

COMPUTATIONAL VISION: Image Features (HOGs)

Master in Artificial Intelligence

Department of Mathematics and Computer Science

2023-2024



UNIVERSITAT DE
BARCELONA

Outline

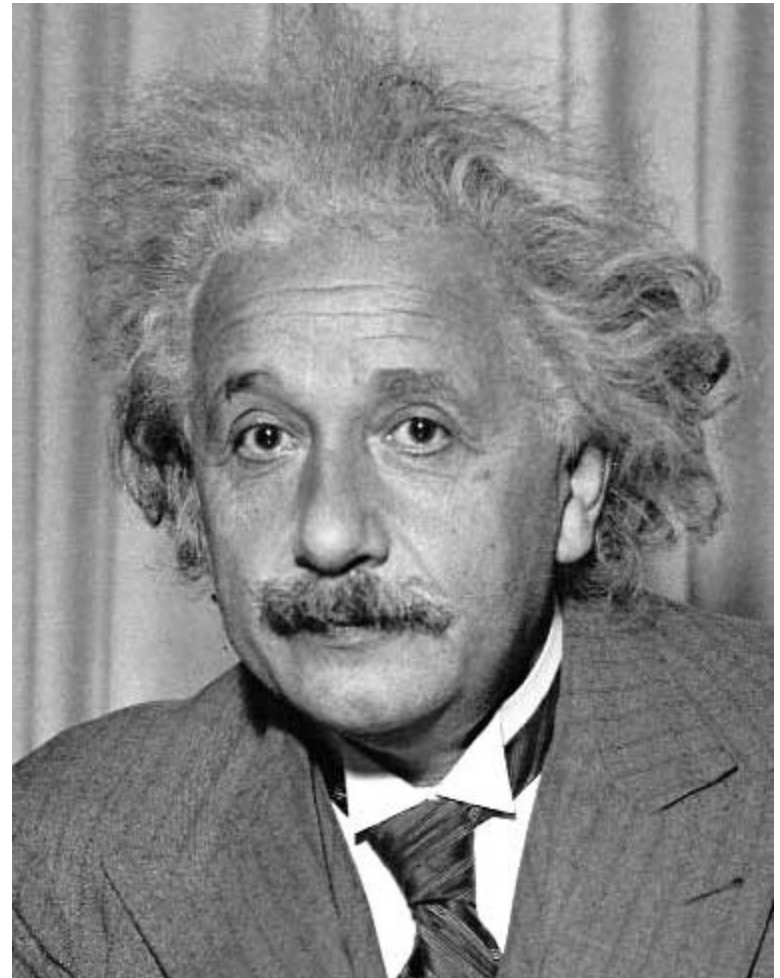
- Template Matching
- Image descriptors: what are and why we need them?
- A particular kind of image descriptor (HOG)
- Image retrieval
- Pedestrian detection with HOG

Template matching


Goal: find the template  in the image

Method: “Compare” the template with patches of the image

Main challenge: What is a good similarity or distance measure between two patches?

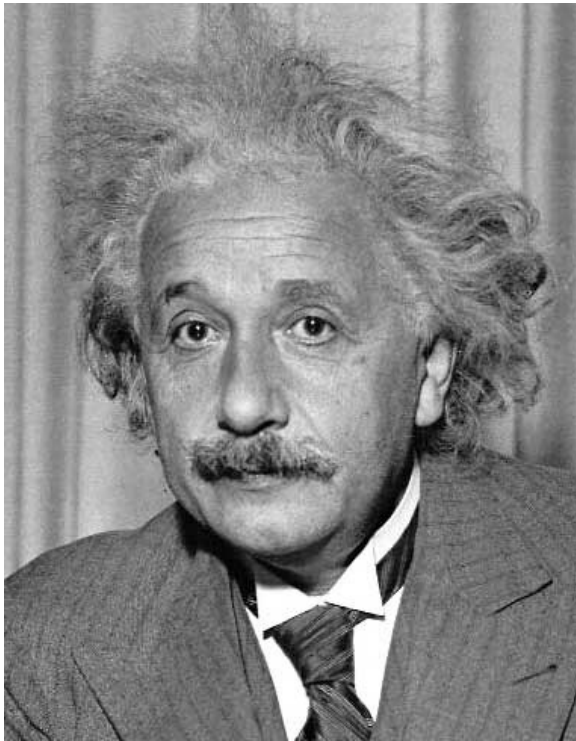


Template Matching with filters

- Goal: find  (= g) in the image (= f)
- Method 1: Sum Square Difference (SSD)

It seems to work fine. BUT ... What will happen if the local brightness changes?

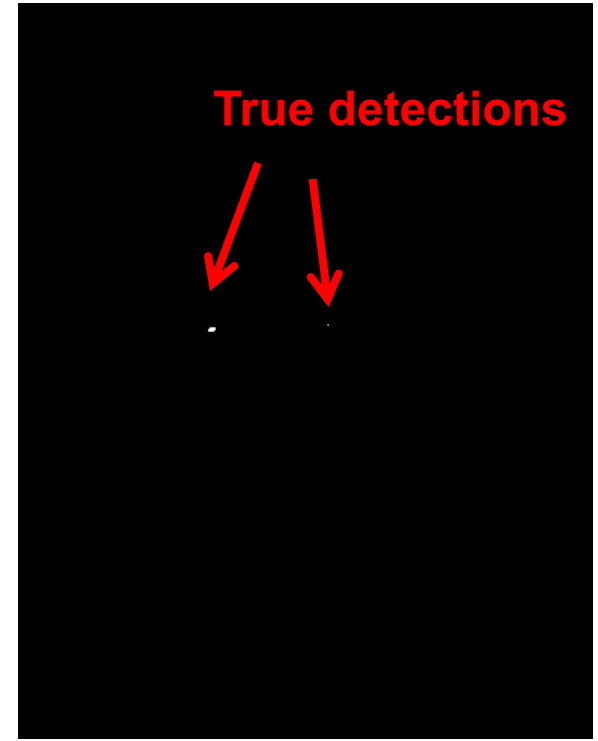
$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$



Input (f)




$1 - \sqrt{\text{SSD}}$

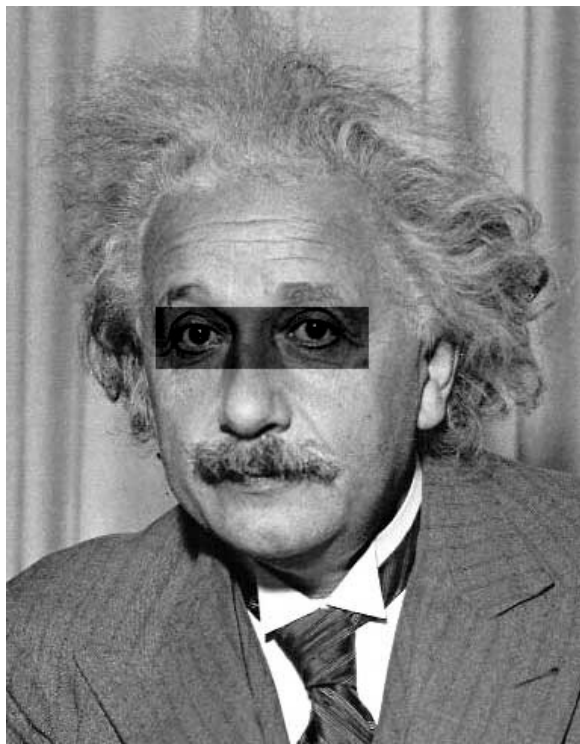


Thresholded Image

Template Matching with filters

- Goal: find  in image with changed contrast
- Method 1: Sum Square Difference (SSD)

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$



Input: Change local brightness



1- sqrt(SSD)

