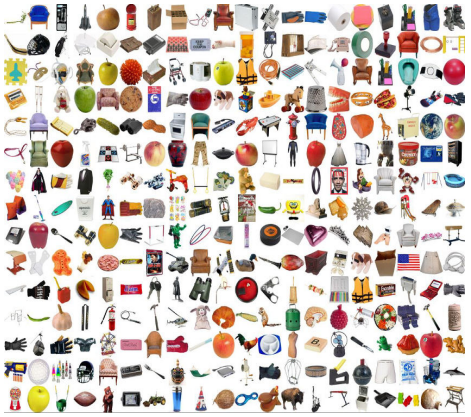
# Bag-of-Words.
# Bag-of-Visual-Words.

Radu Ionescu, Prof. PhD.
raducu.ionescu@gmail.com

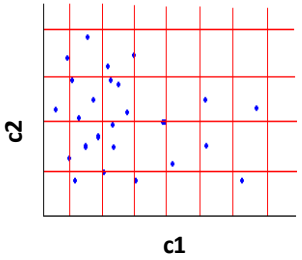
Faculty of Mathematics and Computer Science
University of Bucharest

# Agenda

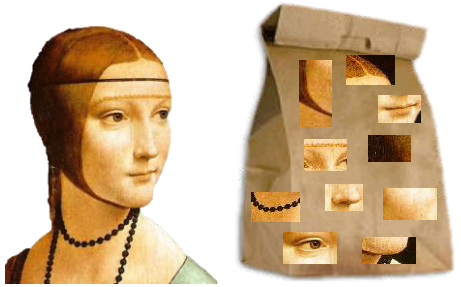
- Object class recognition
  - General picture
  - What is an object class?



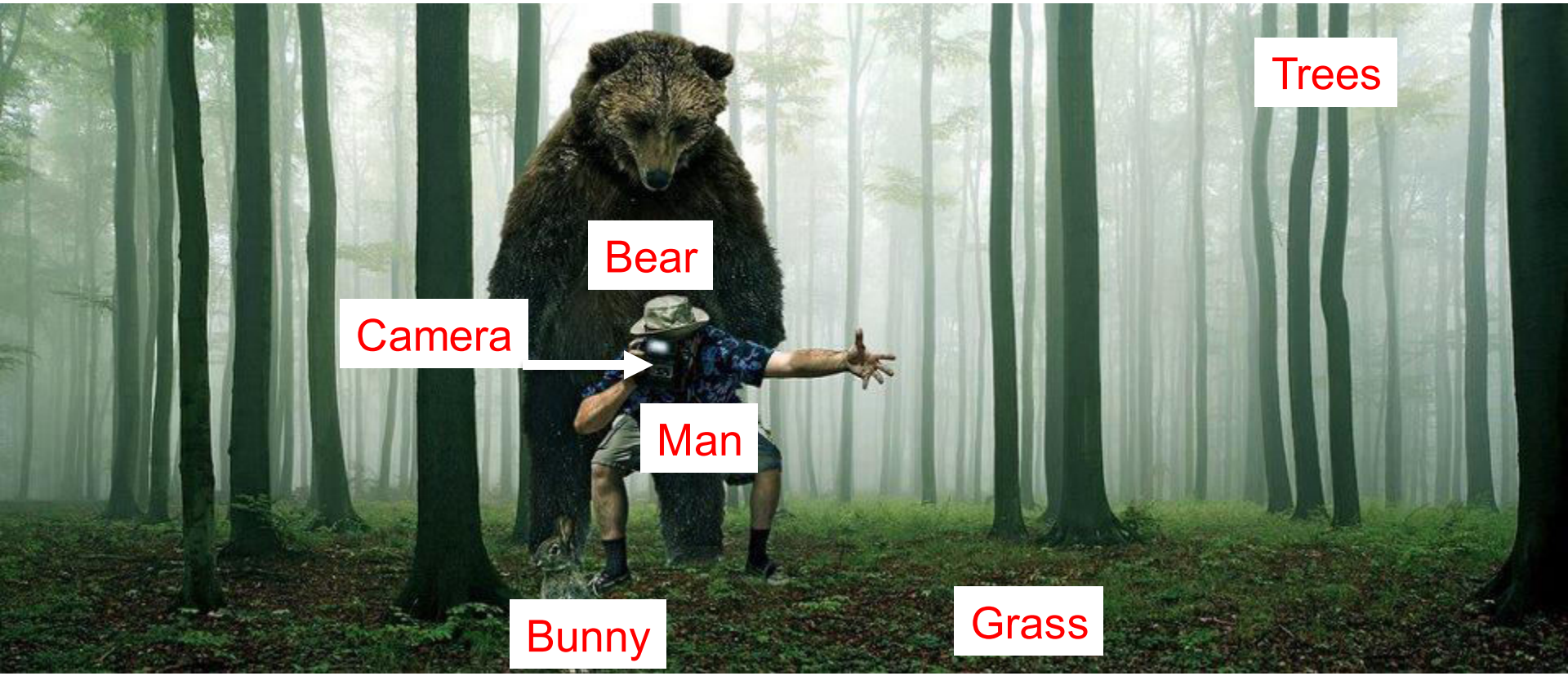
- Text classification
  - Bag-of-Words




- Image classification
  - Histograms for feature representation
  - Bag-of-Visual-Words

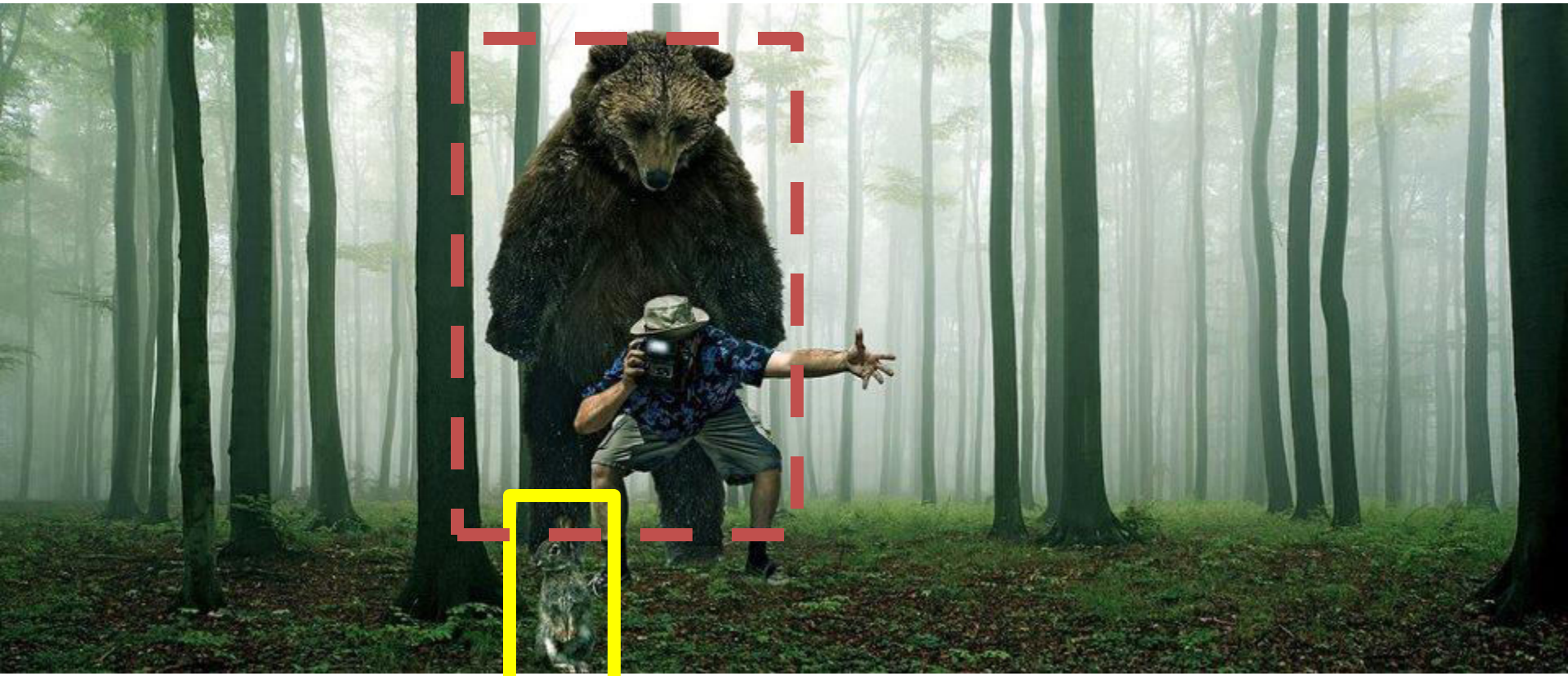


# What do you see?




Forest

Slide adapted after Derek Hoiem



Is it **dangerous**?

How **fast** does it run?

Is it **alive**?

Has **tail**?

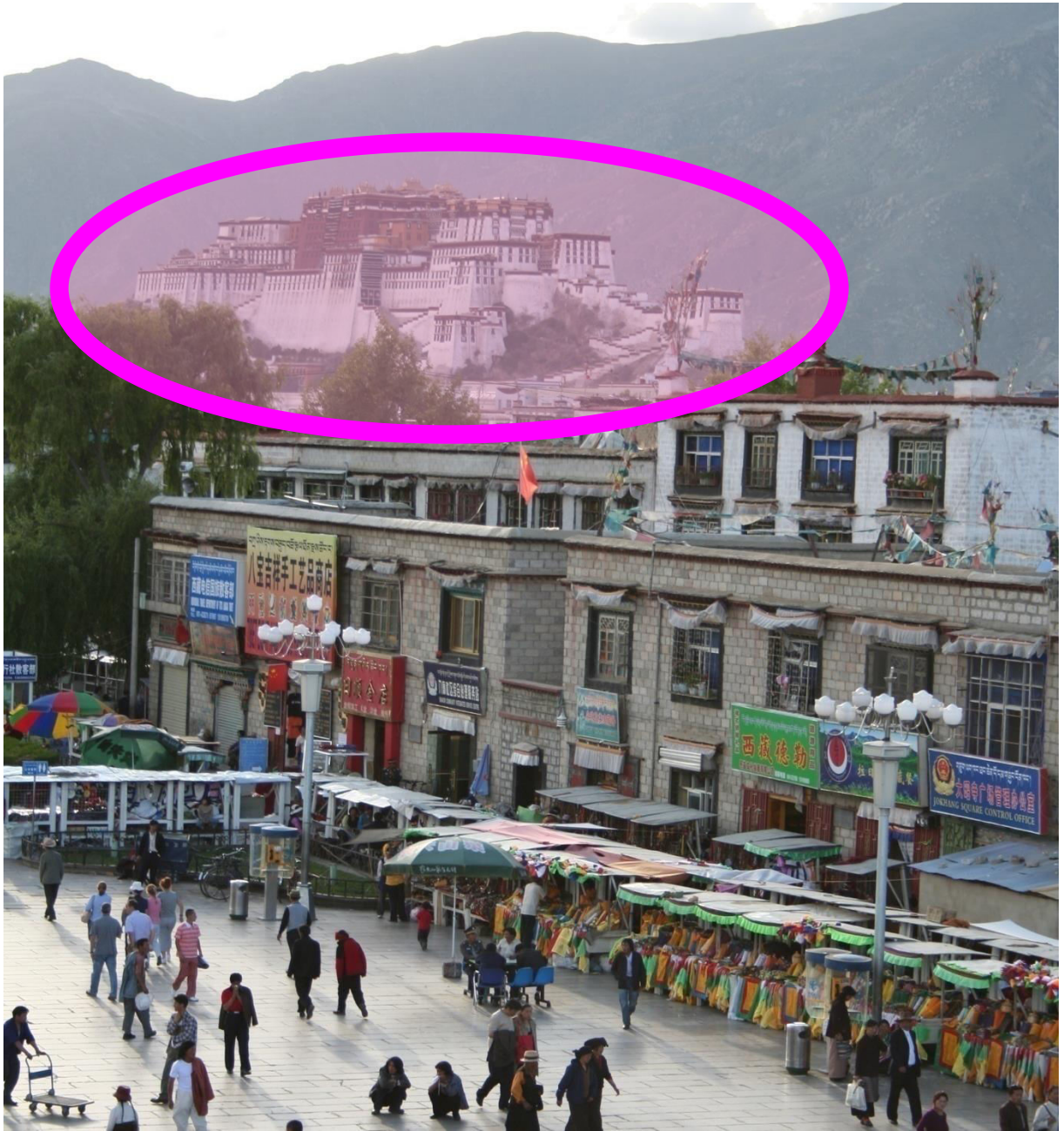
Slide adapted after Derek Hoiem

# Object class recognition

Verification: Is there a light pole? Binary answer: Yes/No

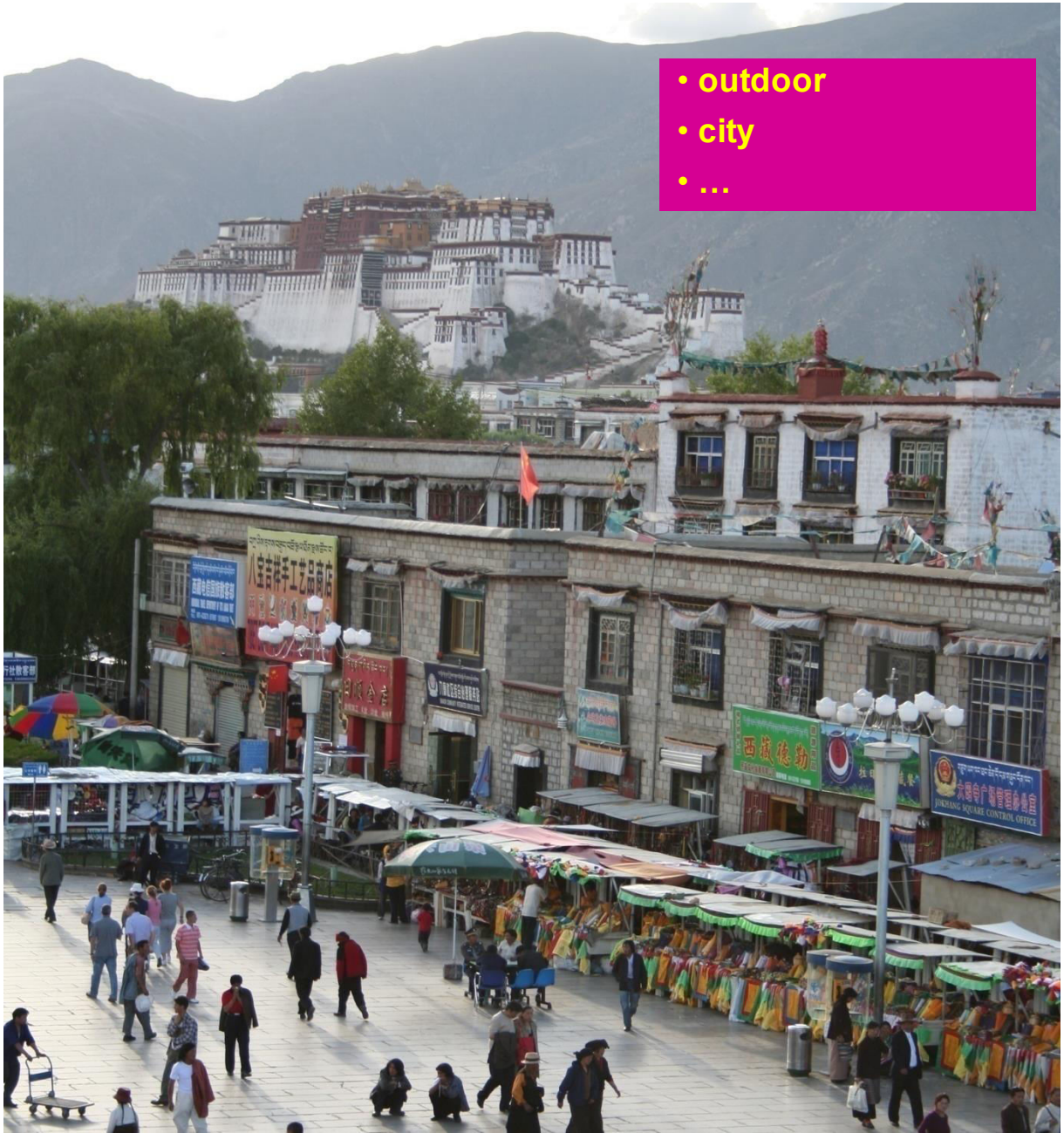
# Detection: Where is the pole? Localization

Identification: Which palace? Instantiation: Potala Palace of Tibet



# Region classification

# Scene classification

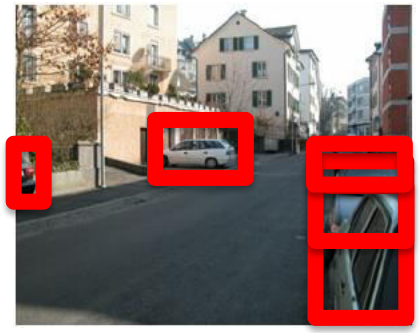


# Instance-level Object Recognition


Slide adapted after K. Grauman

# Generic Object Recognition




Recognizing all instances of cars

Slide adapted after K. Grauman

# Generic Object Recognition

- Problem formulation:
  - "Given a small number of images with an object class, recognize instances of the object class by assigning the correct object class"

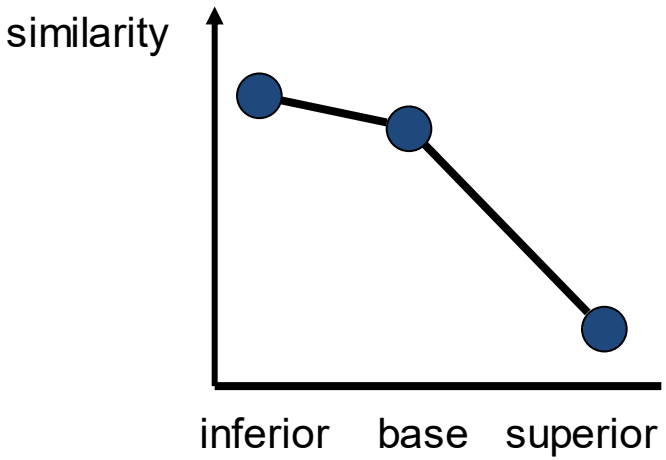
- For which object class you can immediately assign an image?

