
2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

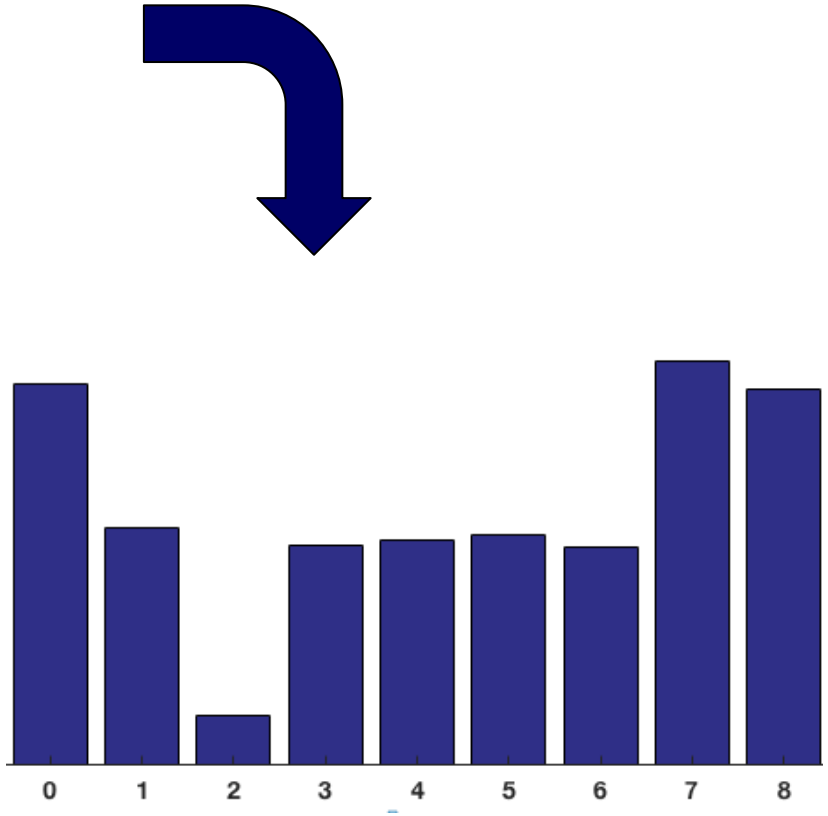
Gradient Magnitude

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

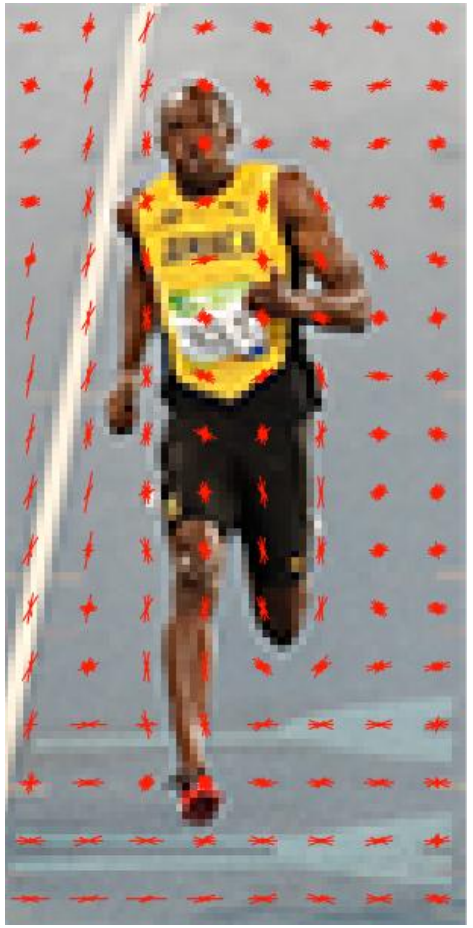
Step 3 – Calculation of histograms of oriented gradients





Steps 4 and 5- Normalization and Feature Vector

- A normalization of the various gradient histograms is performed to reduce sensitivity to lighting variations
- Normalization is performed in blocks of 16x16 pixels
- The final feature vector is composed of:
  - Blocks 16x16 = 15x7 = 105
  - Each 16x16 block contains 4 histograms of 9 positions = 4x9 = 36 positions
  - Final = 105 * 36 = 3780 positions



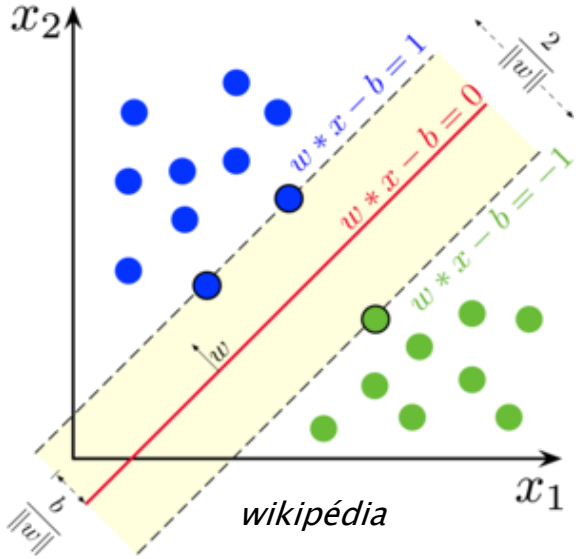
Step 6 – Classification

- The most common application uses Support Vector Machines (SVM) for pedestrian's detection
  - SVM is a Linear combination of all input values (x) and their synaptic weights (w) and a bias (b)
  - Defines a hyperplane that separates two classes

$$w * x - b = 0$$

Let

$$x = (x0, x1, \ldots., xn)$$
$$w = (w0, w1, \ldots., wn)$$



wikipédia

In OpenCV…

```
hog = cv.HOGDescriptor()
hog.setSVMDetector(cv.HOGDescriptor.getDefaultPeopleDetector() )
pedestrians = hog.detectMultiScale(imgOriginal)

for (x,y,w,h) in pedestrians[0]:
 … DRAW RECTANGLES
```

Classifier trained for detection of pedestrians

Image

It has many parameters
Additional

Example