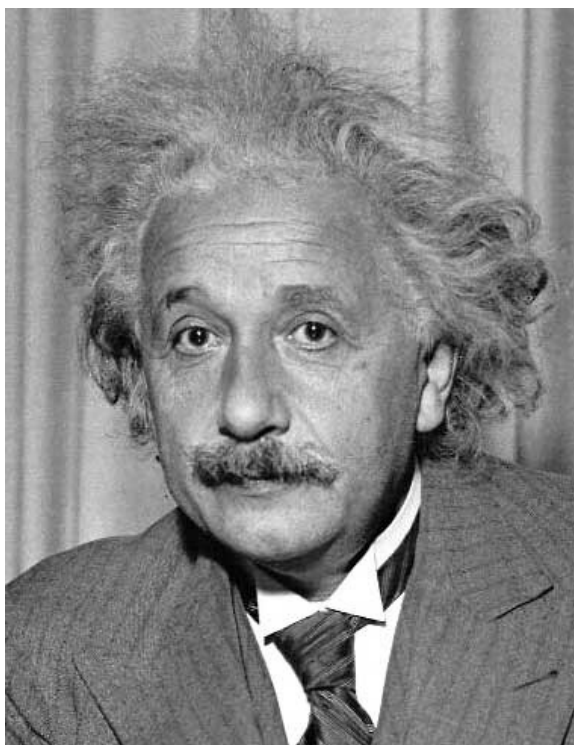
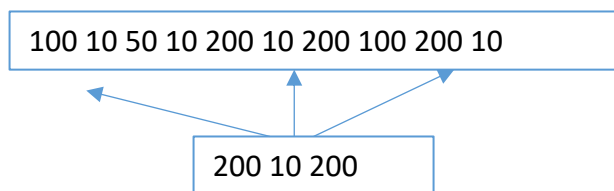
Not robust to changes
in brightness

Template Matching with filters

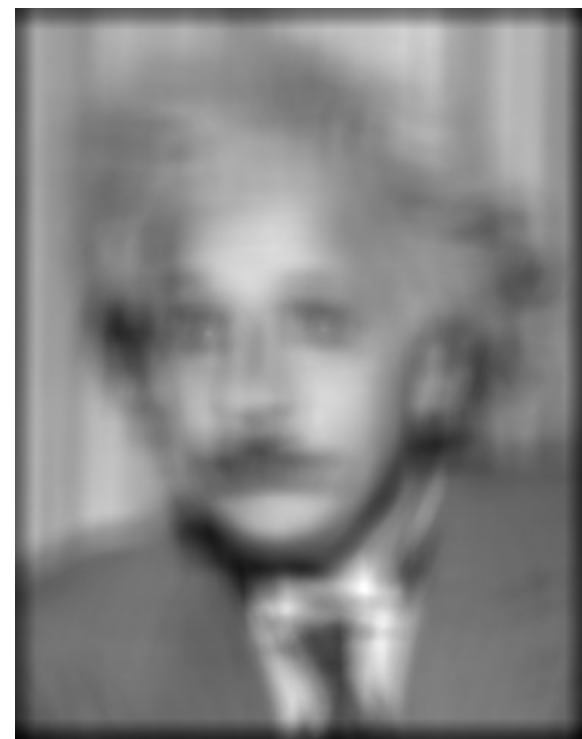
- Goal: find  in image
- Method 2: Convolutional filtering the image with the template (eye patch)

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l] \leftarrow \begin{array}{l} f = \text{image} \\ g = \text{filter} \end{array}$$

output k,l



Input



Filtered Image

What went wrong?

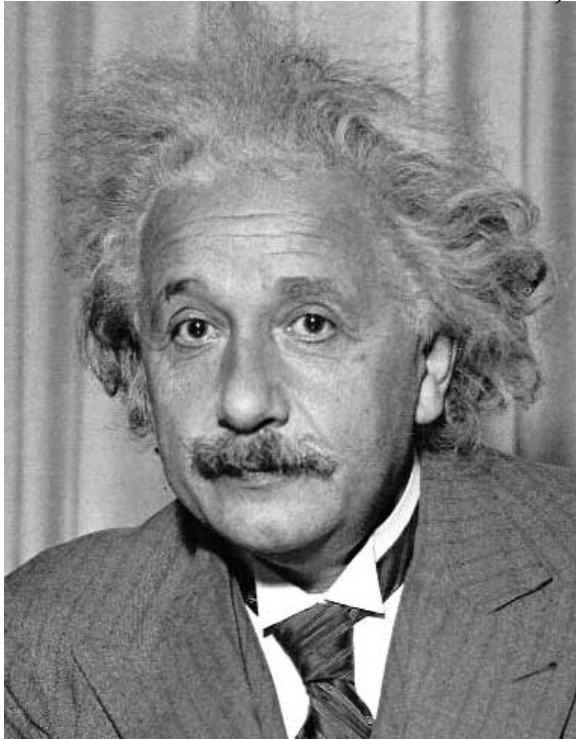
Higher for higher intensities,
needs normalization.

Template Matching with filters

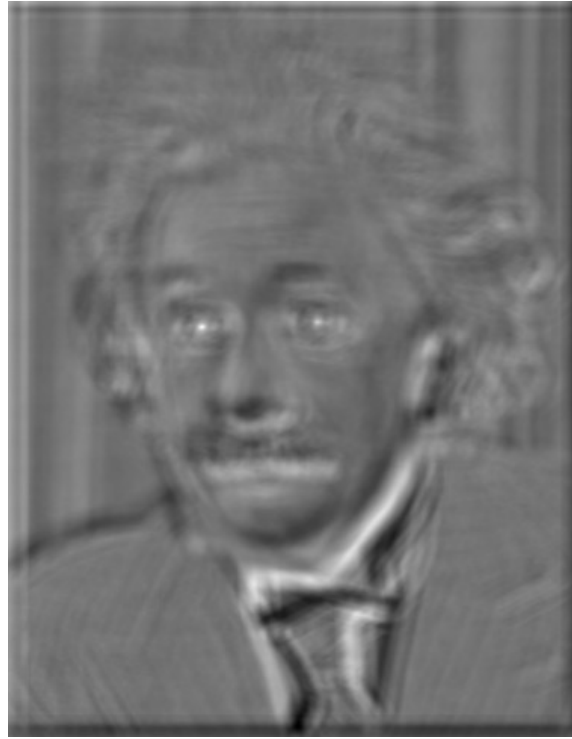
- Goal: find  in image
- Method 3: Convolutional filtering **the normalized image**

$$h[m,n] = \sum_{k,l} (g[k,l])(f[m+k,n+l] - \bar{f})$$

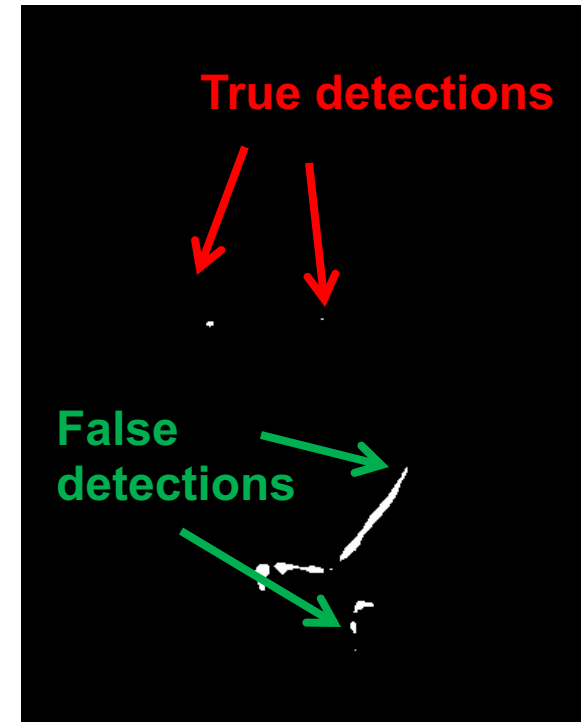
← mean of f



Input

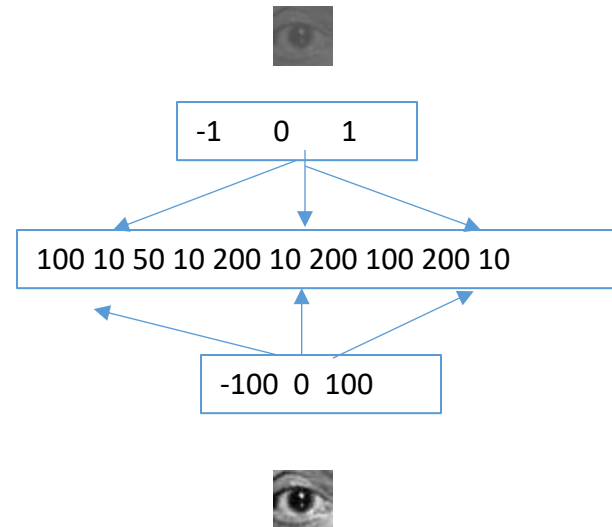
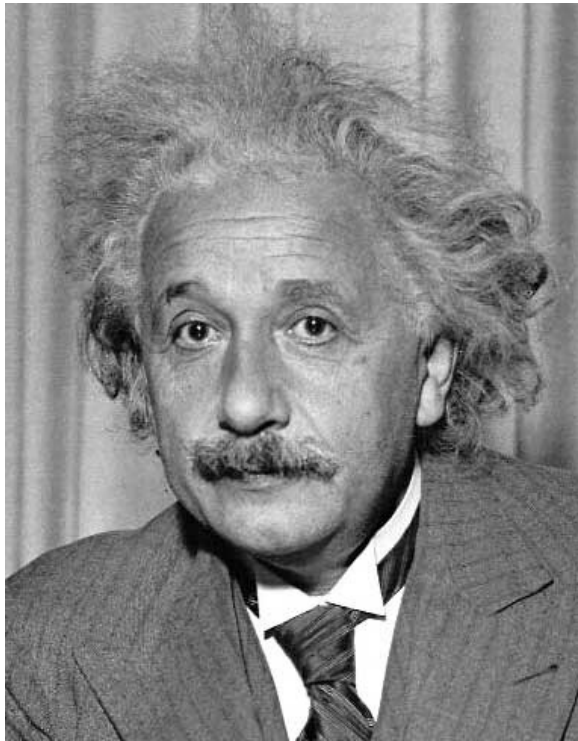


Filtered Image (scaled)



Thresholded Image
Sensitive to high contrast areas

How to make it independent of the template variance?