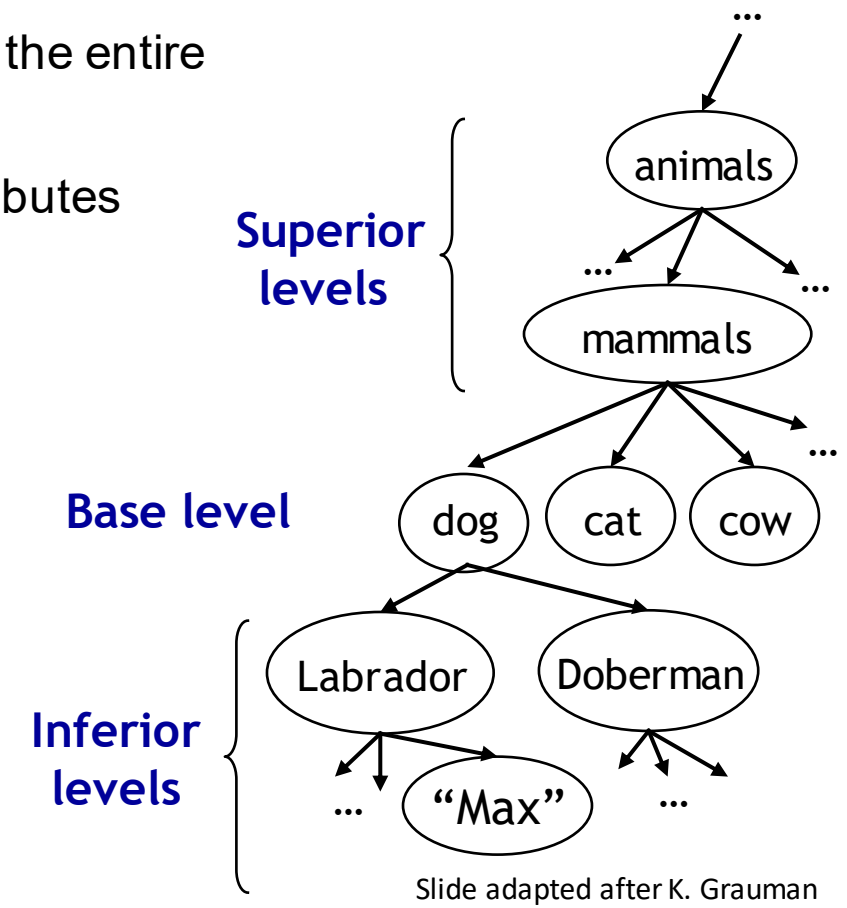

Slide adapted after K. Grauman

# Hierarchical organization of object classes (Rosch – 1976)

Base level:

- Base-level classes are the highest-level classes in which instances are perceived with similar shape
- The highest level for which we can represent the entire class with a single mental image
- There is a significant number of common attributes between member pairs
- First level understood by children

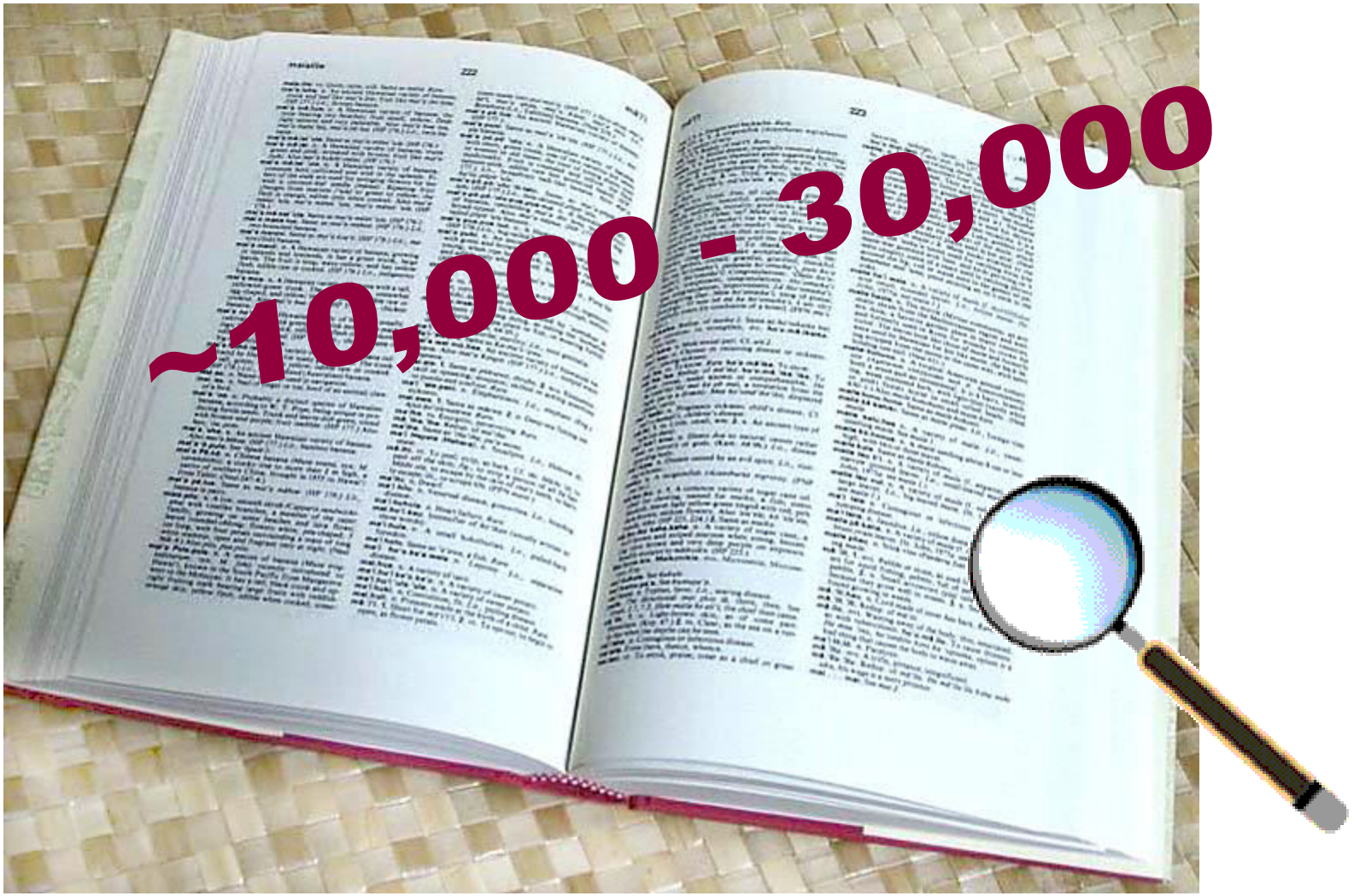



Slide adapted after K. Grauman

# Examples of object classes

# How many object classes?




[Biederman, 1987]

Slide adapted after A. Torralba

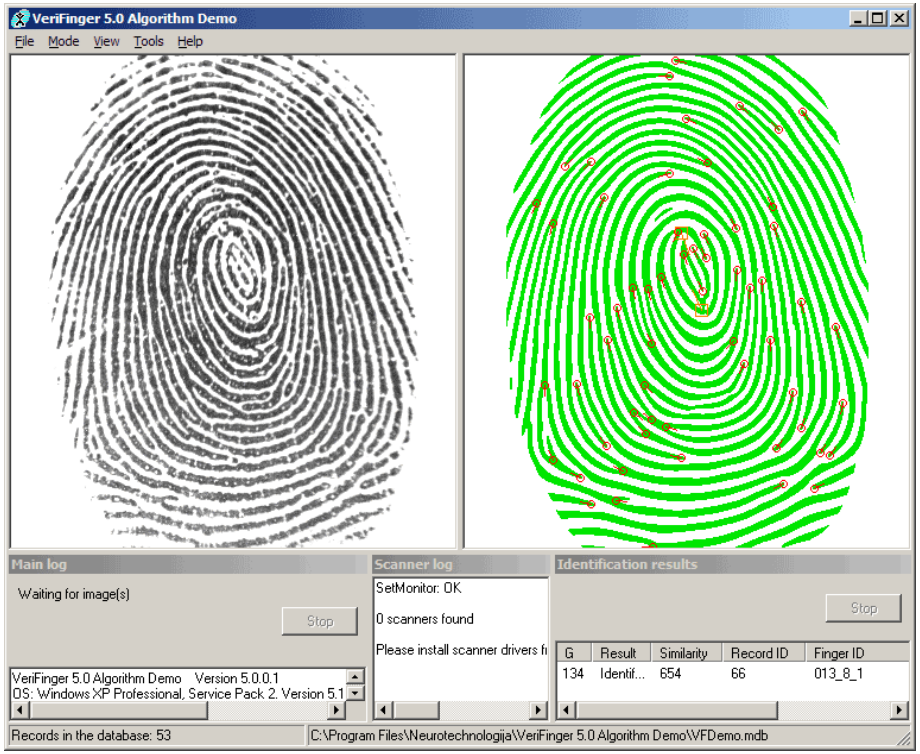
# Object class recognition in computer vision – what works well?

- OCR – license plate recognition, postal codes, etc.

Slide adapted after S. Lazebnik

# Object class recognition in computer vision – what works well?

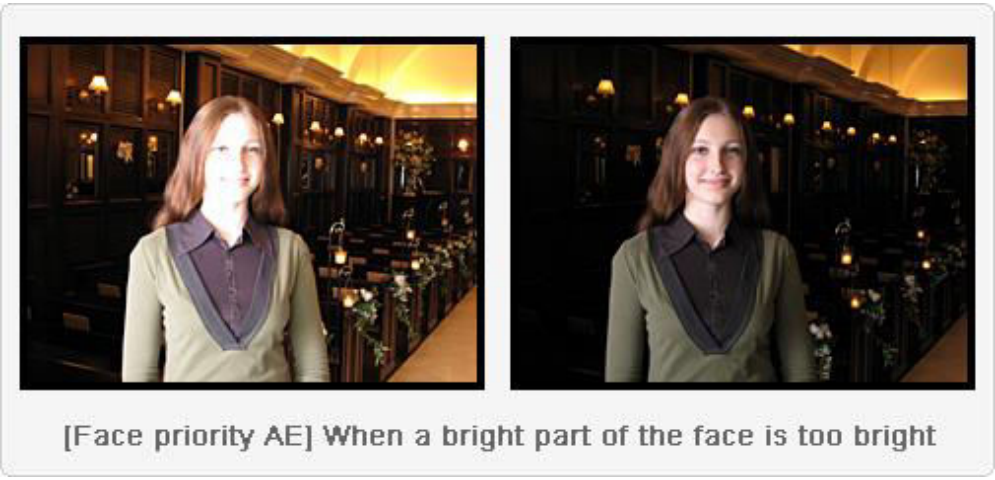
- OCR – license plate recognition, postal codes, etc.
- Fingerprint recognition




Slide adapted after S. Lazebnik

# Object class recognition in computer vision – what works well?

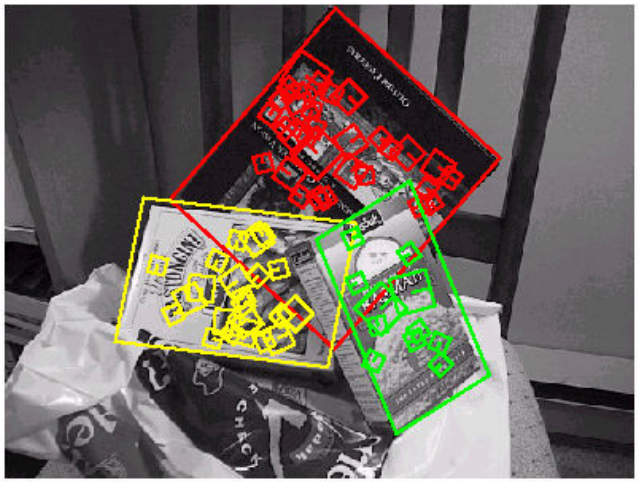
- OCR – license plate recognition, postal codes, etc.
- Fingerprint recognition
- Face detection


Slide adapted after S. Lazebnik

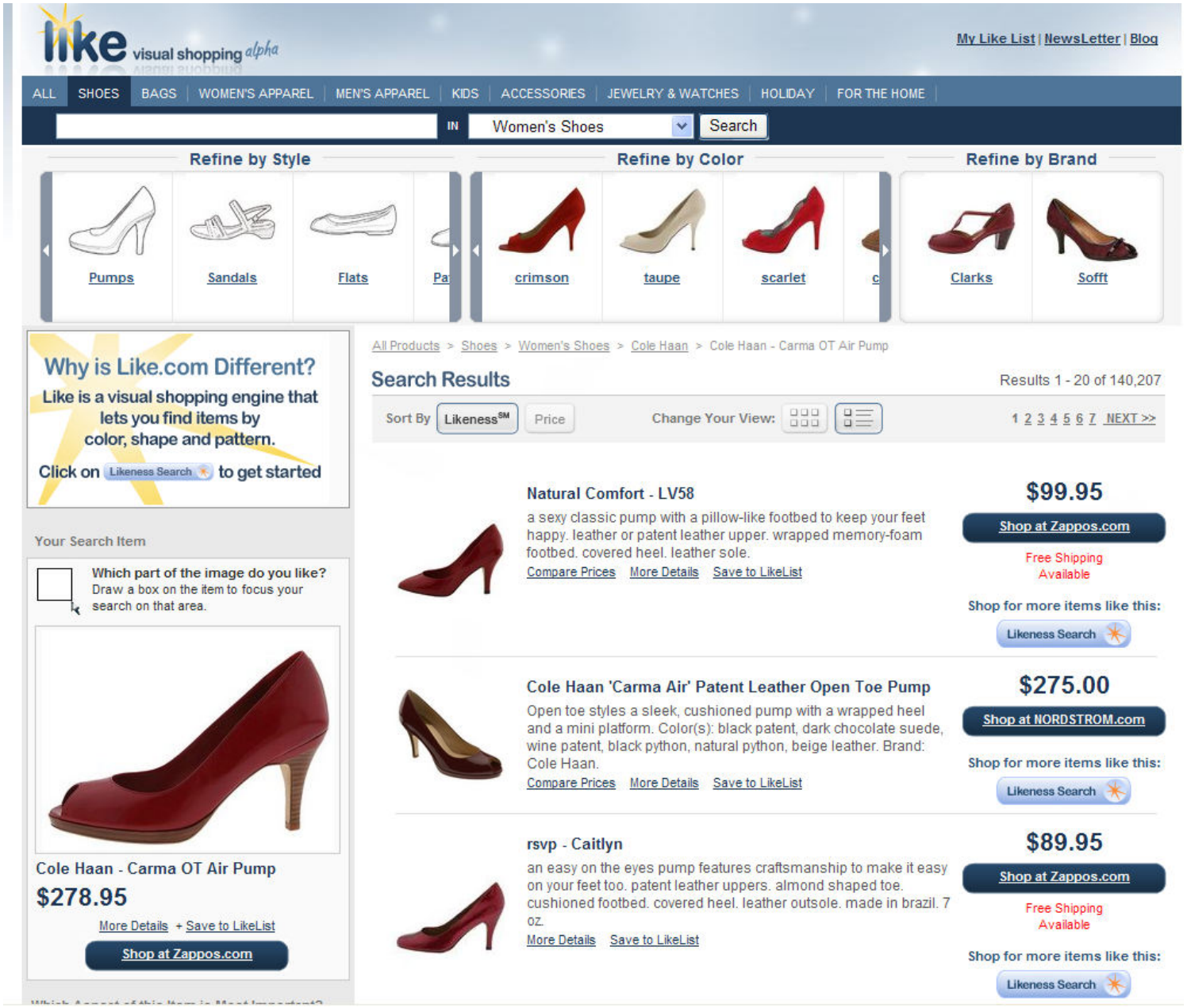
# Object class recognition in computer vision – what works well?

- OCR – license plate recognition, postal codes, etc.
- Fingerprint recognition
- Face detection
- Textured object recognition (CD / book covers)



Slide adapted after S. Lazebnik

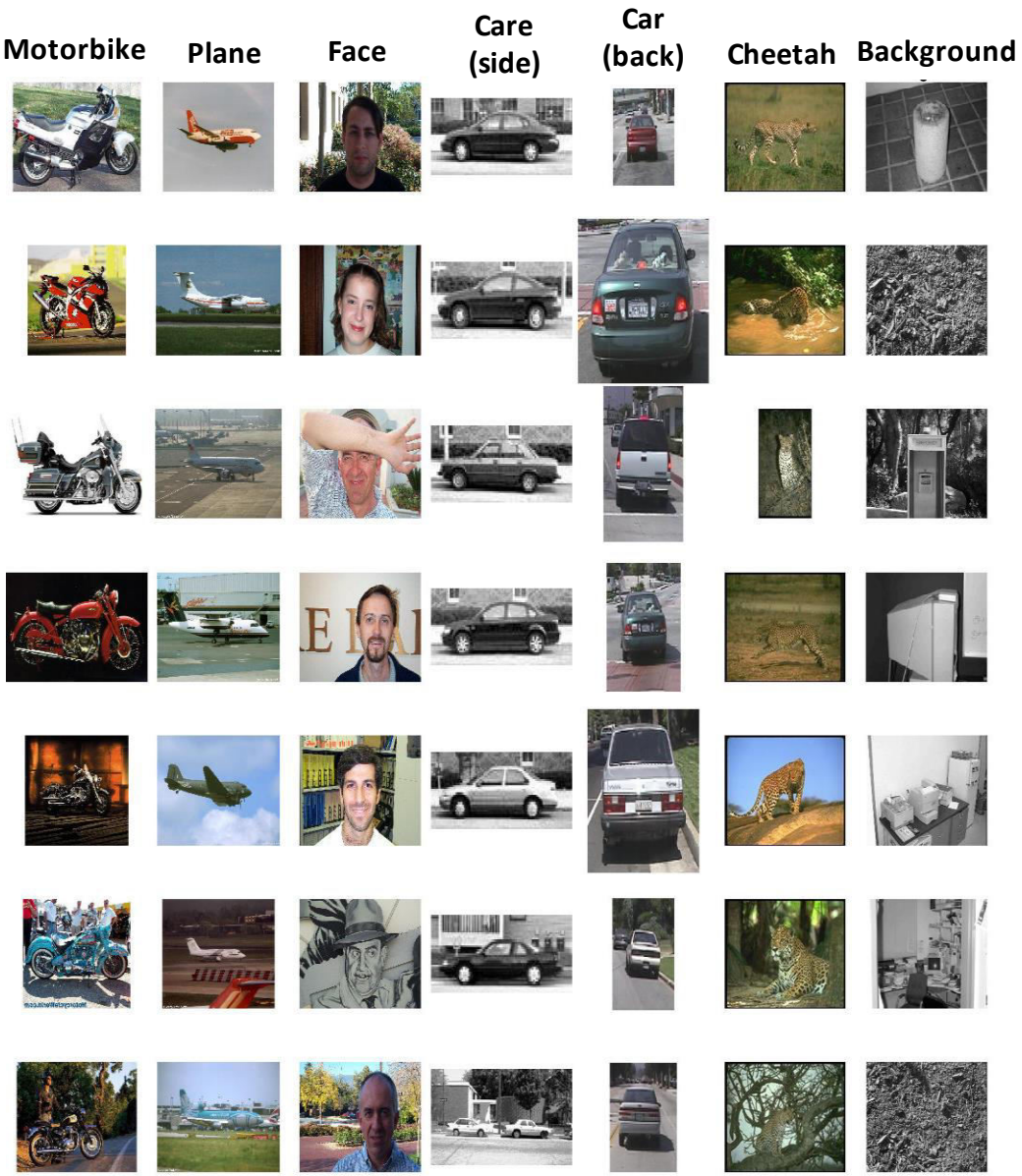
# Finding visually-similar objects



# Object recognition in computer vision

Problems:

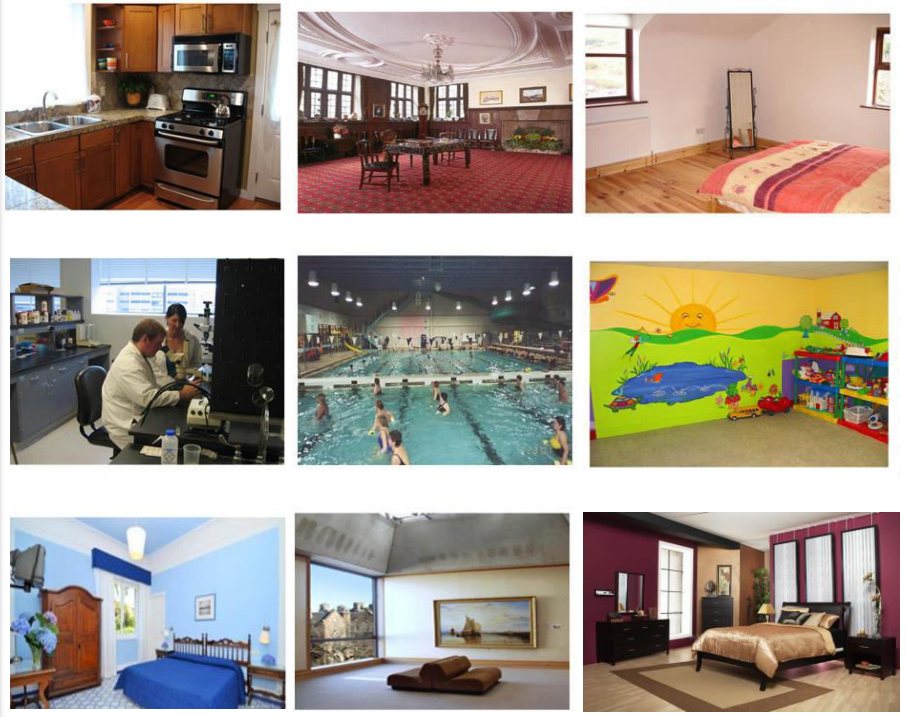
- Intra-class variability
- Inter-class similarity (car, truck, bus)
- Variable object sizes
- Background confusion



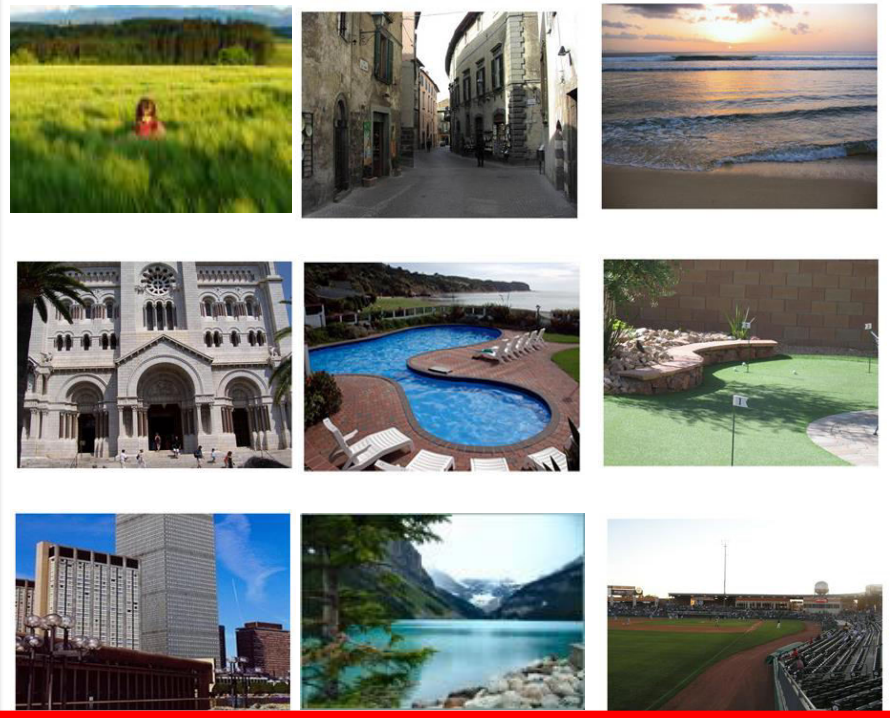
# Image classification

# Example: 2 classes - indoor/outdoor

## Indoor – examples



## Outdoor - examples



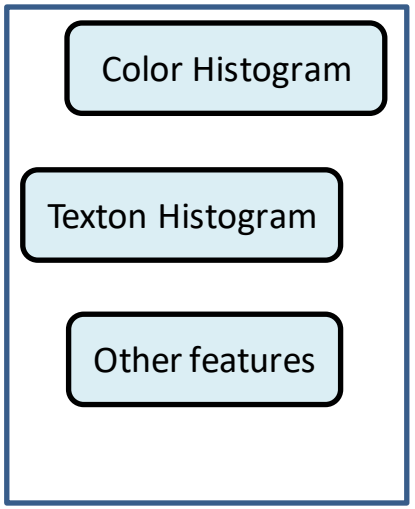
**Classification = assign labels (indoor, outdoor) to new examples**

Test images:

# Supervised Learning

**Examples:**
- **negative**
- **positive**

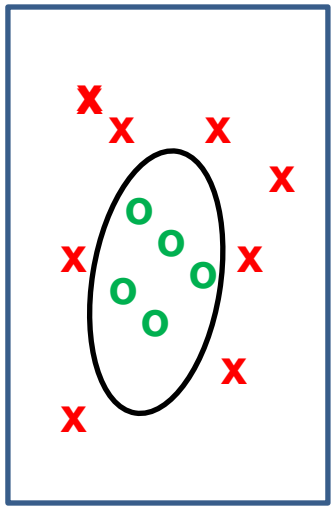
+




Features for image representation

**=** visual descriptors of image content

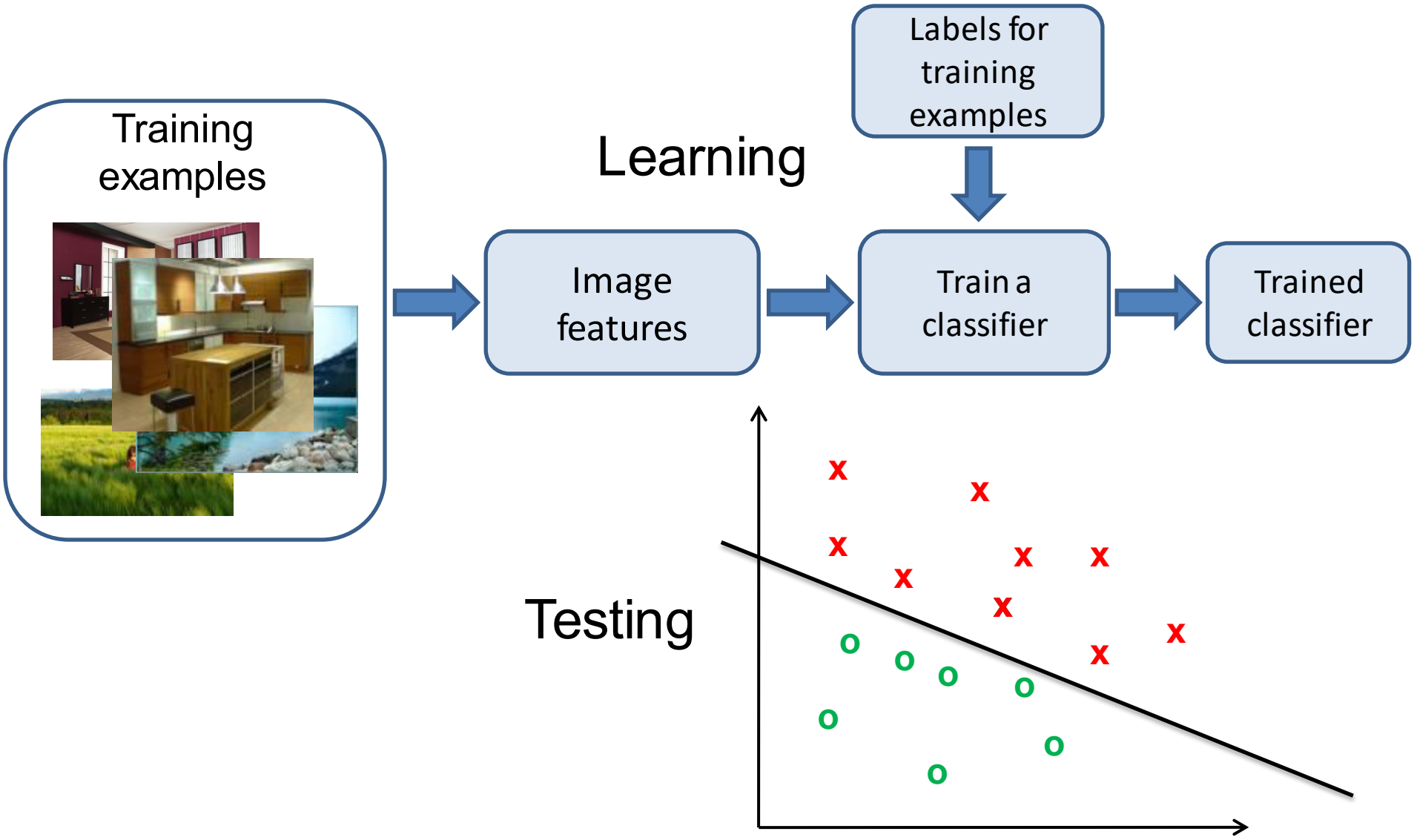
+



Classifier

**=** class label

# Training stage



# Inference stage

## Learning

Training examples

Labels for training examples

Image features

Train a classifier

Trained classifier

## Inference

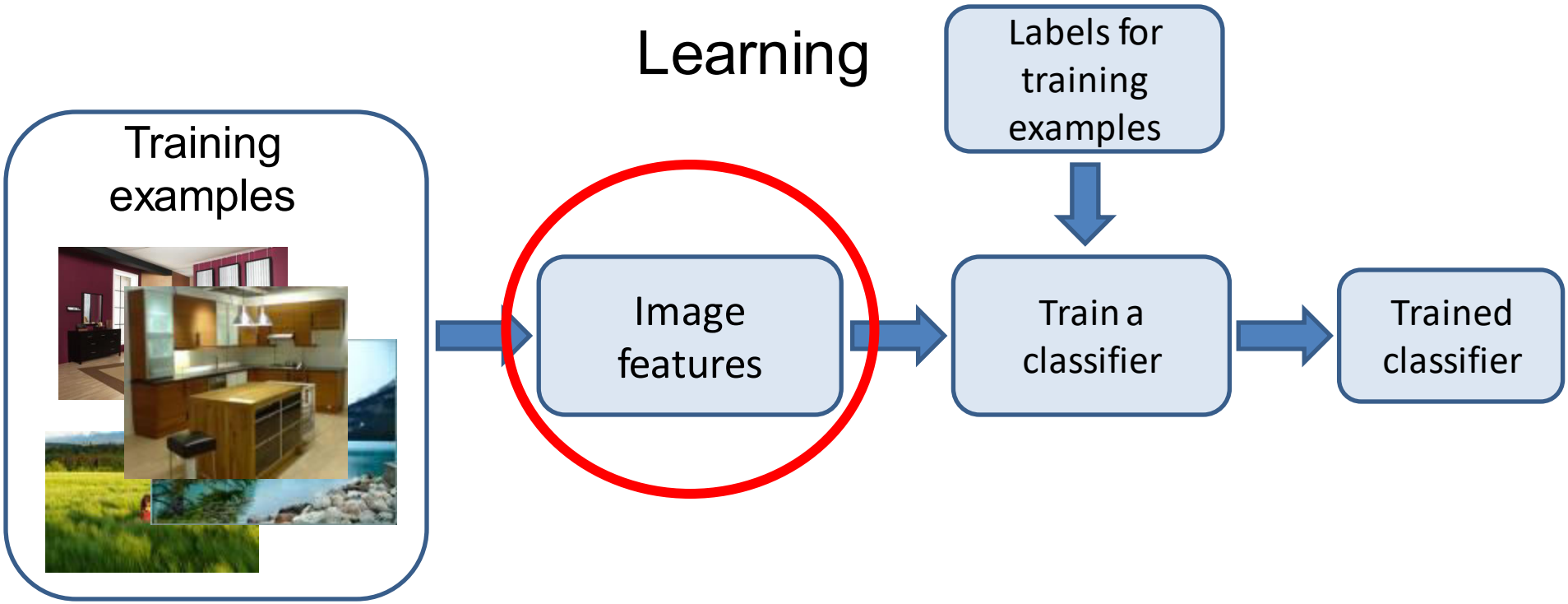
Test image

Image features

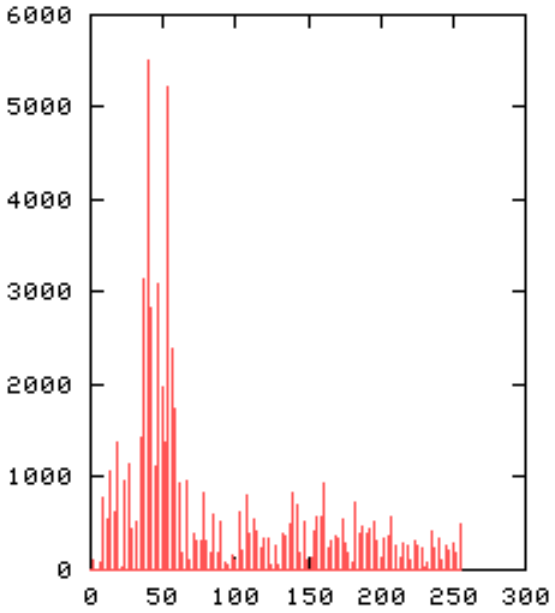
Trained classifier

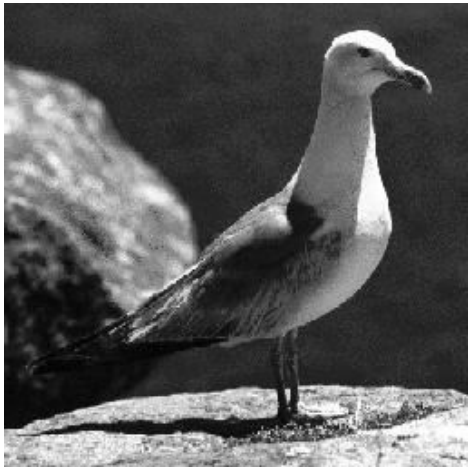
Prediction:
**Outdoor**

# Feature extraction



# Histograms

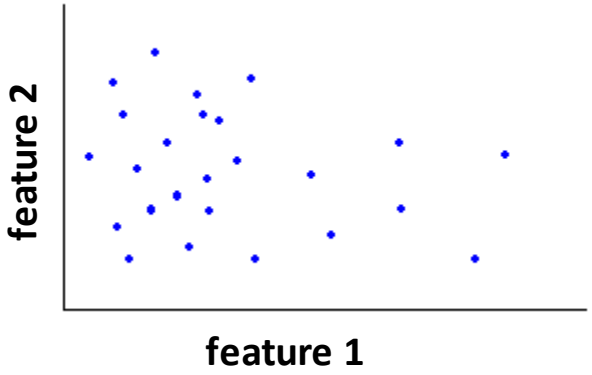


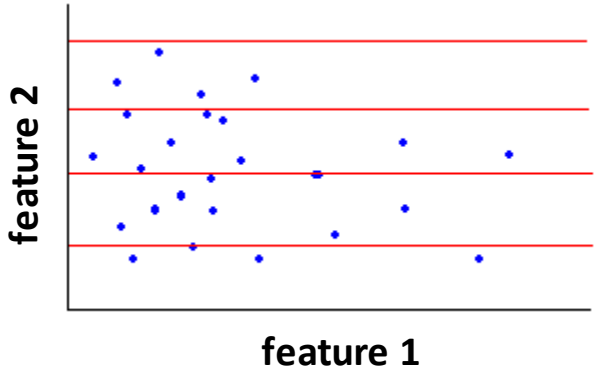



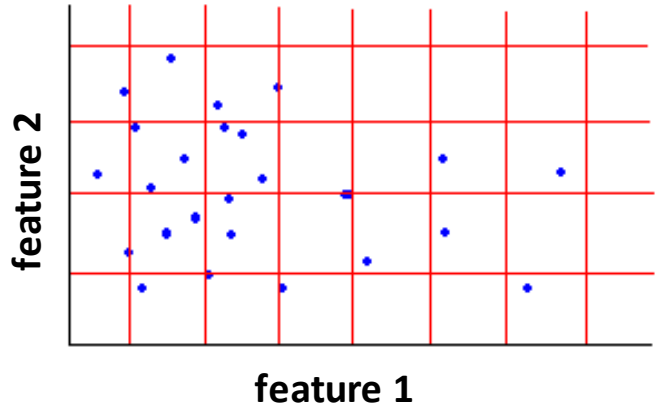
- Measure feature distribution:
  - count how many points “fall” in each bin
  - normalization? sum = 1
  - color, texture, etc…

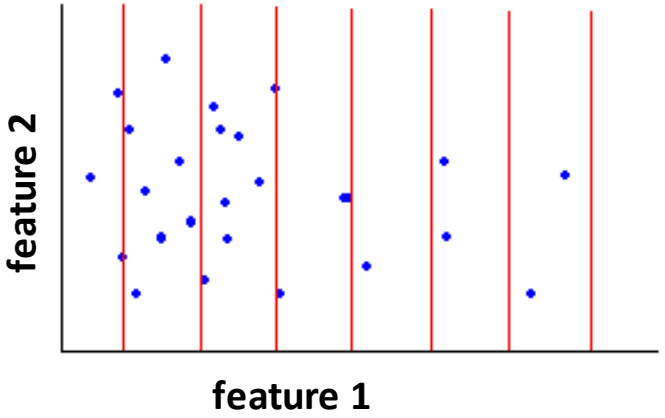
Slide adapted after D. Kauchak

# Histograms



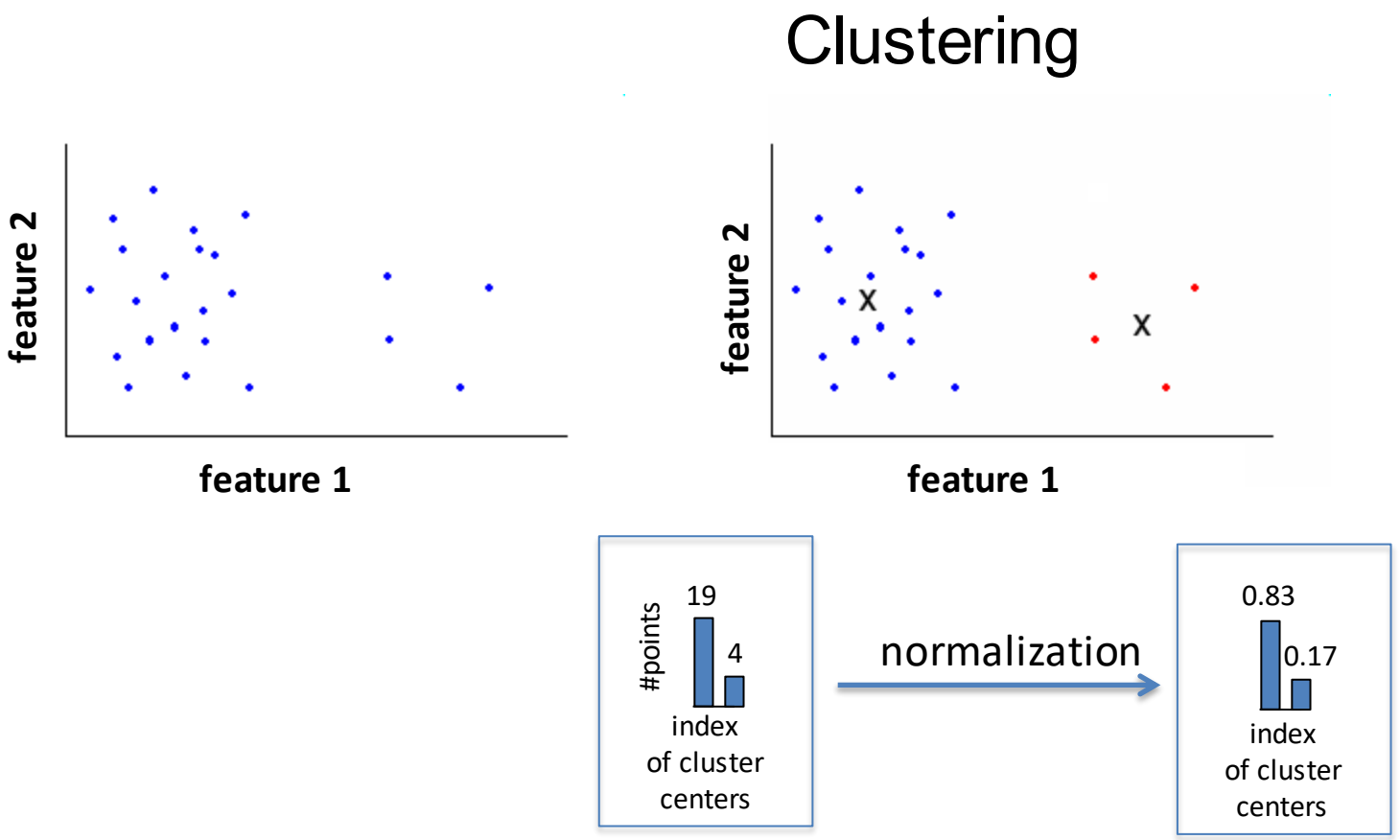




- Joint Histogram
  - Needs a lot of data for a good estimation of the distribution
  - #intervals 1 * #intervals 2

- Independent histogram
  - Needs independent features
  - More data points per interval than joint histogram

# Histograms




- Use the same cluster centroids in all images!