Template Matching with filters

- Goal: find  in image
- Method 4: Using **Normalized cross-correlation** (NCC)

$$h[m, n] = \frac{\sum_{k,l} (g[k, l] - \bar{g})(f[m - k, n - l] - \bar{f}_{m,n})}{\left(\sum_{k,l} (g[k, l] - \bar{g})^2 \sum_{k,l} (f[m - k, n - l] - \bar{f}_{m,n})^2 \right)^{0.5}}$$


template mean image patch mean

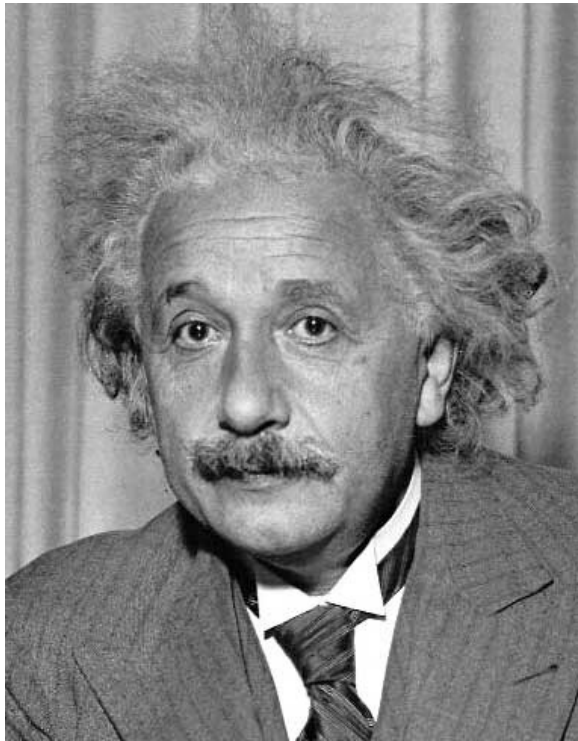
↓ ↓

Skimage:

```
result = match_template(image, template)
ij = np.unravel_index(np.argmax(result), result.shape)
x, y = ij[:-1]
```

Template Matching with filters

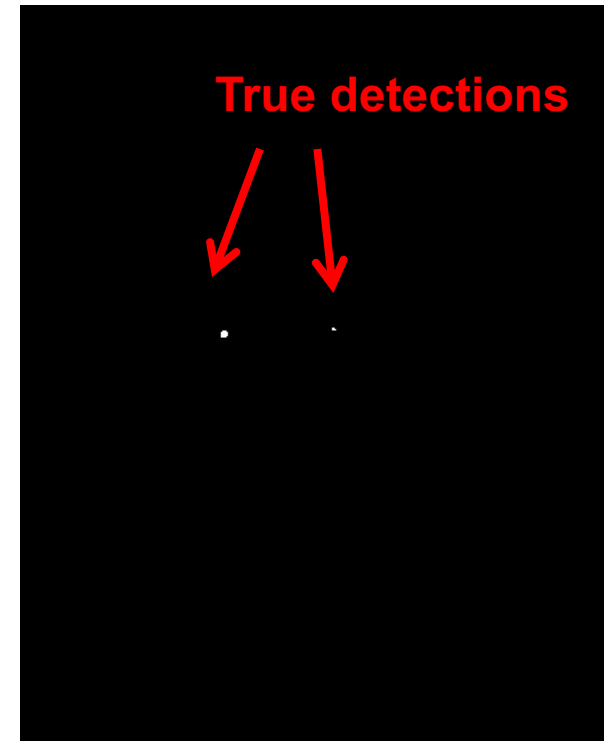
- Goal: find  in image
- Method 4: Normalized cross-correlation



Input




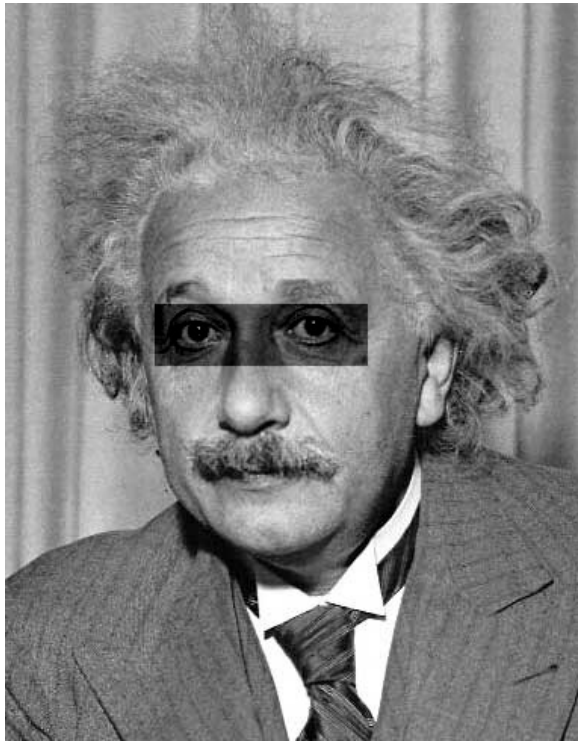
Normalized X-Correlation



Thresholded Image

Template Matching with filters

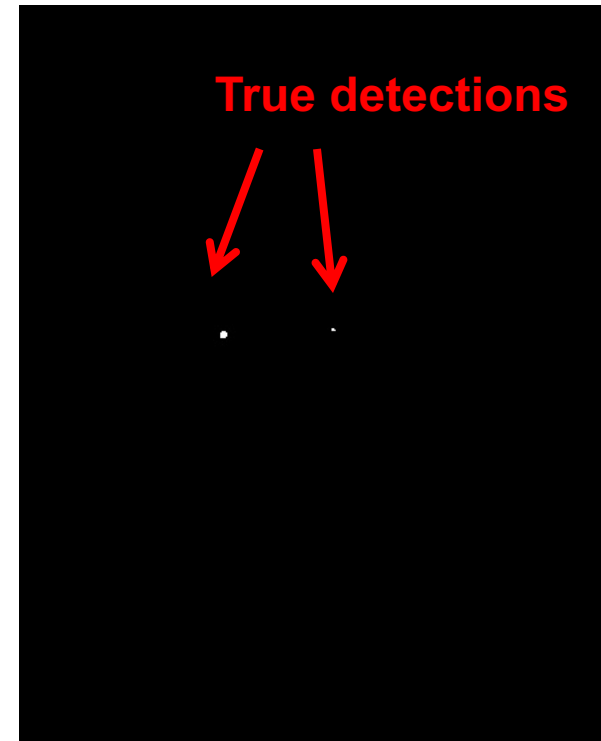
- Goal: find  in image
- Method 4: Normalized cross-correlation