Slide adapted after D. Kauchak

# Computing distance between histograms

- Euclidean distance ($L_2$):

$$d^2(h_i, h_j) = \sum_{m=1}^{K} (h_i(m) - h_j(m))^2$$

- Chi-square distance:

$$\chi^2(h_i, h_j) = \frac{1}{2} \sum_{m=1}^{K} \frac{(h_i(m) - h_j(m))^2}{h_i(m) + h_j(m)}$$

- Histogram intersection distance:

$$histint(h_i, h_j) = 1 - \sum_{m=1}^{K} min(h_i(m), h_j(m))$$

Nearest color histograms (based on Chi-square distance)

# Histograms: implementation details

- Data grouping
  - Intervals: fast, applicable for small images
  - Clustering: slow, but applicable for large images

Few Intervals
Few points for estimation
Coarse representation

Many Intervals
Many points for estimation
Fine representation

- Distance metrics between histograms
  - Euclidean, intersection => fast to compute
  - Chi-square => better accuracy
  - Try out other distance metrics

Slide adapted after D. Hoiem

# Text classification

# Bag-of-words

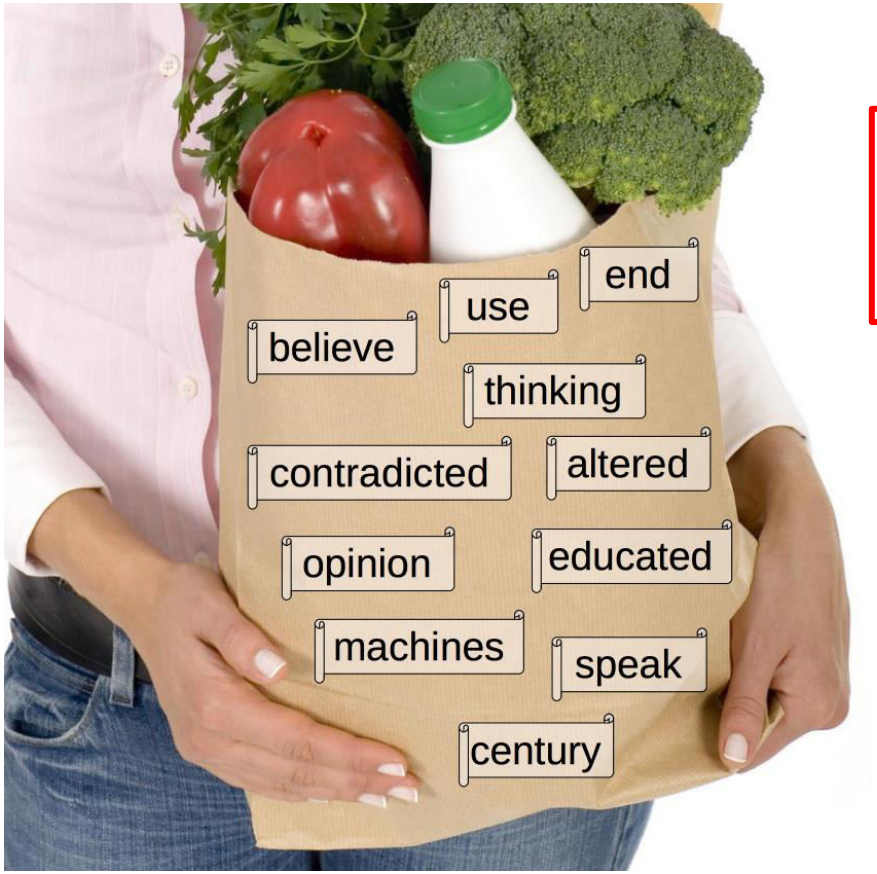
- The bag-of-words model works as follows:

1) Given a set of text documents, all the words from the set are added in a vocabulary

2) A bag-of-words representation is built for each document, by counting down the occurrences of each word in the vocabulary

# Bag-of-words

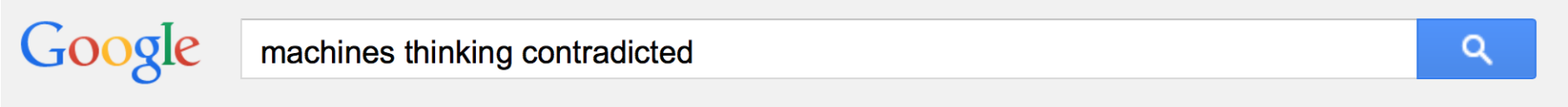
• “I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.” – Alan Turing




Note that stopwords are usually removed!

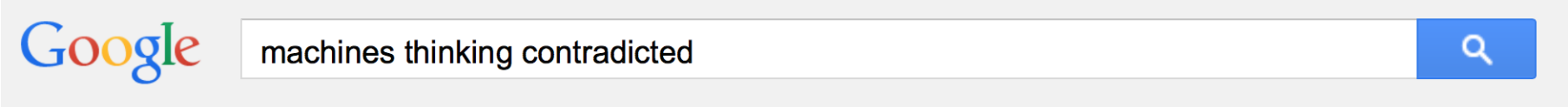
# Bag-of-words

- For a search query such as:

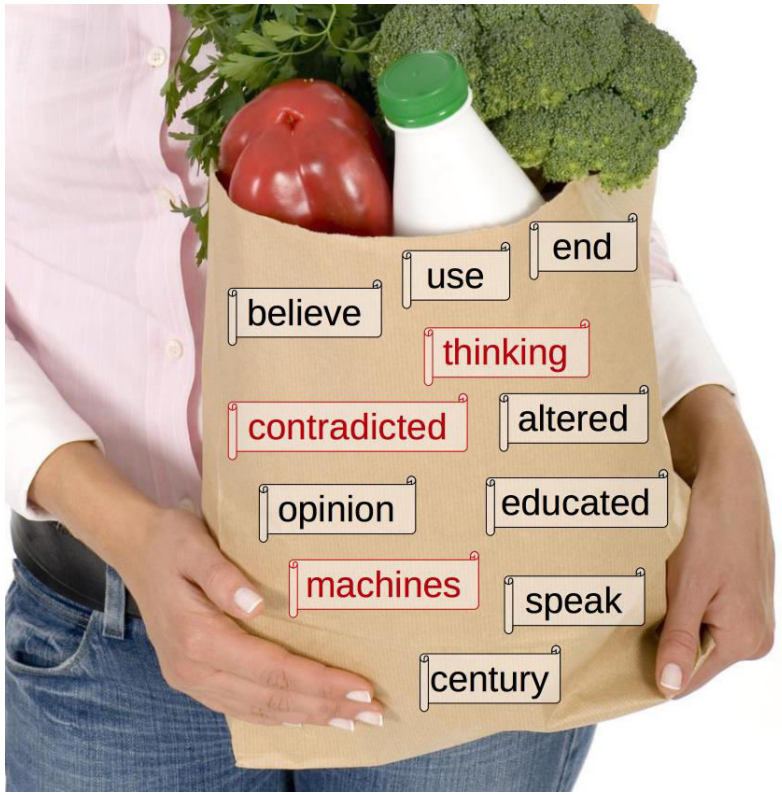


# Bag-of-words

- For a search query such as:

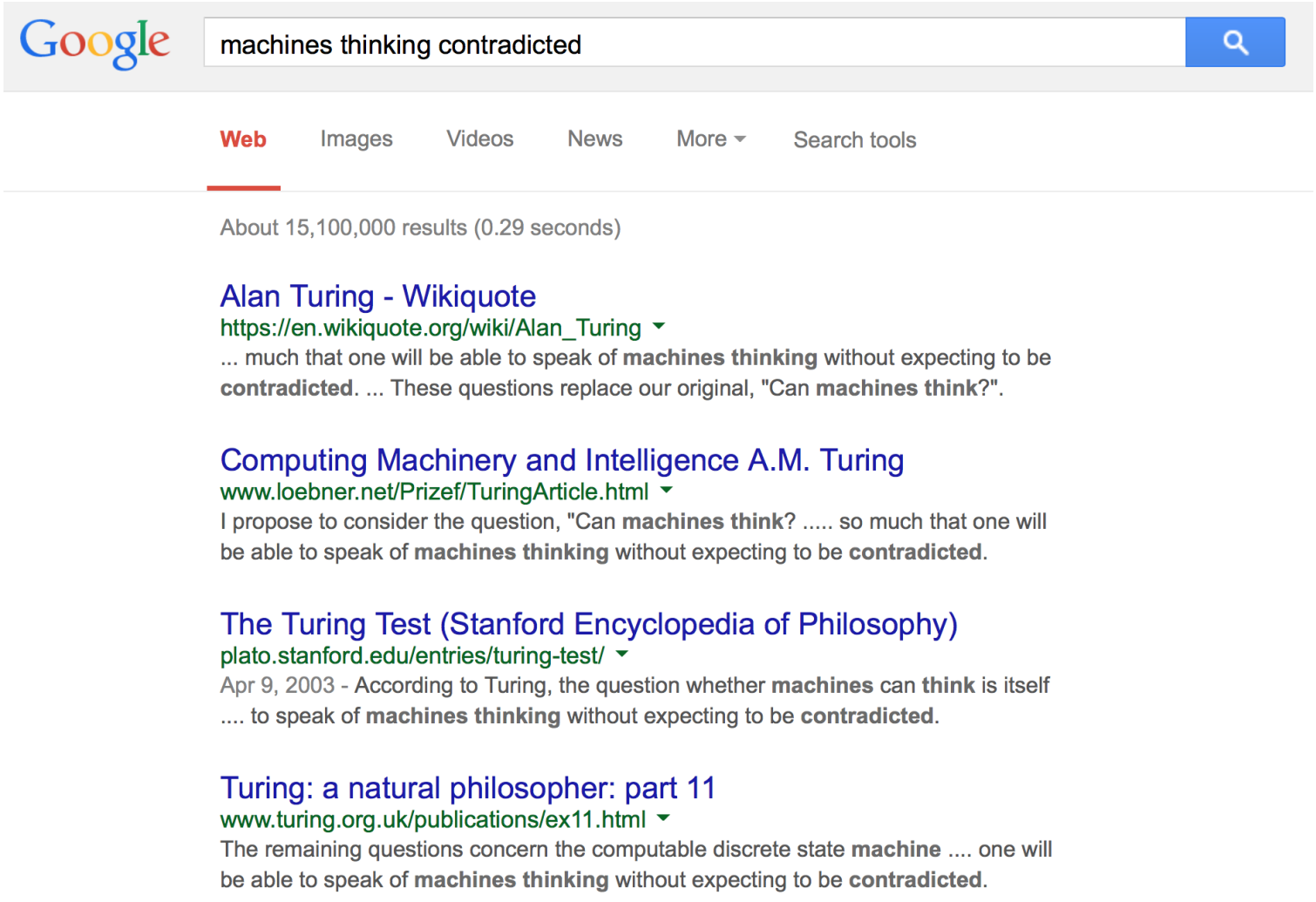



- The system can quickly figure out which documents contain the query terms:



# Bag-of-words

- And return appropriate results:



# Binary term-document incidence matrix

| | Antony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth |
|---|---|---|---|---|---|---|
| Antony | 1 | 1 | 0 | 0 | 0 | 1 |
| Brutus | 1 | 1 | 0 | 1 | 0 | 0 |
| Caesar | 1 | 1 | 0 | 1 | 1 | 1 |
| Calpurnia | 0 | 1 | 0 | 0 | 0 | 0 |
| Cleopatra | 1 | 0 | 0 | 0 | 0 | 0 |
| mercy | 1 | 0 | 1 | 1 | 1 | 1 |
| worser | 1 | 0 | 1 | 1 | 1 | 0 |

Each document is represented by a binary vector $\in \{0,1\}^{|V|}$

# Term-document count matrices

- Consider the number of occurrences of a term in a document:
  - Each document is a count vector in $\mathbb{N}^{|V|}$: a column below

| | Antony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth |
|---|---|---|---|---|---|---|
| Antony | 157 | 73 | 0 | 0 | 0 | 0 |
| Brutus | 4 | 157 | 0 | 1 | 0 | 0 |
| Caesar | 232 | 227 | 0 | 2 | 1 | 1 |
| Calpurnia | 0 | 10 | 0 | 0 | 0 | 0 |
| Cleopatra | 57 | 0 | 0 | 0 | 0 | 0 |
| mercy | 2 | 0 | 3 | 5 | 5 | 1 |
| worser | 2 | 0 | 1 | 1 | 1 | 0 |

# Bag-of-words

- Vector representation doesn't consider the ordering of words in a document.
- Problem: *"John is quicker than Mary"* and *"Mary is quicker than John"* have the same vectors.
- Locations of words can be partially recovered by using:

➢ word n-grams

➢ spatial pyramids [Ionescu & Popescu, Springer, ACVPR, 2016]

# The Vector-Space Model

- Assume *t* distinct terms remain after indexing
- We call them **index terms** or the **vocabulary**
- These "orthogonal" terms form a vector space:

$$\text{Dimension} = t = |\text{vocabulary}|$$

- Each term, *i*, in a document or query, *j*, is given a real-valued weight, $w_{ij.}$
- Both documents and queries are expressed as t-dimensional vectors:

$$d_j = (w_{1j}, w_{2j}, \ldots, w_{tj})$$

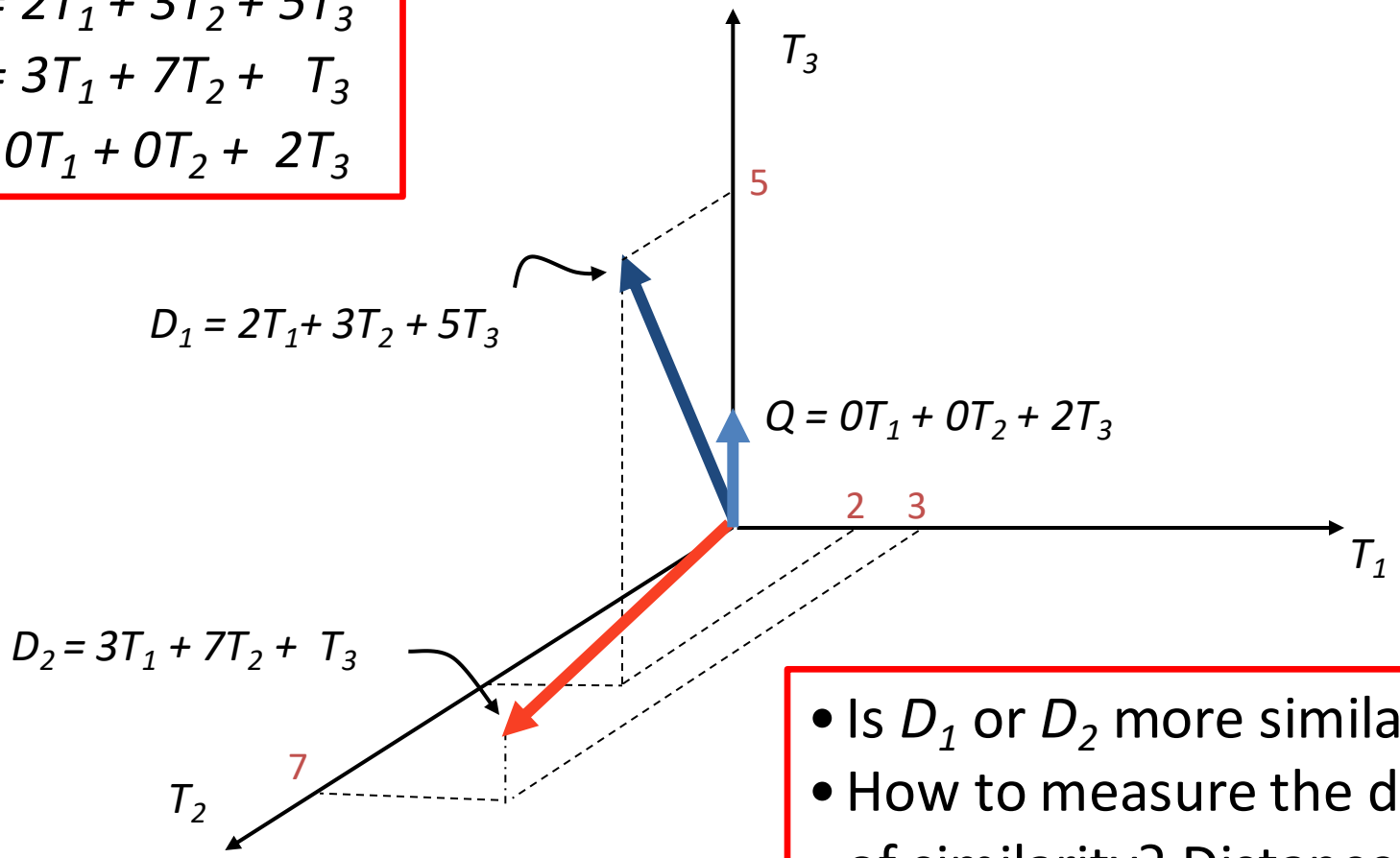
# Graphic Representation

Example:

$D_1 = 2T_1 + 3T_2 + 5T_3$

$D_2 = 3T_1 + 7T_2 + T_3$

$Q = 0T_1 + 0T_2 + 2T_3$




- Is $D_1$ or $D_2$ more similar to Q?
- How to measure the degree of similarity? Distance? Angle? Projection?

# Document Collection

- A collection of *n* documents can be represented in the vector space model by a term-document matrix
- An entry in the matrix corresponds to the "weight" of a term in the document
- Zero means the term has no significance in the document or it simply doesn't exist in the document

| | $T_1$ | $T_2$ | …. | $T_t$ |
|---|---|---|---|---|
| $D_1$ | $w_{11}$ | $w_{21}$ | … | $w_{t1}$ |
| $D_2$ | $w_{12}$ | $w_{22}$ | … | $w_{t2}$ |
| : | : | : | | : |
| : | : | : | | : |
| $D_n$ | $w_{1n}$ | $w_{2n}$ | … | $w_{tn}$ |

# Term Frequency (tf)

- More frequent terms in a document are more important, i.e. more indicative of the topic

  $f_{ij}$ = frequency of term *i* in document *j*

- May want to normalize *term frequency* (*tf*) by dividing by the frequency of the most common term in the document:

$$tf_{ij} = \frac{f_{ij}}{\max_i\{f_{ij}\}}$$

# Term frequency (tf)

- We want to use tf when computing query-document match scores. But how?
- Raw term frequency is not what we want:
  - A document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term
  - But not 10 times more relevant
- Relevance does not increase proportionally with term frequency.

# Document frequency (df)

- Rare terms are more informative than frequent terms
  - E.g.: stop words such as the, an, of, in
- Consider a term in the query that is rare in the collection (e.g., *brewery*).
- A document containing this term is very likely to be relevant to the query *"brewery"*.

➢ We want a high weight for rare terms like *brewery!*

# Inverse Document Frequency (idf)

- Terms that appear in many *different* documents are *less* indicative of overall topic.
- Document frequency of term *i* is:

  $df_i$ = number of documents containing term *i*
- Inverse document frequency of term *i* is:

$$idf_i = \log_2 \left( \frac{N}{df_i} \right)$$

  where N is the total number of documents.
- An indication of a term's *discrimination* power.
- Log used to dampen the effect relative to *tf*.

# TF-IDF Weighting

- A typical combined term importance indicator is *tf-idf weighting*:

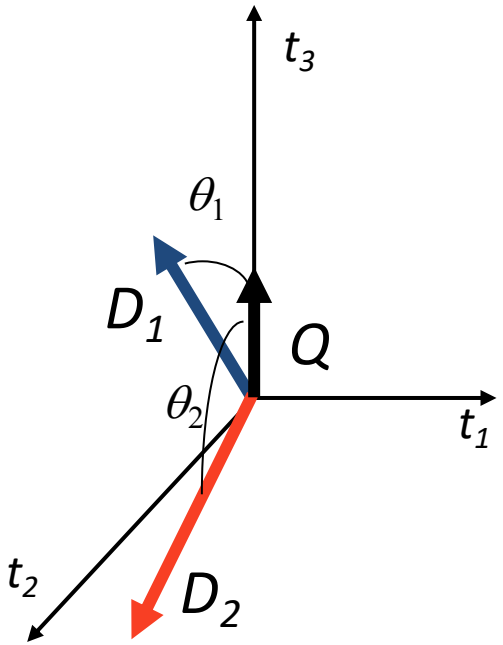
$$w_{ij} = tf_{ij} \cdot idf_i = tf_{ij} \cdot \log_2 \left( \frac{N}{df_i} \right)$$

- A term occurring frequently in the document but rarely in the rest of the collection is given high weight.
- Many other ways of determining term weights have been proposed.
- Experimentally, *tf-idf* has been found to work well.

# Cosine Similarity Measure

- Cosine similarity measures the cosine of the angle between two vectors
- Inner product normalized by the vector lengths:

$$CosSim(\boldsymbol{d}_j, \boldsymbol{q}) = \frac{\langle d_j, q \rangle}{\|d_j\| \cdot \|q\|} = \frac{\sum_{i=1}^{t} w_{ij} \cdot w_{iq}}{\sqrt{\sum_{i=1}^{t} w_{ij}^2} \cdot \sqrt{\sum_{i=1}^{t} w_{iq}^2}}$$




$D_1 = 2T_1 + 3T_2 + 5T_3$ $\quad CosSim(D_1, Q) = \frac{10}{\sqrt{(4+9+25)(0+0+4)}} = 0.81$

$D_2 = 3T_1 + 7T_2 + 1T_3$ $\quad CosSim(D_2, Q) = \frac{2}{\sqrt{(9+49+1)(0+0+4)}} = 0.13$

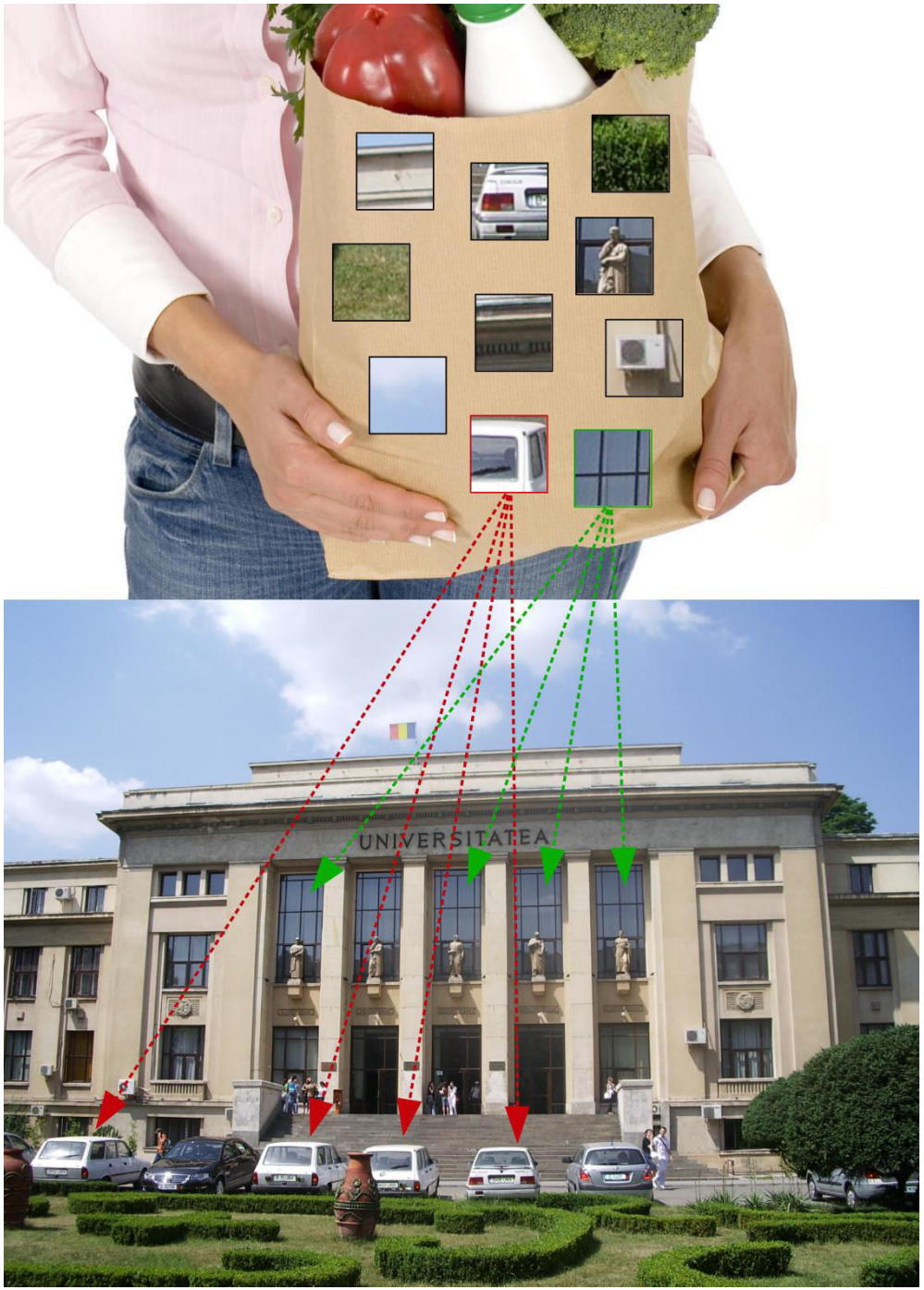
$Q = 0T_1 + 0T_2 + 2T_3$

- $D_1$ is 6 times better than $D_2$ using cosine similarity but only 5 times better using inner product.

# Object class recognition

# Bag-of-visual-words

- A similar model for image classification and image retrieval would be very useful
- But how can we define words in images?
- Words are the atoms that form coherent text
- Atomic repetitive patterns can be observed in images as well


UNIVERSITATEA

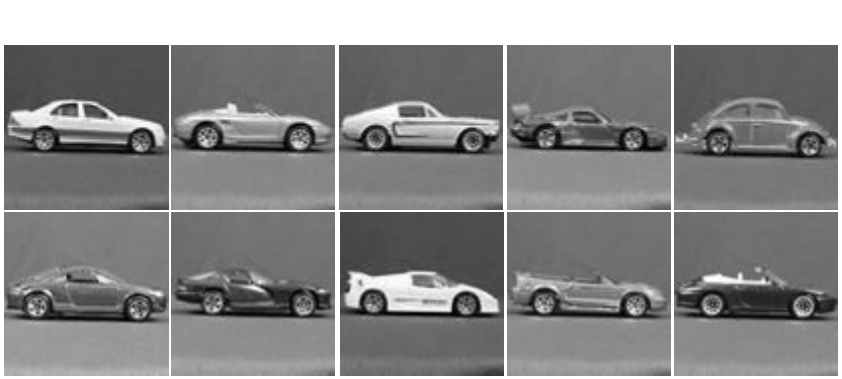
# Bag-of-visual-words

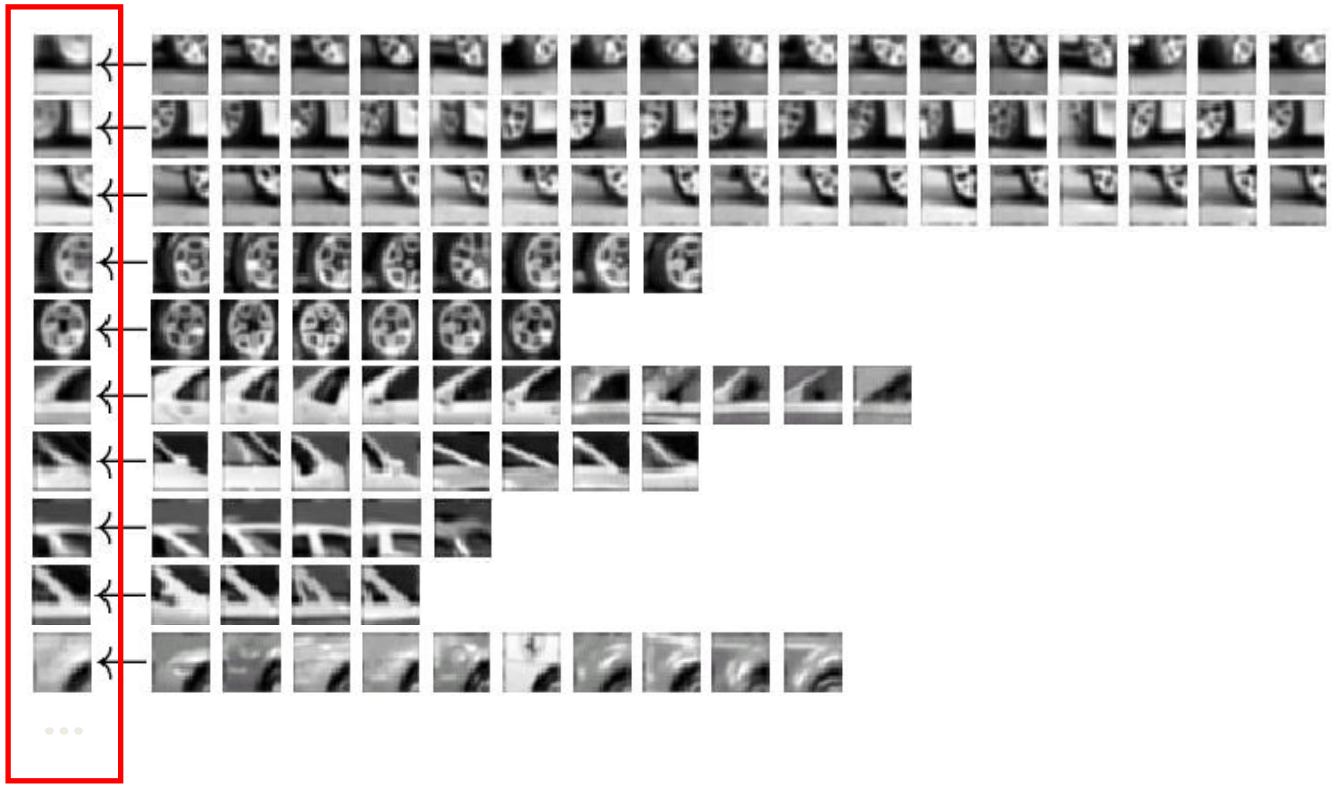
- Atomic repetitive patterns can be observed in images as well
- However these atomic patterns are not identical in all locations in the image, because of various transformations

# Bag-of-visual-words

- A visual word is defined as a group of similar (but not necessarily identical) local image patterns
- Visual words are usually obtained by vector quantization, e.g. k-means clustering

# Visual word vocabulary



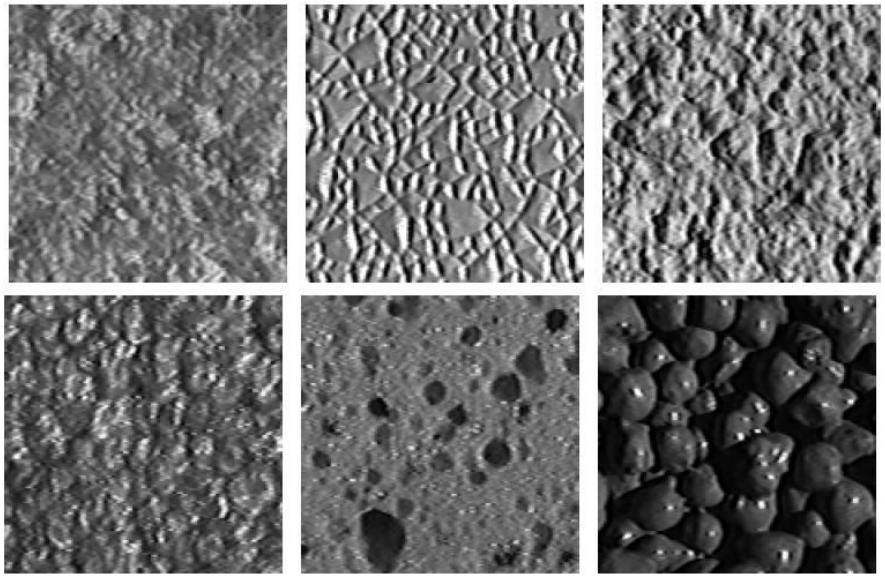


# Object

→ **Bag-of-visual-words**

Slide adapted after L. Fei-Fei

# Bag-of-words-like models: texture representation



- Idea explored for the first time in texture representation
- *Textons* = centroids of clusters obtained by clustering filter responses applied over images
- Representation/description of textures based on the distribution of textons (prototype elements)

Slide adapted after J. Sivic

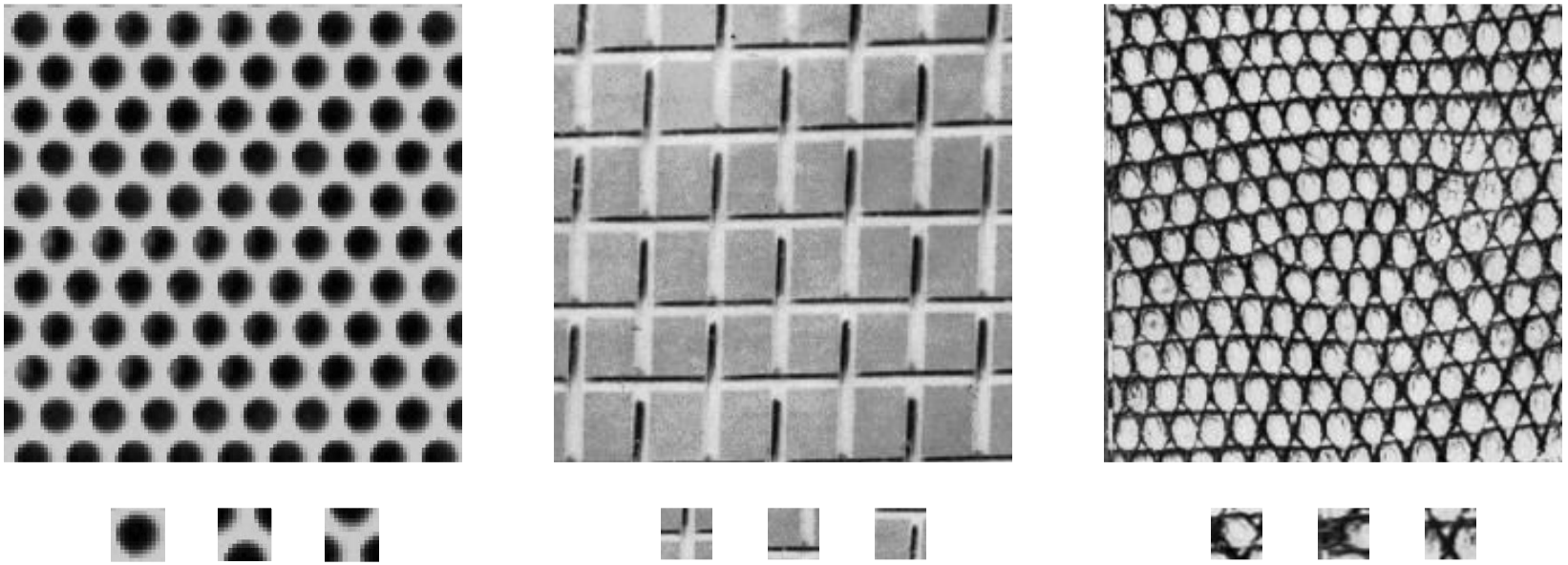
# Texture recognition

regular → stochastic

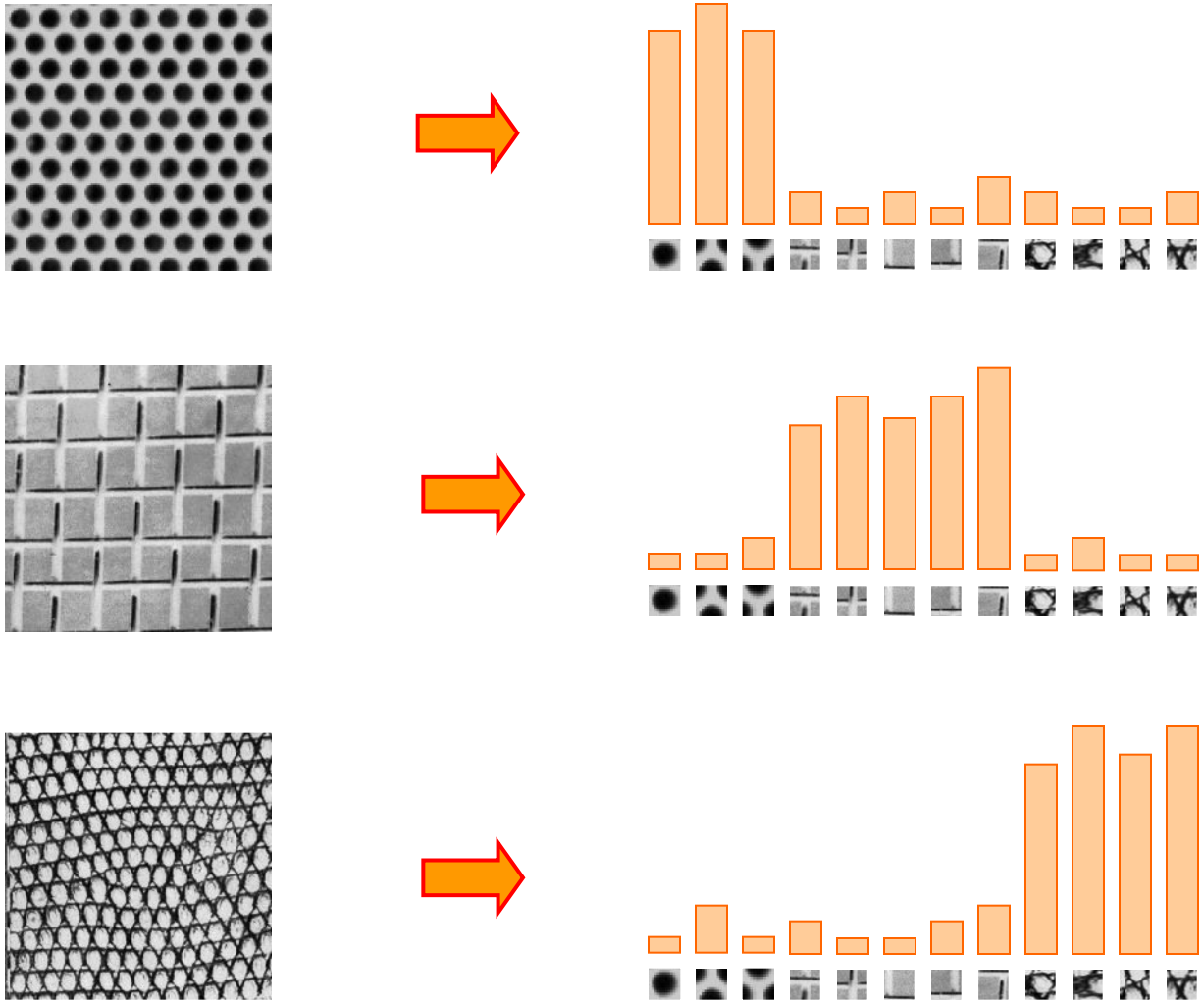
Texture samples (Wikipedia)