Input: Change local brightness



Normalized X-Correlation



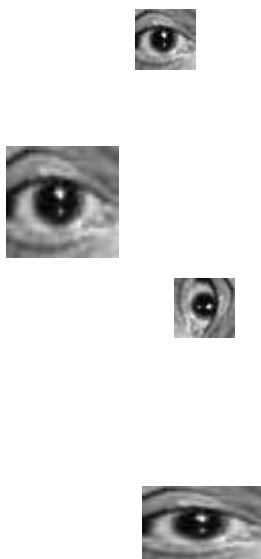
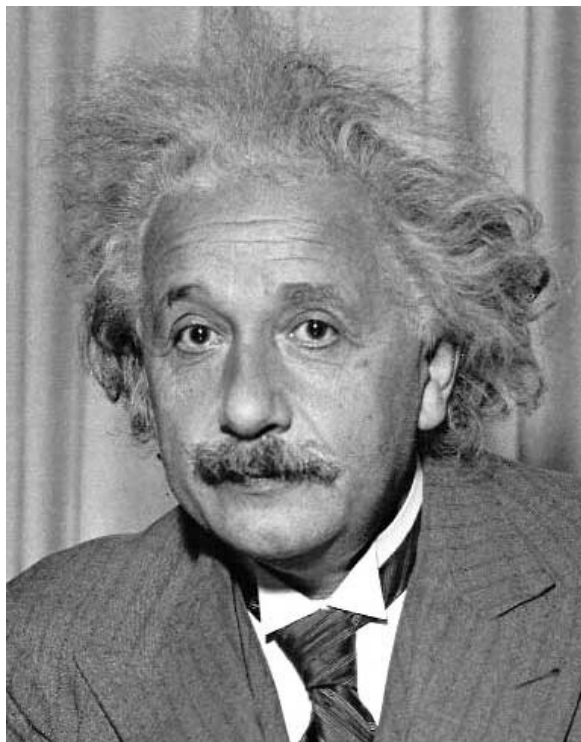
Thresholded Image

Template matching

Similarity or distance measures between two patches:

Distance	Properties
Sum Square Difference	Fast, but sensitive
Correlation	Less sensitive to illumination
Zero-mean correlation	Less sensitive to mask values
Normalized Cross Correlation	Less sensitive to mask variance

When the template matching will (not) work?



Outline

- Template Matching
- **Image descriptors: what are and why we need them?**
- A particular kind of image descriptor (HOG)
- Image retrieval
- Pedestrian detection with HOG

Problem 1: Image Description

Given an image, how can we automatically construct a description in order to **describe objects** and discriminate between 'building' and 'nature' images?



Which **visual characteristics** of the image can be used for this goal?

Problem 2: Image Retrieval

Given a 'building' image (query), how to retrieve other 'building' images in a database?

→ Given an image (query image), the image retrieval consists of sorting the rest of images according to the similarity to the query image.

Query image



Database



Why we need a feature?

To solve real world problems (image retrieval, image classification, etc.), we need to find a connection between:

- A matrix of pixels (raw representation),
- What humans see in an image (face, smile, emoticon).



shape
color
....

face
smile
emoticon

Image descriptors or features allow to describe and represent the image by quantities (color, shape, regions, textures and motion) closer to the visual characteristics perceived by humans.

Algorithm for image retrieval

Query image



Database



Algorithm:

1. Define the image descriptor
2. Extract the image descriptors of the database images
3. Given a query image, extract its descriptor vector
4. Sort the database images according to the similarity with the query image descriptor.

General approach to retrieval problems

Let's suppose for now that the descriptor is simply **the mean color**.



Building



Building



Nature



Nature



**'Building' or
'Nature'?**

Algorithm:

1. Define the image descriptor: color
2. Extract the image descriptors of the database images
3. Given a query image, extract its descriptor vector
4. Sort the database images according to the similarity with the query image descriptor.

R	G	B
20	30	200
34	166	111
12	220	222
25	244	30

General approach to retrieval problems

Let's suppose for now that the descriptor is simply the mean color.



'Building' or
'Nature'?

Algorithm:

1. Define the image descriptor: color
2. Extract the image descriptors of the database images
3. Given a query image, extract its descriptor vector
4. Sort the database images according to the similarity with the query image descriptor.

R	G	B	Order
20	30	200	1
34	166	111	2
12	220	222	3
25	244	30	4
233	55	211	Query

Do we explicitly extract what does represent the image (building, tree, person)?

Algorithm for image classification

Labeled images (training set)



Building



Building



Nature



Nature

Test image



**'Building' or
'Nature'?**

Training:

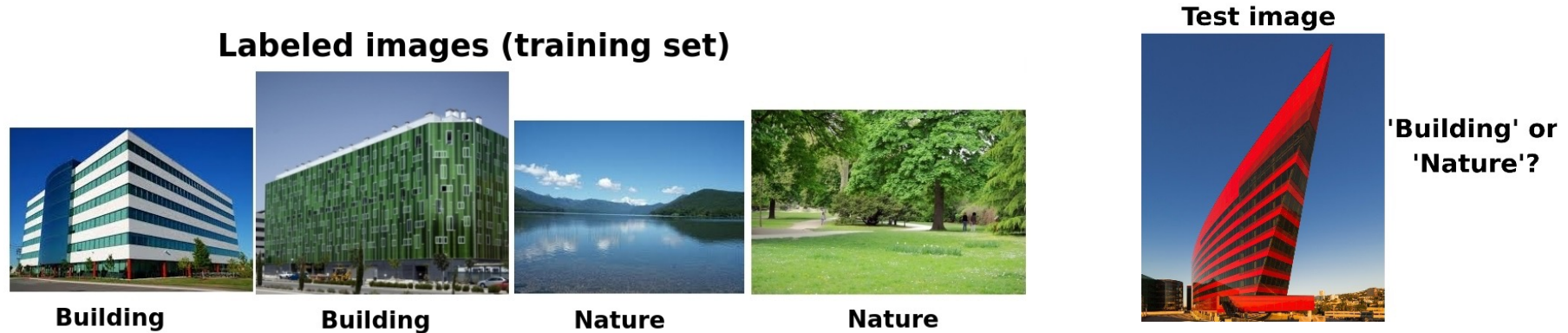
1. Define the image descriptor
2. Represent each image of the training set by its descriptor
3. Store the descriptors and class labels of the training samples (labeled images)
4. Build a model

Test:

4. Given a test example extract its descriptor
5. Apply the model (compare with the training examples to decide its label)

General approach to classification problems

Let's suppose for now that the descriptor is simply the mean color.



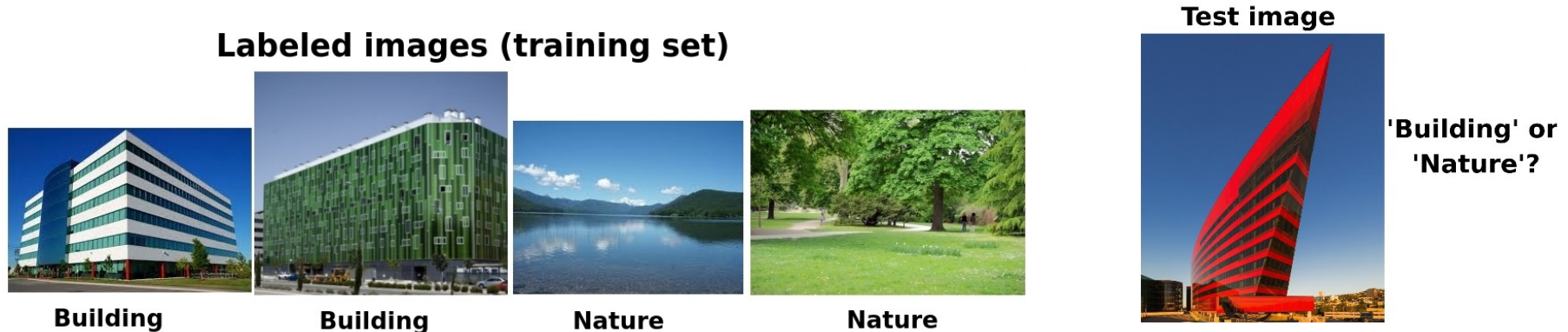
Training phase:

- Represent each image of the training set by its descriptor
- Store the descriptors and class labels of the training samples (labeled images)
- **Model:**
 - if $G > 200$, \rightarrow label 0 (nature).
 - Otherwise, label 1 (building).

R	G	B	Label
20	30	200	1
34	166	111	1
12	220	222	0
25	244	30	0

General approach to classification problems

Let's suppose for now that the descriptor is simply the mean color.



Test phase:

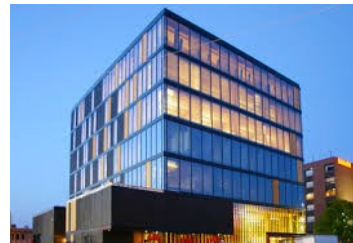
- Compute the descriptor of the test image
- Apply the model to compare the descriptor of the test image to the descriptors of the training images in order to determine its class.

R	G	B	Label
20	30	200	1
34	166	111	1
12	220	222	0
25	244	30	0
233	55	211	?

How to choose the descriptor?

The **descriptor (or feature vector)** should describe the image in a way that is invariant to all the image changes that are suitable to our application:

- Color,
- Illumination,
- Noise,
- Scale



What do have in common all these buildings?

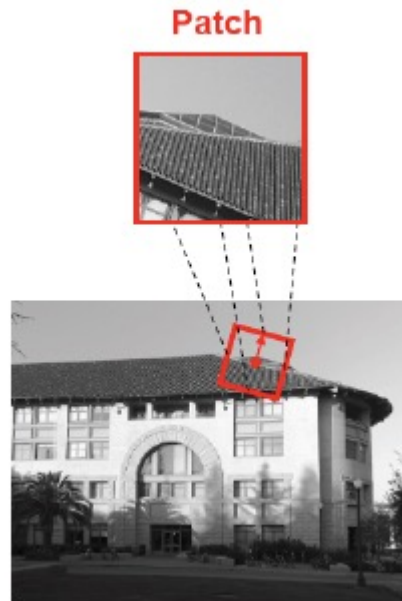
What does distinguish them from images of natural lands?

Outline

- Template Matching
- Image descriptors: what are and why we need them?
- **A particular kind of image descriptor (HOG)**
- Image retrieval
- Pedestrian detection with HOG

How to choose the descriptor?

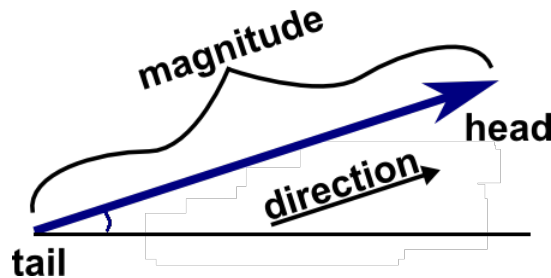
Can we discriminate objects based on their local shape and appearance?



Is the gradient structure characteristic of local or global shape and appearance?

Gradient vector

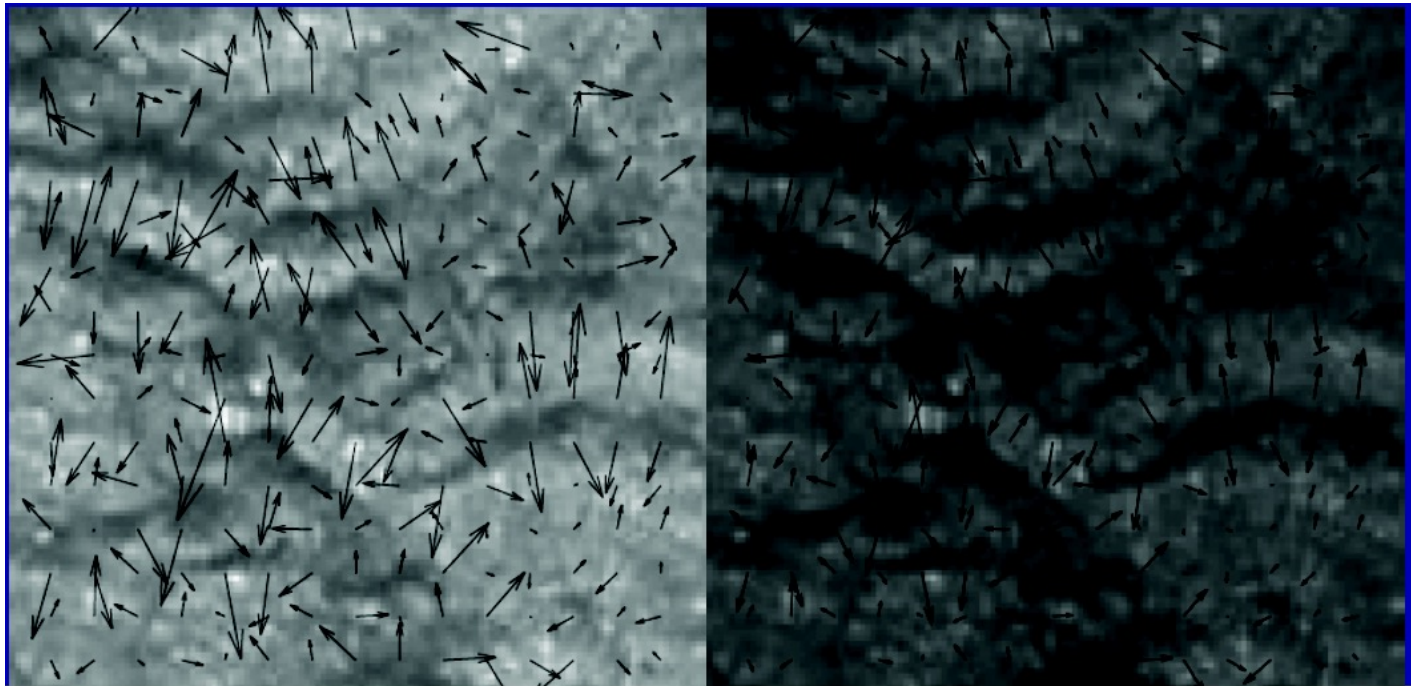
Remember what is the image gradient....



- The image gradient at each pixel is a vector.
- As a vector, it has a magnitude and a direction.

Gradient vector

Would the gradient magnitude be useful?

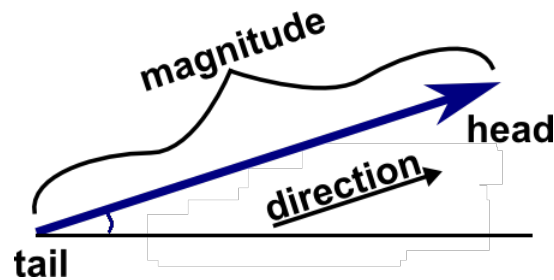


Gradient magnitude is affected by illumination changes!

But the direction is not!

Histogram of gradients

What could be the histogram of gradients?

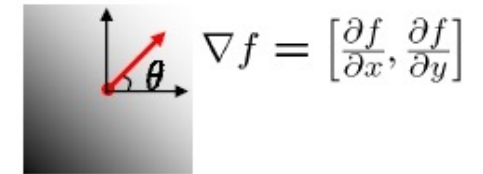
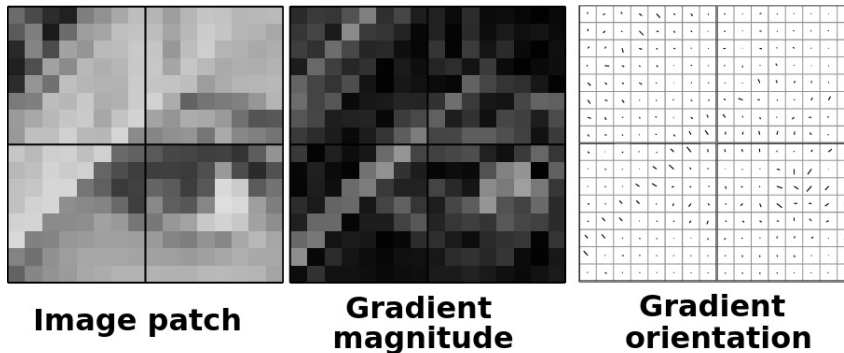


A diagram showing a red vector in a 2D coordinate system. The vector starts at the origin and points into the first quadrant. The angle between the vector and the positive x-axis is labeled θ . To the right of the diagram, the gradient is defined as $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$.

$$\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$$

Histogram of Oriented Gradients (HOG)

Gradient of an image:



$$\theta = \tan^{-1} \left(\frac{\partial f / \partial y}{\partial f / \partial x} \right)$$

How to obtain the overall orientation of the pixels gradients?

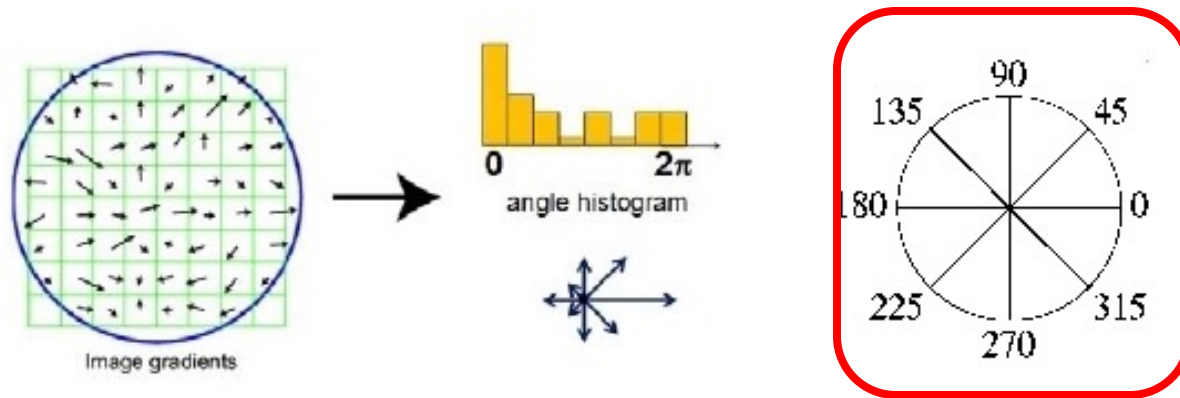
Histogram

Remember what is the histogram of color...