# Texture recognition

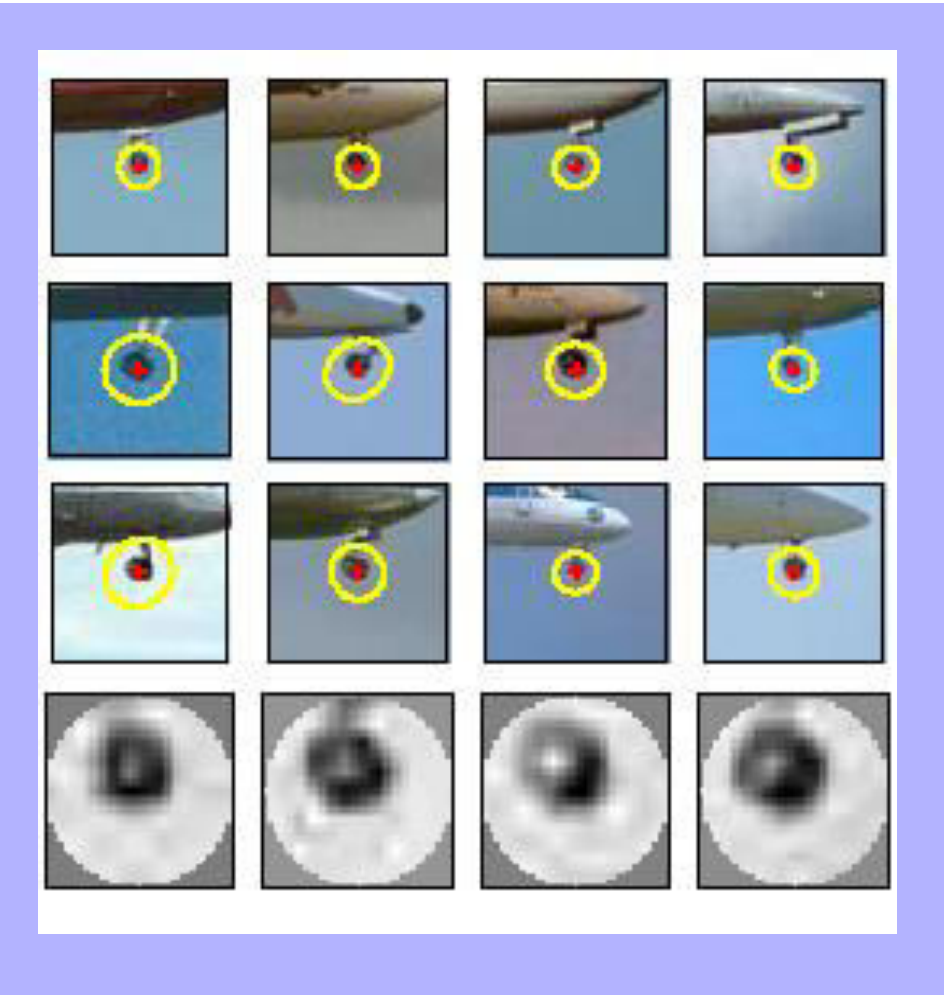
- Texture is characterized by the repetition of basic elements, *textons*
- We represent textures through histograms of textons (we count how many textons of certain type appear in each image)

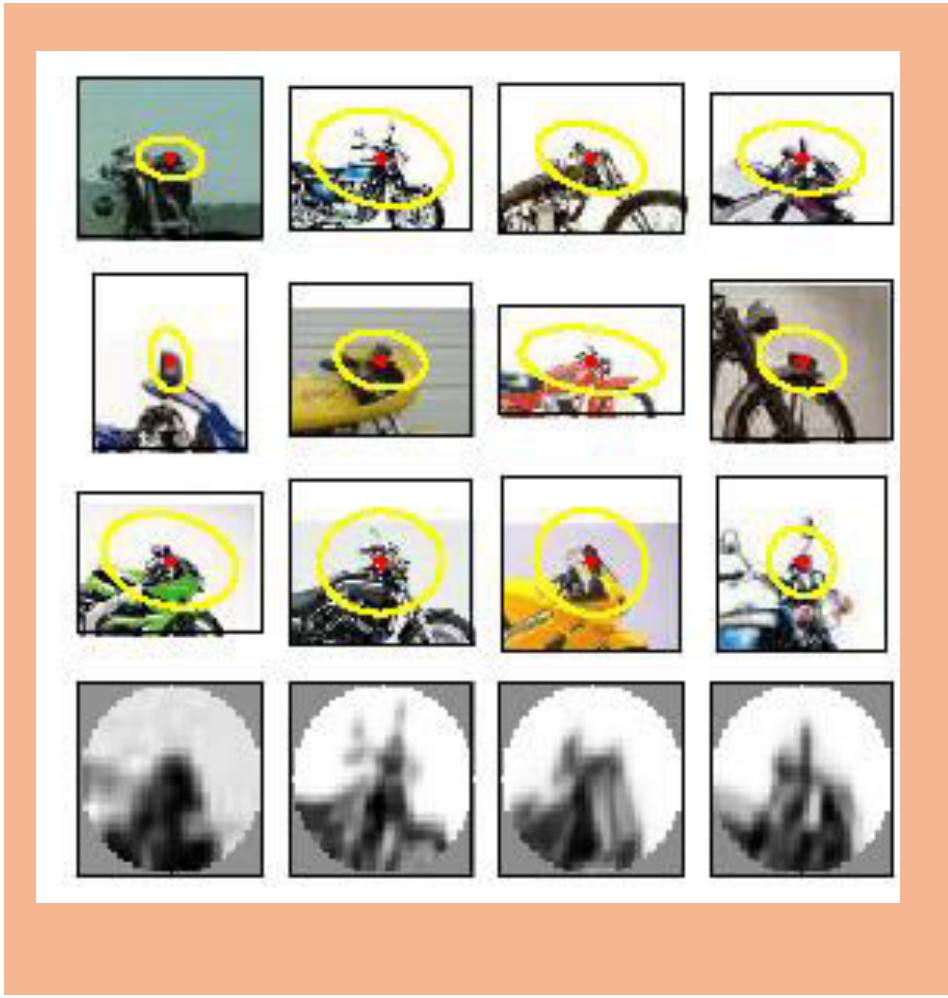


# Texture recognition



# Visual words

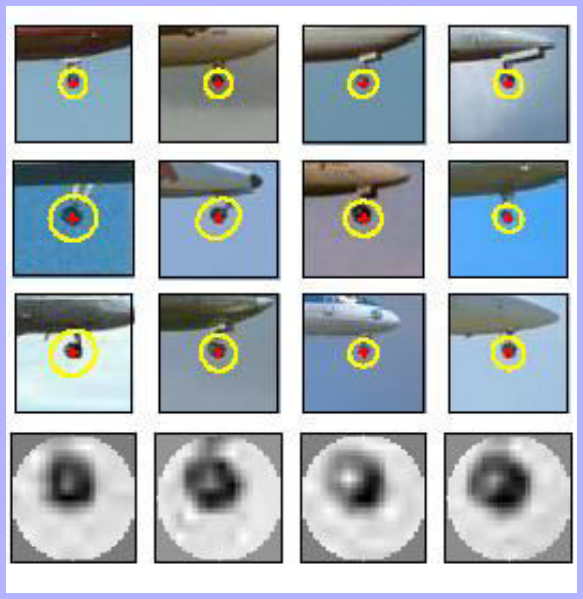


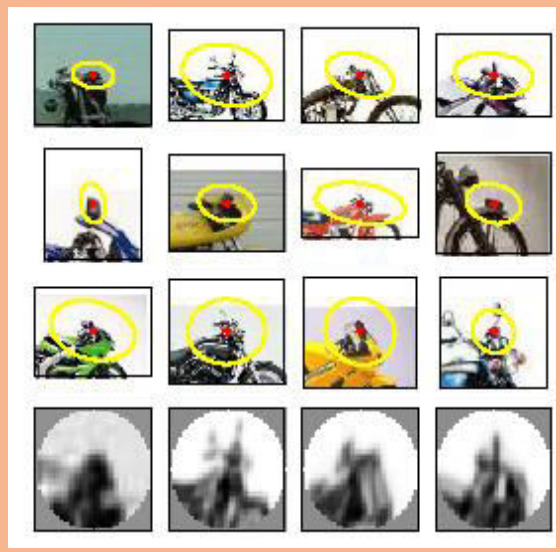


Slide adapted after J. Sivic

# Visual words

- Used to describe object appearance
- Object appearance can vary a lot, even for the same class of objects
- Local appearance (of object parts) has smaller variation
- **Ideally: visual word = object part**



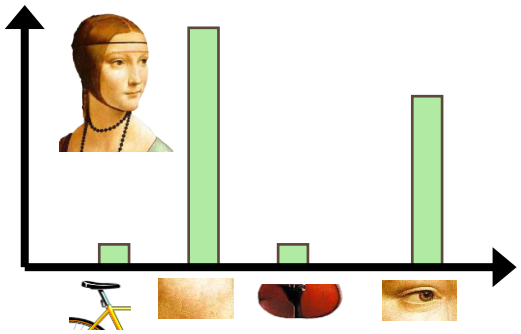
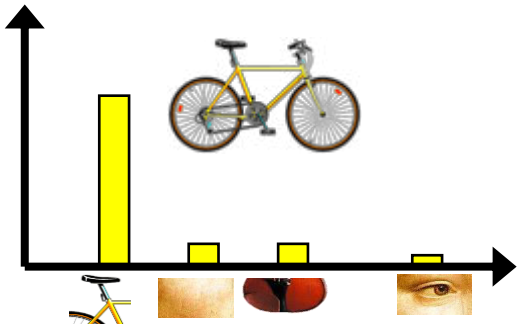
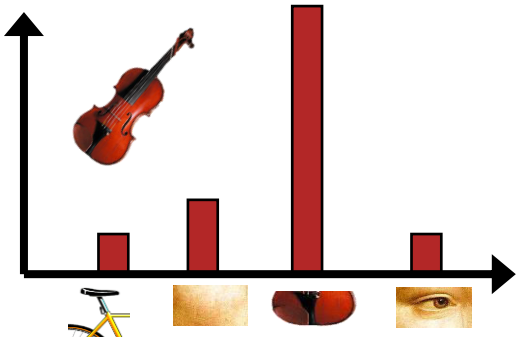


# Bag-of-visual-words

1. Image set

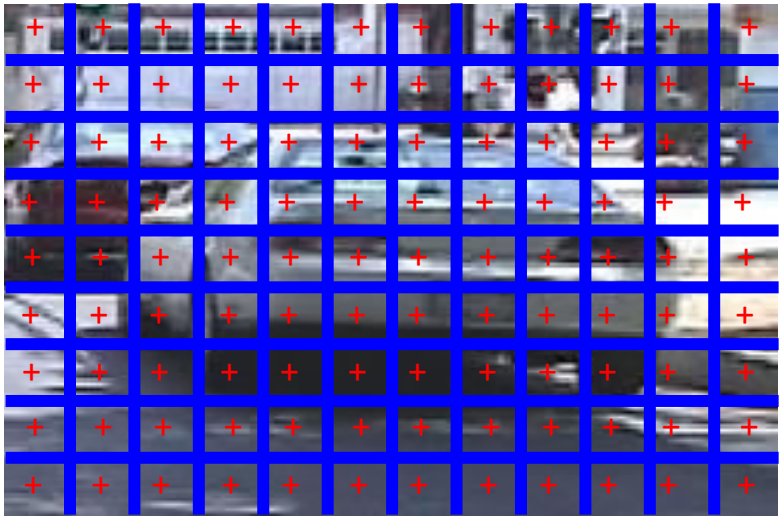


2. Extract features from each image
3. Learn a visual vocabulary = codebook
4. Assign each feature to the nearest “visual word” from the codebook
5. Represent images based on a histogram of visual word frequencies

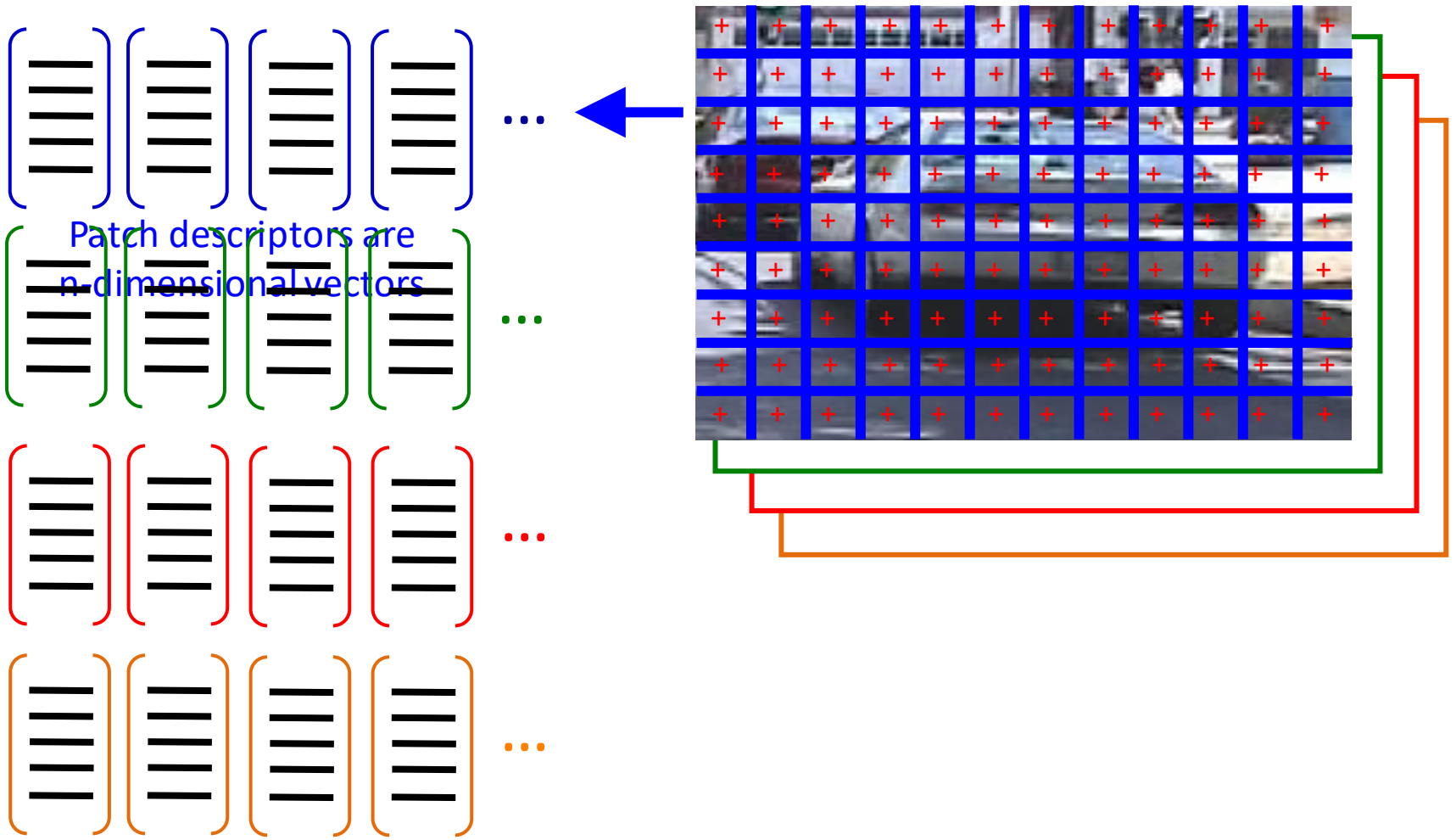
# 2. Feature extraction

- Keypoints on a grid, points of interest or random points in the images
- Describe the **patch** (small image region) centered on each keypoint



Slide adapted after J. Sivic

# 2. Feature extraction



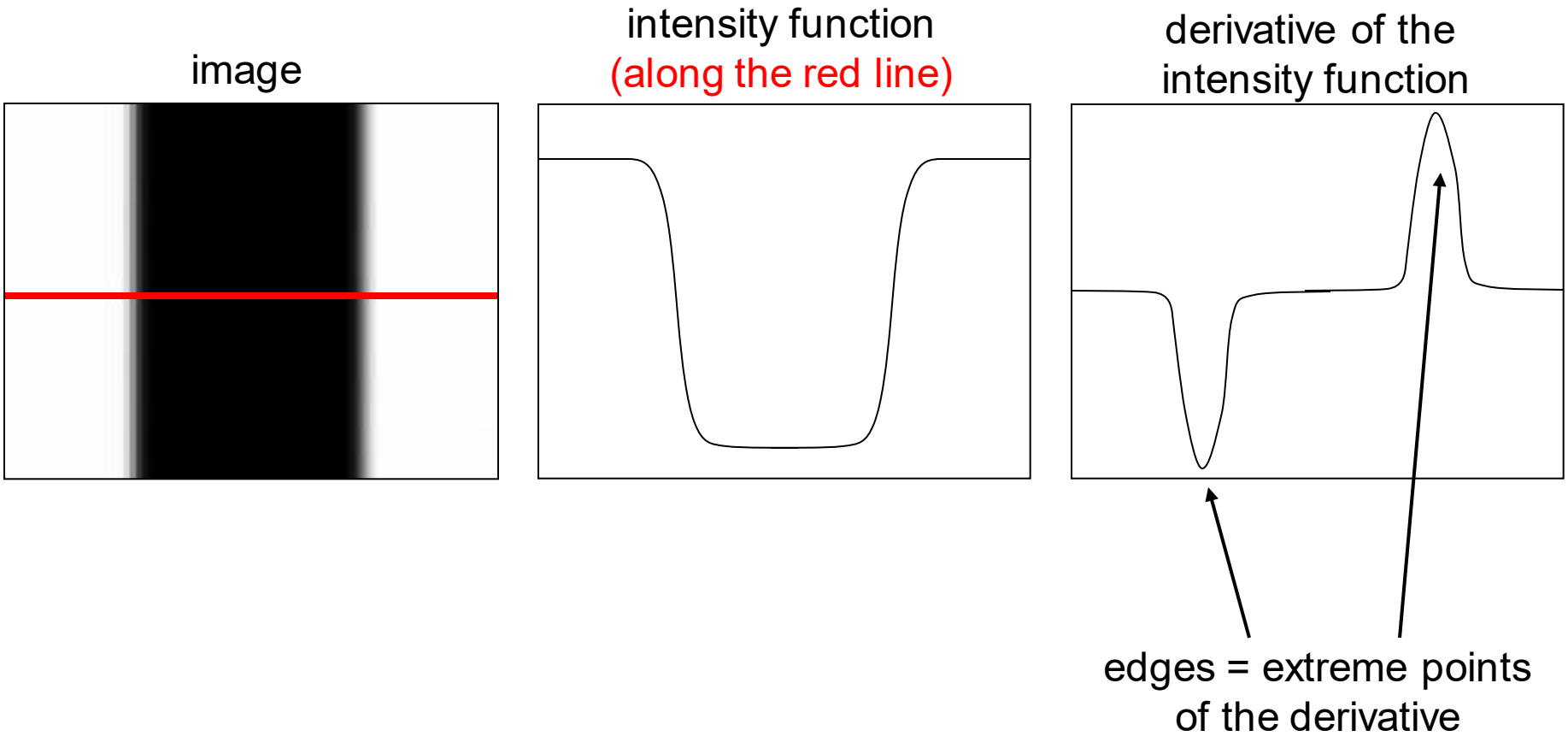

Slide adapted after J. Sivic

# 2. Feature extraction

- Commonly used descriptors:
  - SIFT – Scale Invariant Feature Transform
  - HoG – Histogram of Oriented Gradients (next)

# Edges and derivatives

- An edge is the place where there is a large variation of the intensity function




Slide adapted after S. Lazebnik

# Computing derivatives through convolution

- For a function f(x,y), the partial derivative with respect to x is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

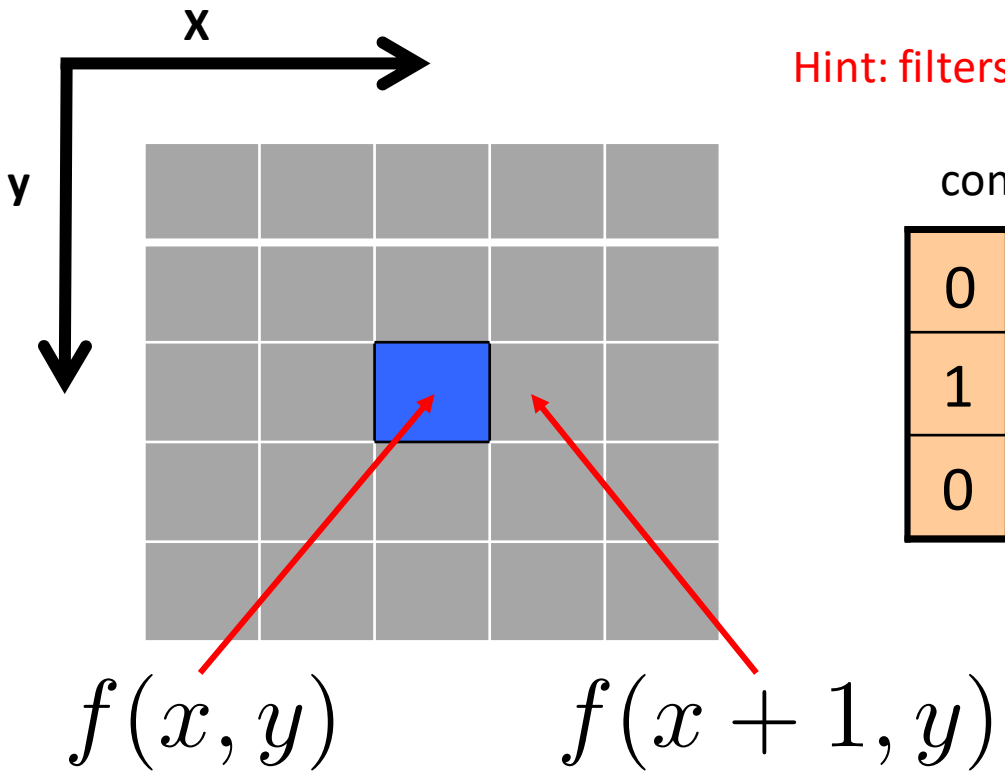
- In the discrete case (images formed of pixels) we can approximate the derivative using finite differences:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

- Can we implement this with a filter? How does the filter look like?