Histogram of Oriented Gradients (HOG)

Histogram of gradient orientations

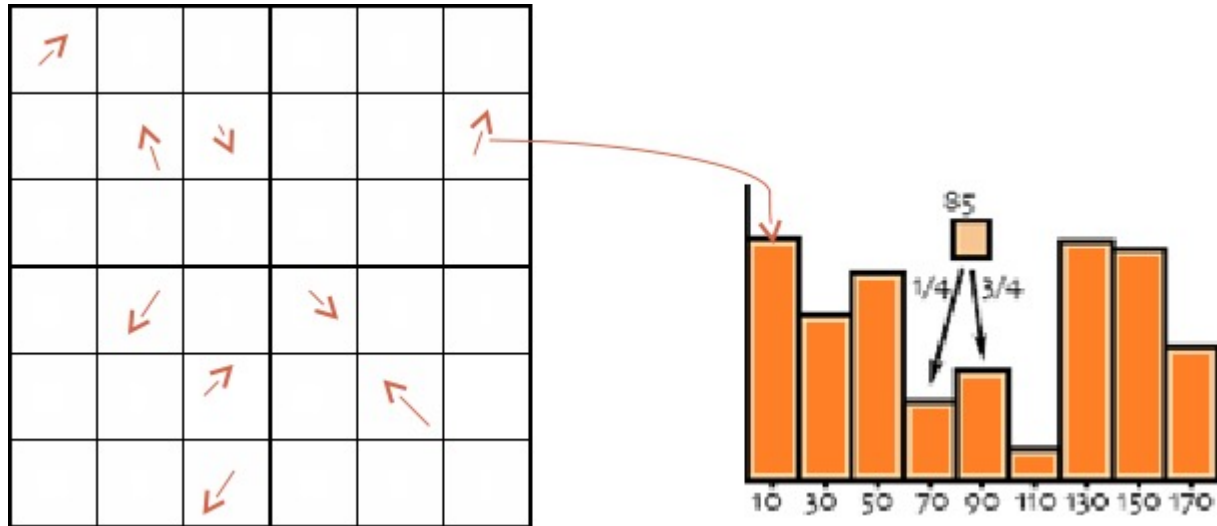


- The gradient orientation is an angle
- Count occurrences of gradient orientation in a patch
- Quantize to 8 bins, each bins cover 45 degrees
- We can use the histogram and a visual representation of the histogram

Histogram of Oriented Gradients (HOG)

Histogram of gradient orientations.

Example:

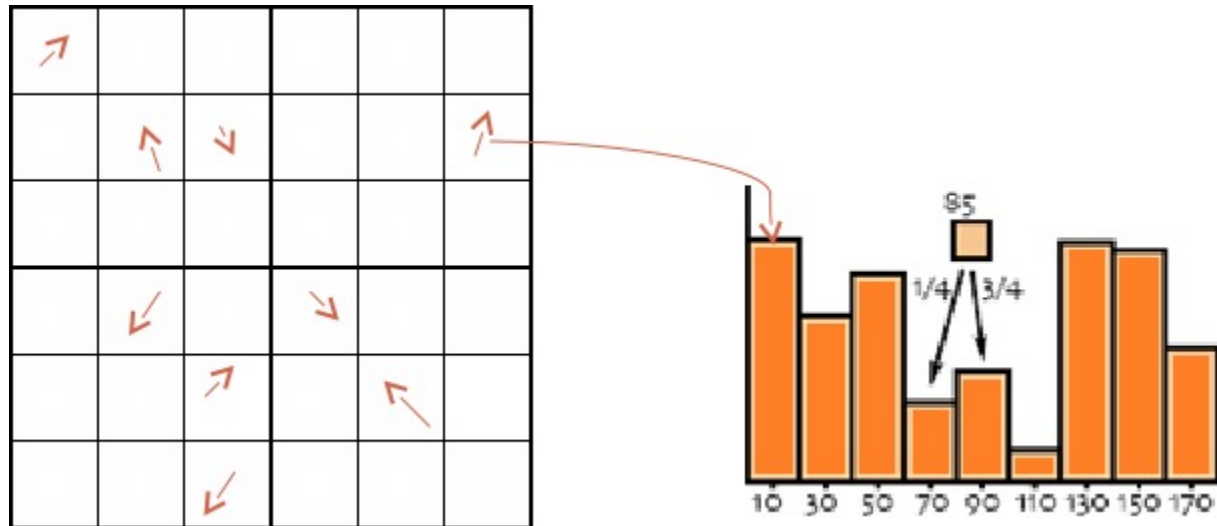


- From 0 to 180 degrees, 9 bins, 20 degrees per bin
- $\theta = 85$ degrees
- To which bin it contributes?

Histogram of Oriented Gradients (HOG)

Histogram of gradient orientations

Example:

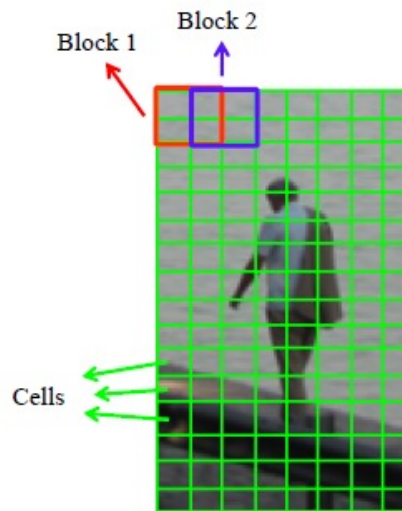


- Compute the distance to adjacent bin centers (from Bin 70 and Bin 90 are 15 and 5 degrees, respectively).
- Divide the distance by the size of the bins (distance/binsize): $5/20=1/4$, $15/20=3/4$
- Weight the contribution by the gradient magnitude

Histogram of Oriented Gradients (HOG)

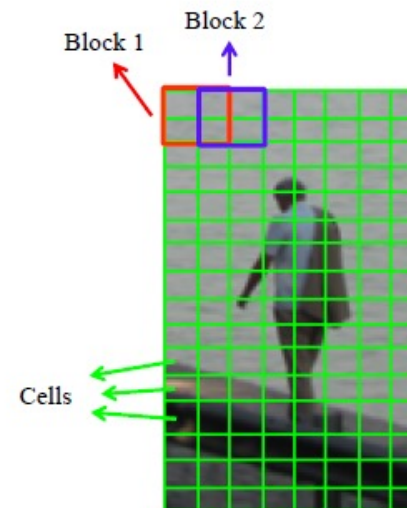
To compute the image HOG descriptor/feature:

- Divide the image into small connected regions called **cells**.
- Compute a local histogram for each cell weighted by gradient magnitude.
- Simply concatenate the histogram of all the cells.



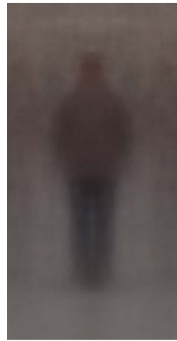
HOG: Contrast - normalization

- Gradient strengths vary over a wide range owing to local variations in illumination and foreground-background contrast.
- **How to achieve invariance to changes in illumination or shadowing?**
- Compute a measure of intensity across a larger region than a cell (a block)
- Normalize all cells within the block with this intensity value
- L_2 normalization:
$$L_2 = \sqrt{||v||_2^2 + \epsilon^2},$$
$$\epsilon \text{ is a regularization parameter.}$$

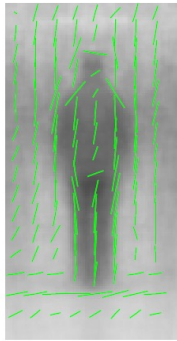


Histogram of Oriented Gradients (HOG)

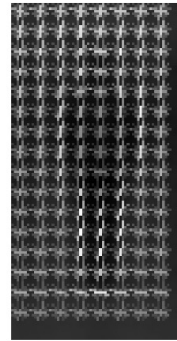
- Graphical representation:



Averaged
examples



Predominant
direction



Histograms of
gradients

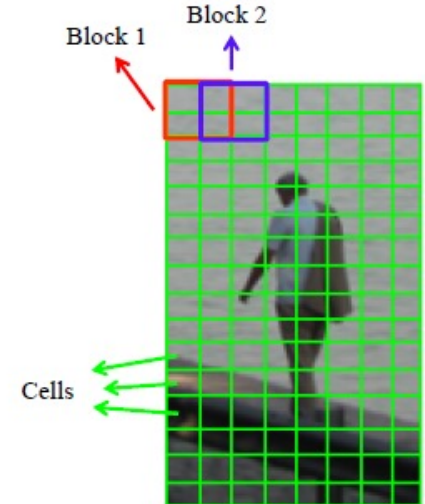
Histogram of Oriented Gradients (HOG)

Example:

- For a 64x128 image
- Divide the image in cells of 8x8 pixels (8x16 cells)
- Group cells into blocks of 2x2 cells (16x16 pixels) of 50% overlap

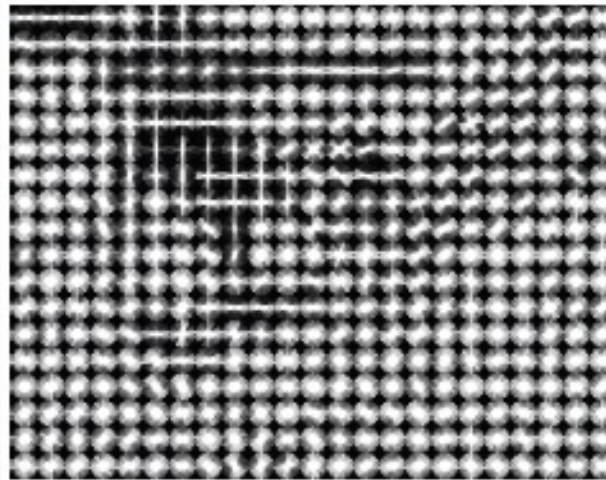
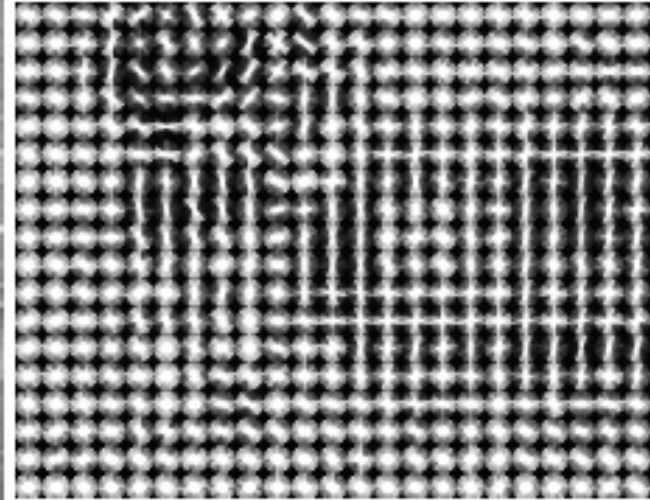
→ Total number of blocks: $7 \times 15 = 105$

- Quantize the gradient orientation into 9 bins
- Concatenate histograms to obtain the feature vector of size: $105 \times 4 \times 9 = 3780$ (each block has 4 histograms).



Histogram of Oriented Gradients (HOG)

Can we say that the HOG is able to describe local shape and appearance?



Does the HOG descriptor carry information about the gradient or edge positions?

Histogram of Oriented Gradients (HOG)

- The essential thought 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 HOG descriptor is thus particularly suited for human detection in images

Compute gradient in practice

- Convolve the image with discrete derivative mask:
 - $D_x = [-1, 0, 1]$, $D_y = [1, 0, -1]^T$
- Angles: $\tan^{-1} (D_y/D_x)$
- Magnitude: $\sqrt{D_y^2 + D_x^2}$

Outline

- Template Matching
- Image descriptors: what are and why we need them?
- A particular kind of image descriptor (HOG)
- **Image retrieval**
- Pedestrian detection with HOG

Image retrieval

Given an image (query), find all similar images in the database.



K-Nearest Neighbors for retrieval

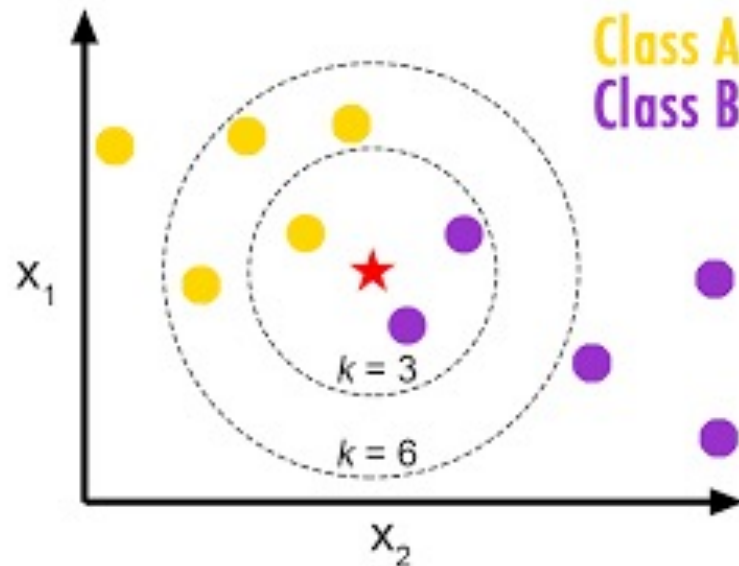
- The feature vector of an image is a point in our feature space (d-dimensional space, where d is the number of features)
- The query is an unlabeled vector in our feature space
- Retrieve the k-closest neighbors as the **relevant items** to a query
 - k is a user-defined parameter

K-Nearest Neighbors for retrieval

- The feature vector of an image is a point in our feature space (d-dimensional space, where d is the number of features)
- The query is an unlabeled vector in our feature space
- Retrieve the k-closest neighbors as the **relevant items** to a query
 - k is a user-defined parameter
 - For retrieval: Database images do not necessarily have labels.
- Sort based on distance/similarity:
 $d(\text{template}, \text{image}) = d(\text{HOG}(\text{template}), \text{HOG}(\text{image}))$

K-Nearest Neighbors for classification

- The image is classified by assigning the label which is most frequent among the k training samples nearest to the test point.
- k is a user-defined parameter (How to choose k ?)
- Can the classification change for different k ?



Outline

- Template Matching
- Image descriptors: what are and why we need them?
- A particular kind of image descriptor (HOG)
- Image retrieval
- **Pedestrian detection with HOG**

Pedestrian detection



Pedestrian detection

Why is the problem difficult?

- Wide variety of articulated poses
- Variable appearance/clothing
- Complex backgrounds
- Unconstrained illumination
- Occlusions
- Different scales

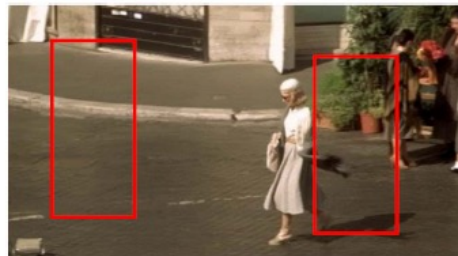


Pedestrian detection

- Transform the detection problem into a binary (yes/not) classification problem:
 - Use sliding window strategy
 - Is a given window representing a pedestrian or not?
 - Positive data – 1208 positive window examples

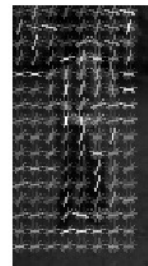
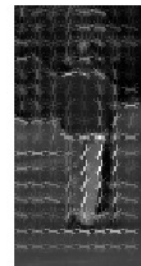
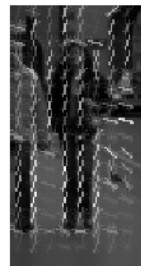
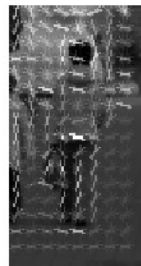
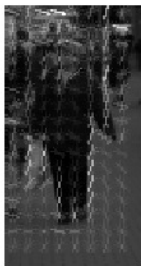
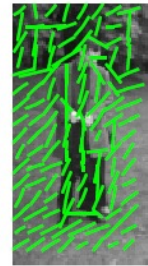


- Negative data – 1218 negative window examples (initially)