# Computing derivatives through convolution

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{1}$$



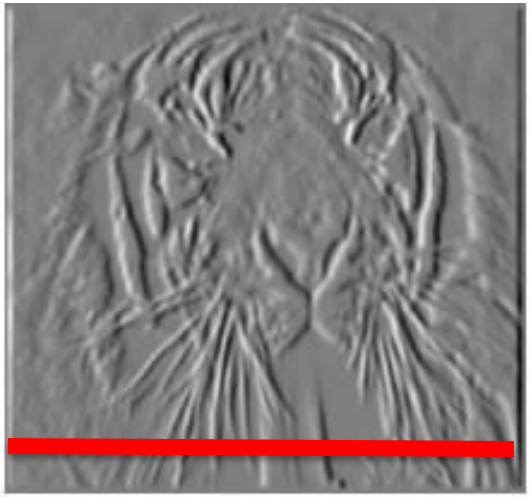

Hint: filters are rotated by 180° during convolution

convolution

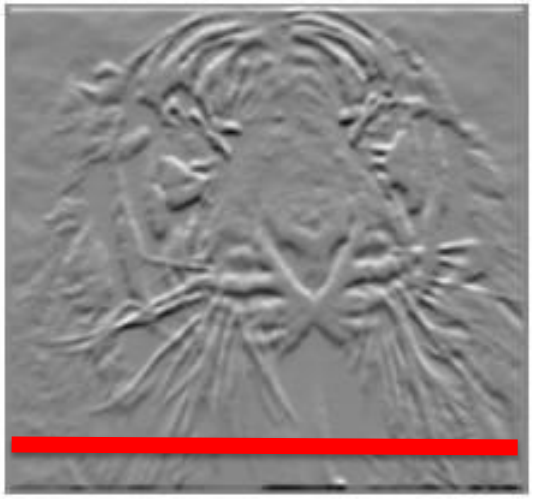
| | | |
|---|---|---|
| 0 | 0 | 0 |
| 1 | -1 | 0 |
| 0 | 0 | 0 |

# Partial derivatives of an image

$$\frac{\partial f(x,y)}{\partial x}$$



| 1 | -1 |
|---|---|



$$\frac{\partial f(x,y)}{\partial y}$$

?

| 1 |
|---|
| -1 |

or

| -1 |
|---|
| 1 |

Which image shows the variantion on x / y?

(check out the convolution filters)

Slide adapted after S. Lazebnik

# Sobel filters

- There are filters that can better approximate the derivatives
- ➢ **Sobel filters** compute the partial derivatives taking into account larger vicinities

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

Vertical Sobel filter for computing partial derivative

$$\frac{\partial f(x,y)}{\partial x}$$

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

Horizontal Sobel filter for computing partial derivative

$$\frac{\partial f(x,y)}{\partial y}$$

# Computing image gradient

$M_x$ =

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

$M_y$ =

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

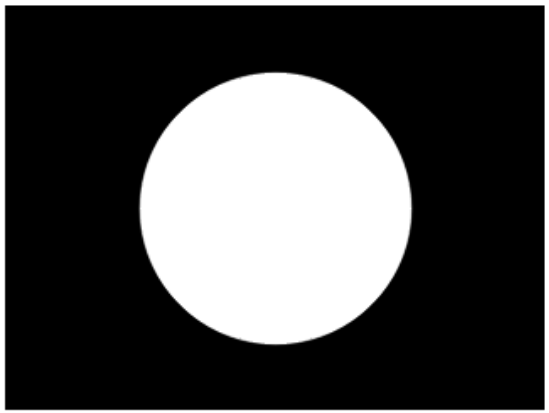
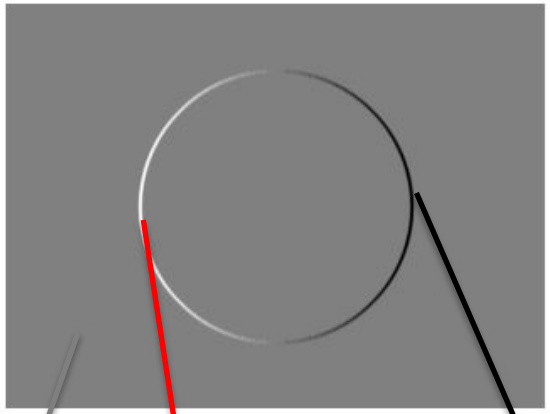
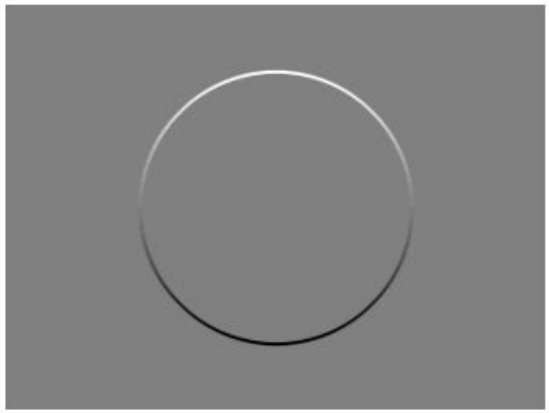


image f



$\frac{\partial f}{\partial x}$



$\frac{\partial f}{\partial y}$

values = 0

large values

small values

# Computing image gradient

$M_x$ =

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

$M_y$ =

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

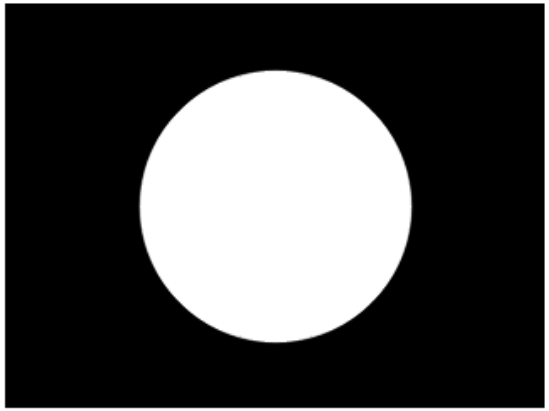
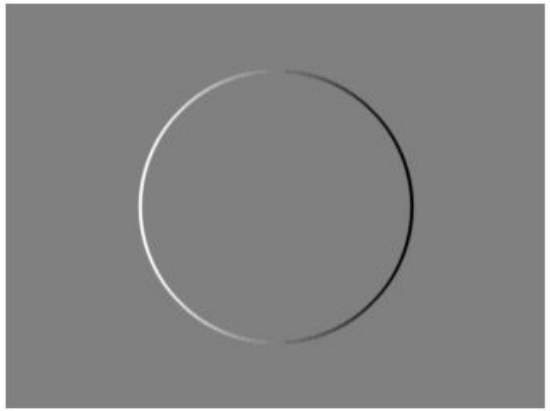
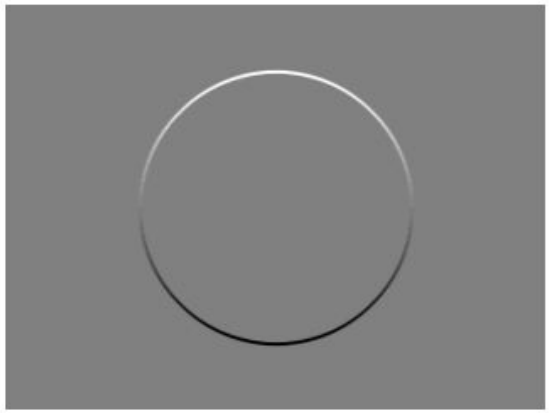


image f



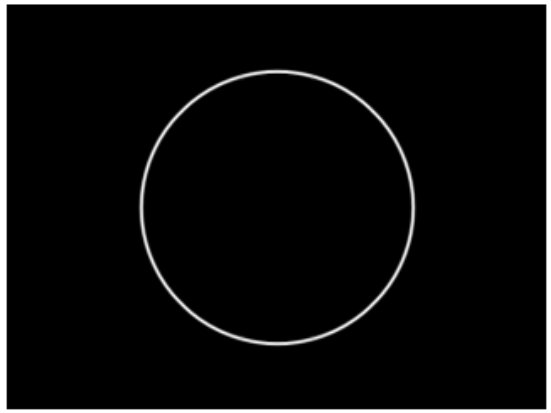
$$\frac{\partial f}{\partial x}$$



$$\frac{\partial f}{\partial y}$$

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Gradient magnitude



# Computing image gradient

$M_x$ =

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

$M_y$ =

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

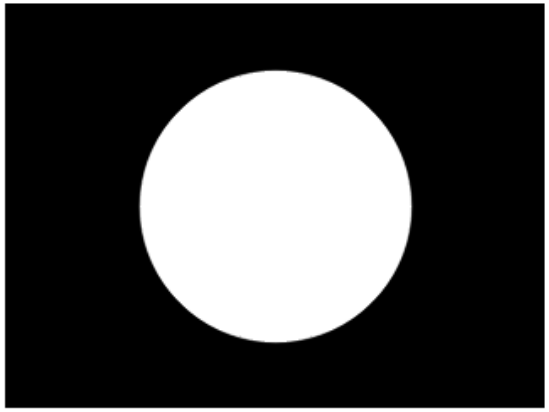
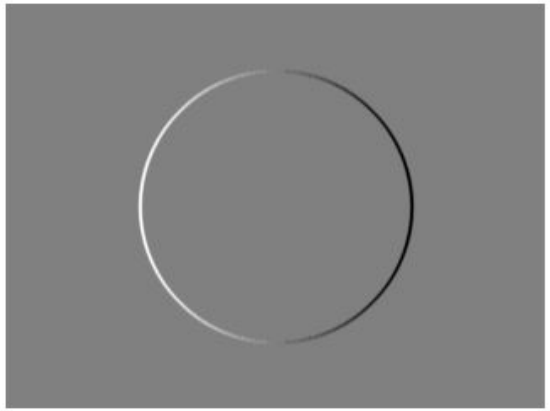
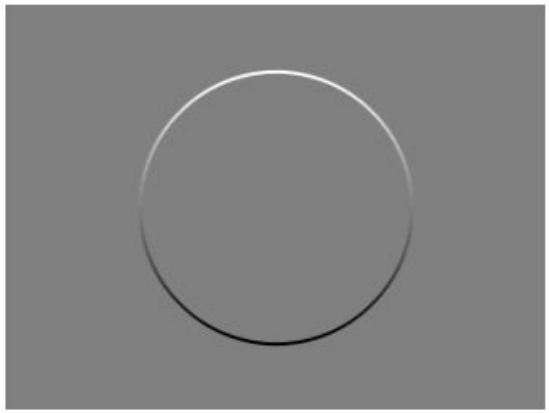


image f



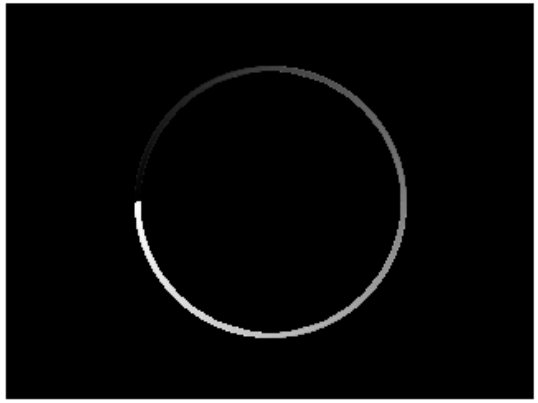
$$\frac{\partial f}{\partial x}$$

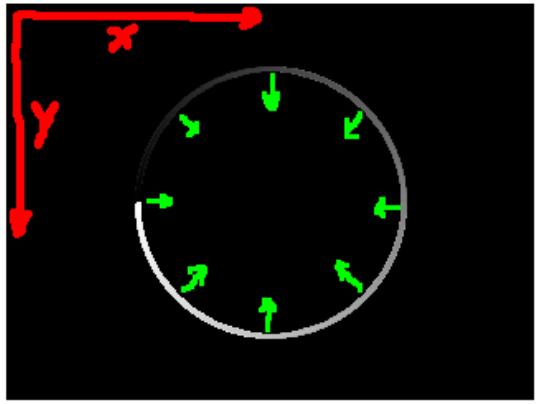


$$\frac{\partial f}{\partial y}$$

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

Gradient orientation





# Computing image gradient

$M_x$ =

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

$M_y$ =

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

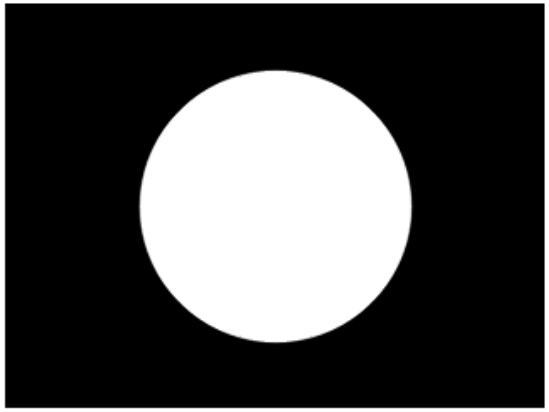
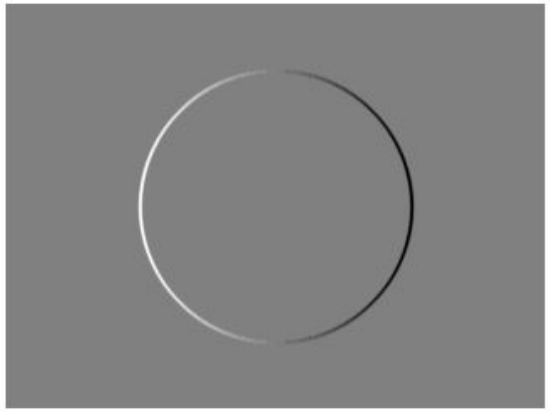
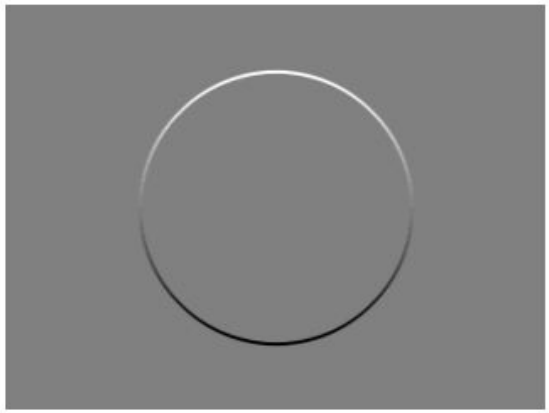


image f



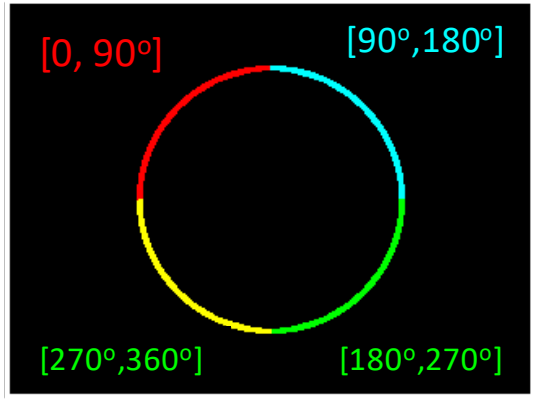
$$\frac{\partial f}{\partial x}$$

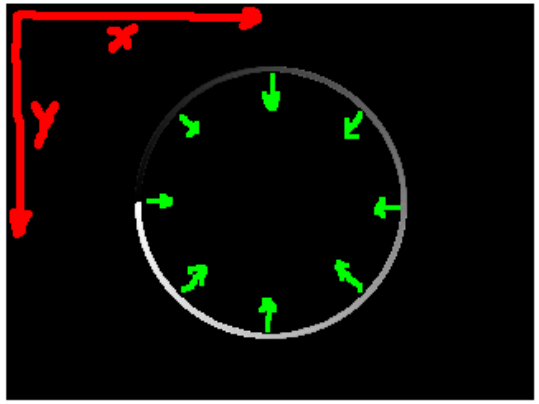


$$\frac{\partial f}{\partial y}$$

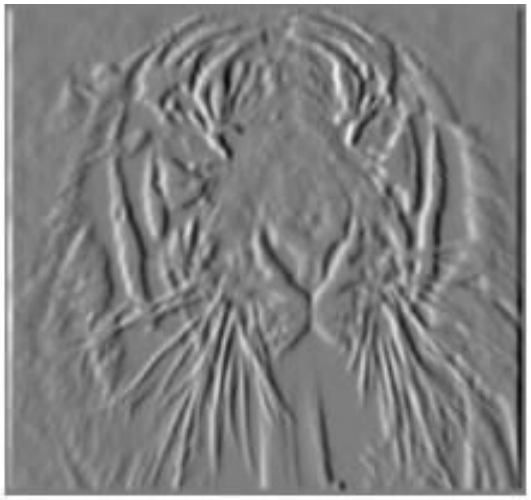
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

[0, 90°]
[90°, 180°]
[180°, 270°]
[270°, 360°]

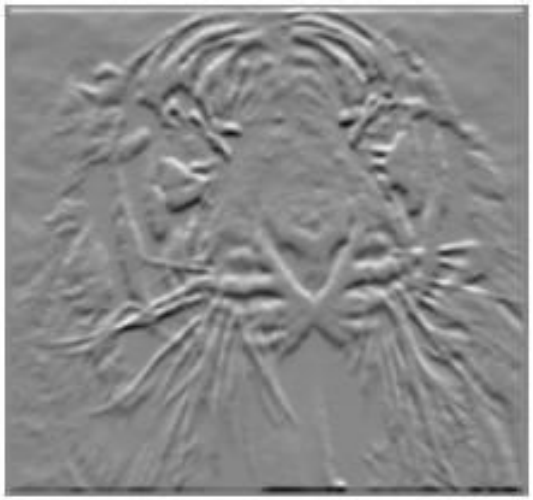


# Computing image gradient



$$\frac{\partial f(x,y)}{\partial x}$$



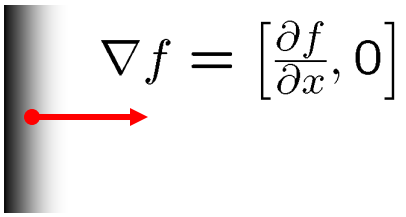
$$\frac{\partial f(x,y)}{\partial y}$$

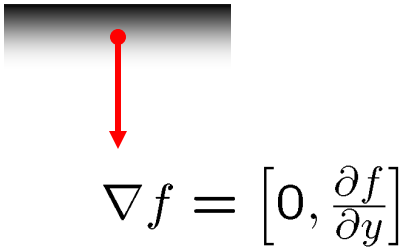
$$\sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}$$