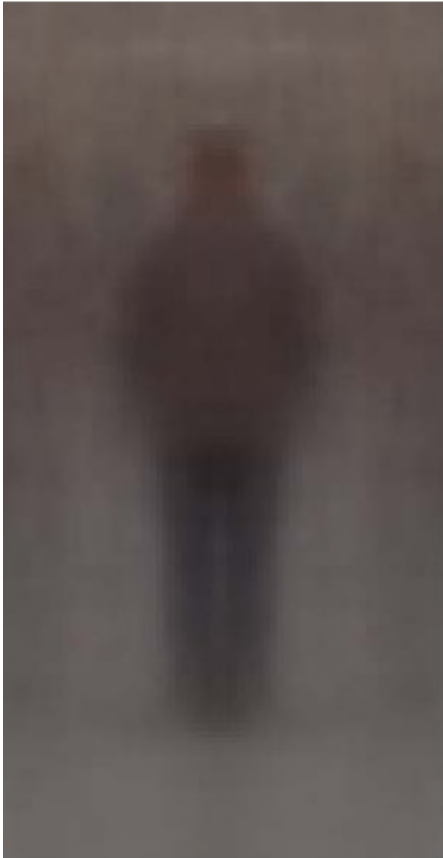


Pedestrian detection

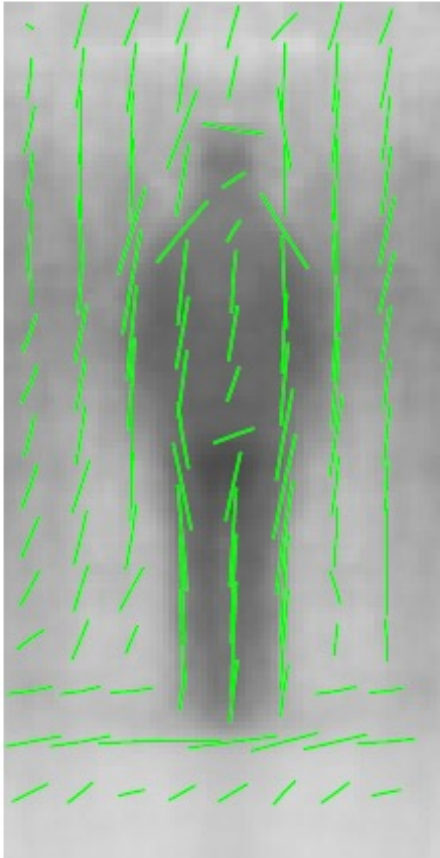
Compute HOG descriptor for all training samples



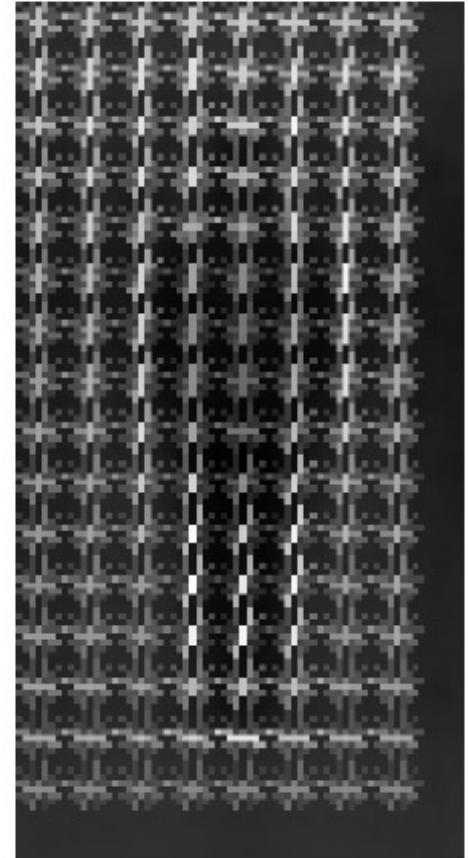
Pedestrian detection



Averaged positive examples

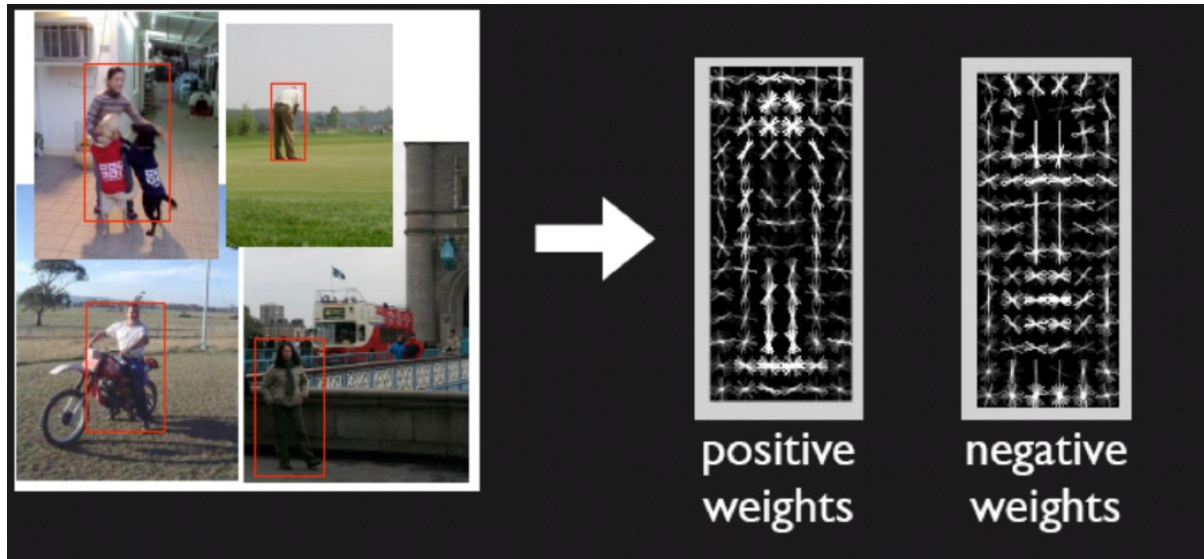


Predominant direction



Histograms of gradients

Apply correlation with a pedestrian template



Slide from Deva Ramanan

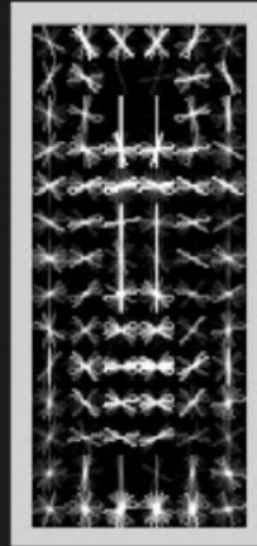
- Positive weights show edge orientations highly correlated with the pedestrians images.
- Negative weights show edge orientations non correlated with image regions containing a pedestrian (horizontal edges in the region of the legs)

Meaning of negative weights

pedestrian
model



>



pedestrian
background
model

Pedestrian detection based on HOG

Descriptor extraction:

- (a) Get the average gradient image over query and test examples
- (b) Extract HOGs for the query image,

HOG-based Retrieval:

- (c) For each dataset image, extract the region bottom left and compare to the query HOG.
- (d) Apply the sliding window technique all over the image and compute the HOGs distances
- (e) Compute the maximum over the distances.



Pedestrian detection results



Pedestrian detection: Common pipeline

- Extract features which are discriminant for your problem
 - Each image is represented by a high-dimensional feature vector.
- Use a Machine Learning algorithm
 - This always requires to first extract features from all images of the training set.
- Make predictions (classification) on the test images to detect the looked for model/object.
 - K-nn
 - Other classifiers

References

- Source for slides are from:
 - K. Grauman and B. Leibe and J. Tompkin,
 - James Hays, Derek Hoiem
 - Navneet Dalal and Bill Triggs
 - Petia Radeva

Readings

- Szeliski, Ch 4.1
- (optional) K. Mikolajczyk, C. Schmid, A performance evaluation of local descriptors. In PAMI 27(10):1615-1630
 - http://www.robots.ox.ac.uk/~vgg/research/affine/det_eval_files/mikolajczyk

Lecture videos

- Template Matching
 - From 128 - Intro. Until 142 - End.
 - Except 130, 131, 132, 134, 135

Total time: 9,82 minutes.

From *Introduction to Computer Vision*:

<https://www.udacity.com/course/introduction-to-computer-vision--ud810>

Final Test

- **Go to:** *Socrative.com*
 - Choose Student Login
 - Room Name: COMPUTERVISION
 - Enter your name: It can be anonymous.

COMPUTATIONAL VISION: Image Features (HOGs)

Master in Artificial Intelligence

Department of Mathematics and Computer Science



UNIVERSITAT DE
BARCELONA