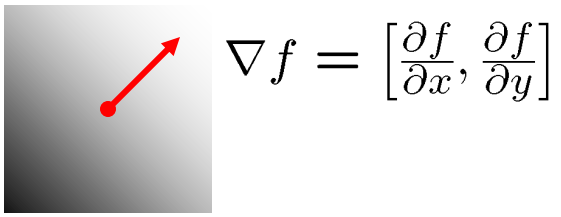
Gradient magnitude

# Image Gradient: Summary

Image Gradient: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$




The gradient shows the direction of the largest intensity variation

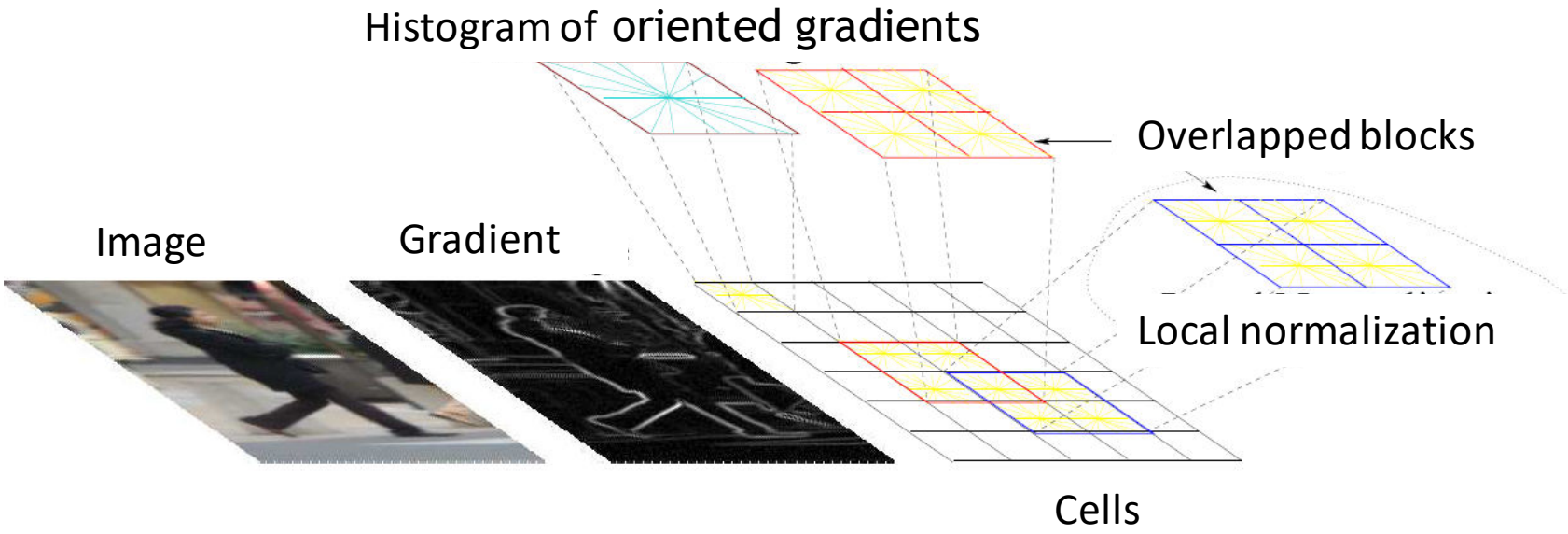
- What is the link between the direction of the gradient and that of the edge?
- **The gradient direction is perpendicular on the edge direction**

Gradient direction is given by: $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

An edge is characterized by the gradient magnitude:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \quad \text{or} \quad \|\nabla f\| = \left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right|$$

# HoG for human detection




- Split window into cells
- For each cell, compute a histogram of oriented gradients
- Group cells into blocks
- HoG = histograms of blocks

Navneet Dalal and Bill Triggs,
Histograms of Oriented Gradients for Human Detection, CVPR05

http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

Slide adapted after K. Grauman

# HoG for human detection

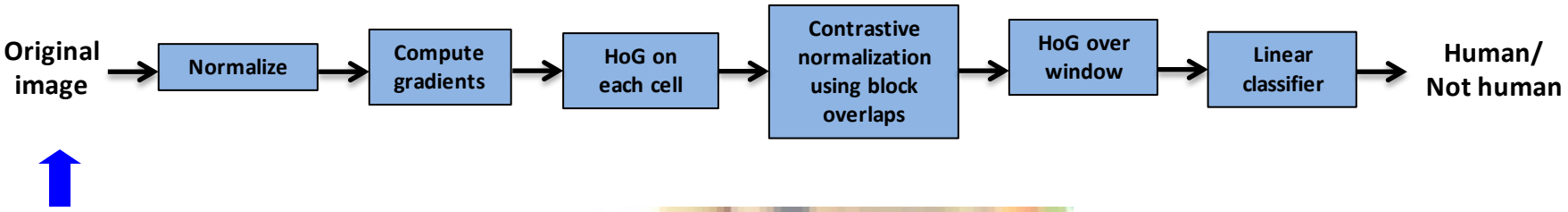

# HoG for human detection


- resize window to 128 x 64 pixels
- divide image into cells of 8 x 8 pixels
- compute image gradient (for each pixel we have orientation and magnitude)
- for each cell compute a histogram of oriented gradients
- group cells into blocks
- HoG = a descriptor obtained by concatenating histograms of 105 blocks

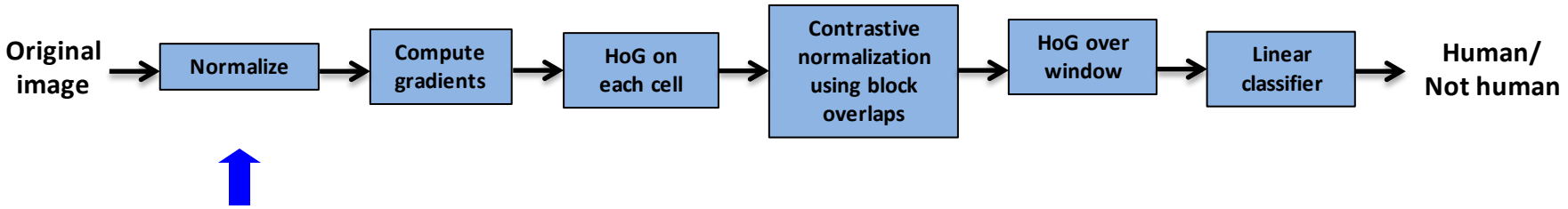
Navneet Dalal and Bill Triggs,
Histograms of Oriented Gradients for Human Detection, CVPR05

http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

Slide adapted after K. Grauman

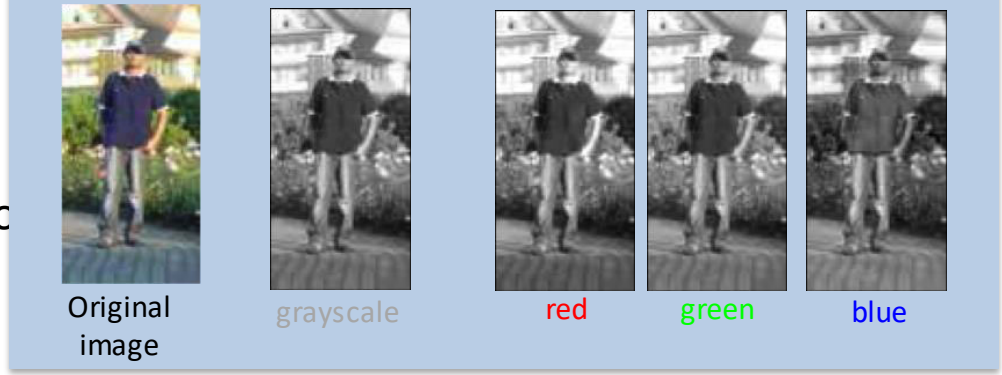



Slide adapted after N. Dalal




- Tested with
  - RGB
  - grayscale

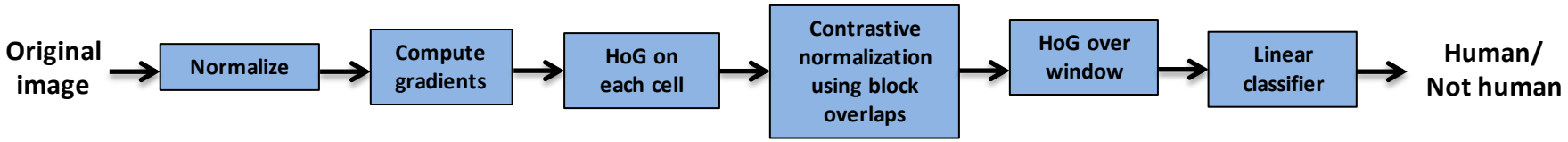
  } better perfo

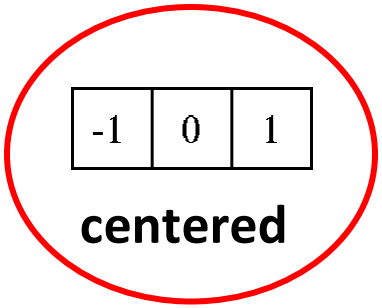


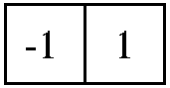

- Gamma normalization (transforming intensity values/colors)
  - without normalization
  - squared root
  - logarithm

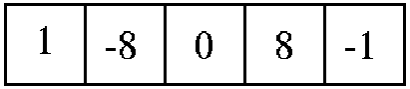
  } better performance with square root

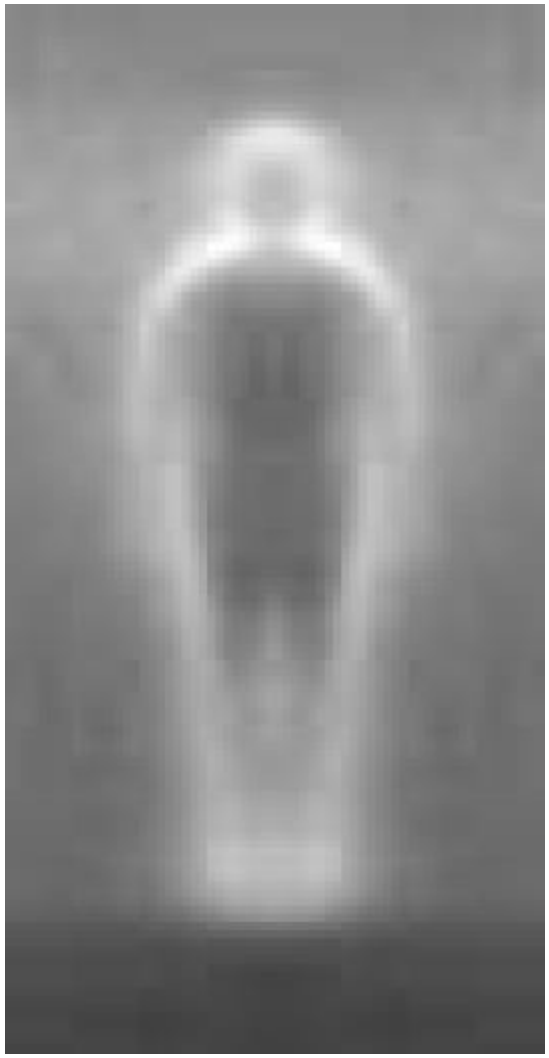
Slide adapted after N. Dalal

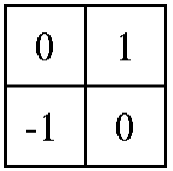


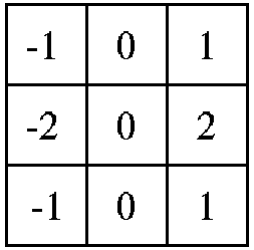



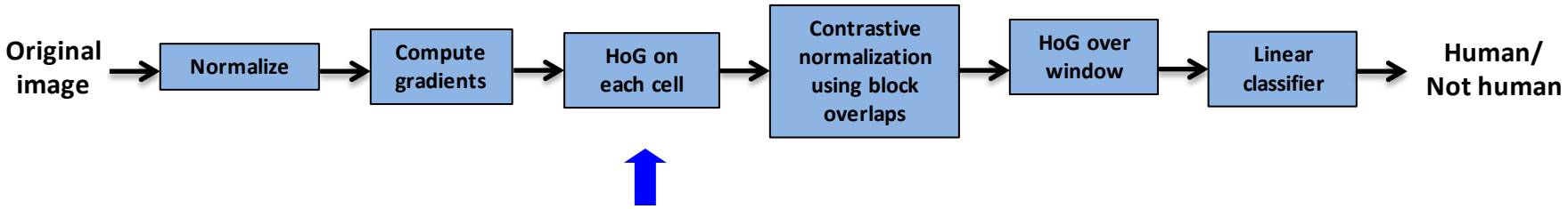



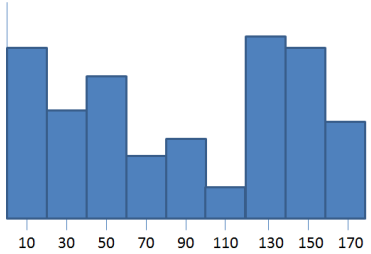


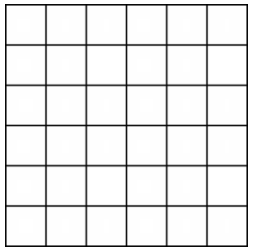




Slide adapted after N. Dalal




- Histograms of Oriented Gradients
  - Orientation-based: 9 intervals for orientations between [0°, 180°]

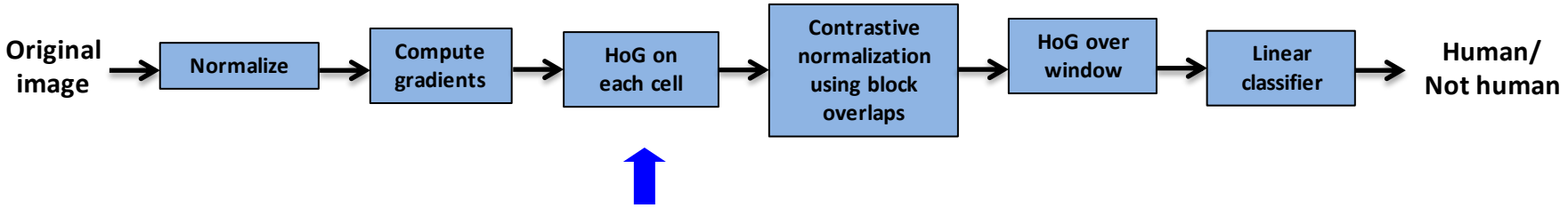


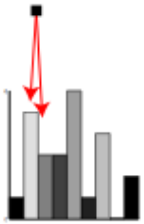
  - Cell-based

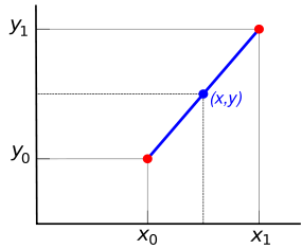
  - The contribution of each pixel is directly proportional with gradient magnitude

Slide adapted after N. Dalal

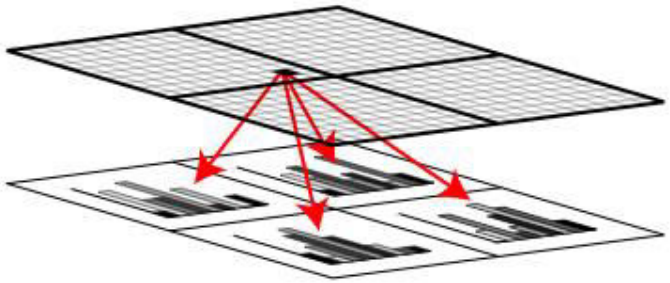


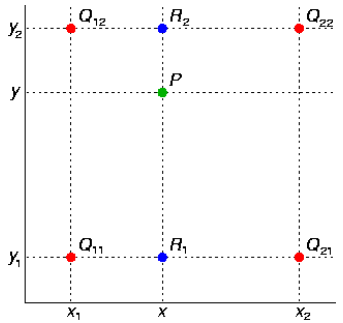

- Histograms of Oriented Gradients
  - Trilinear interpolation:
    - linear based on orientation



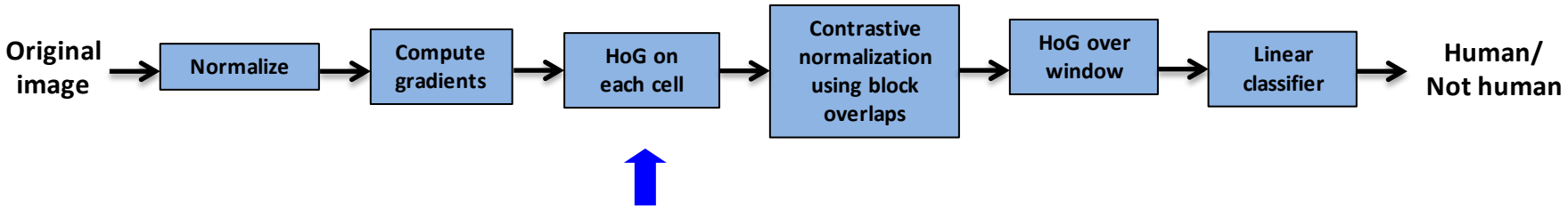



    - bilinear based on cells

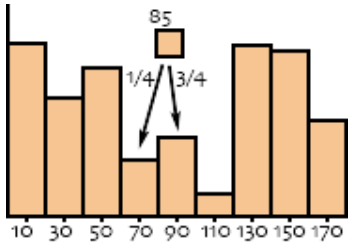





Slide adapted after N. Dalal

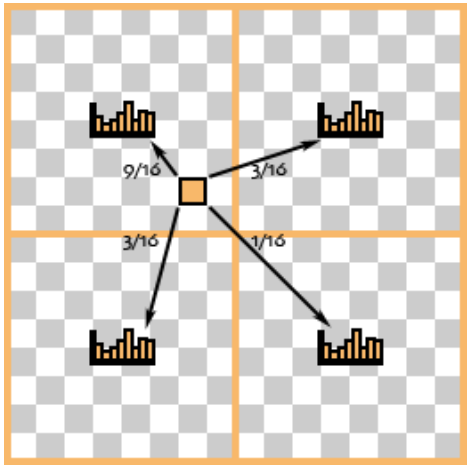



- Trilinear interpolation example:
  - linear based on orientation:
    - Θ = 85°
    - distances from centers 70°, 90° are 15°, 5°
    - 5/20 = ¼ contribution in interval centered in 70°
    - 15/20 = ¾ contribution in interval centered in 90°
    - soft-assignment

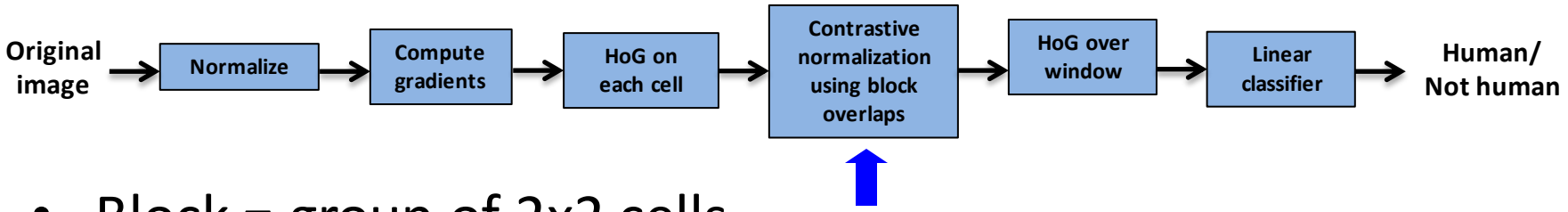


Orientarea gradienților

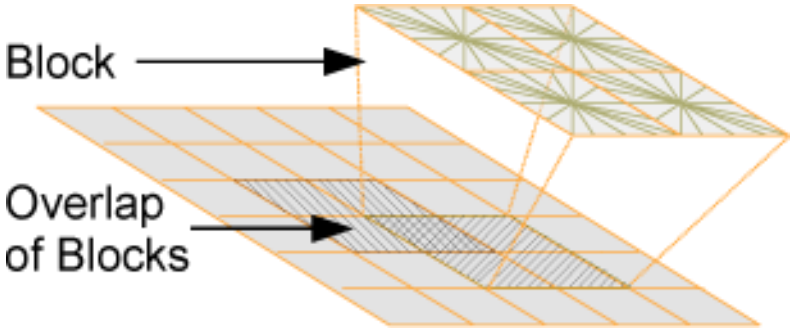
  - bilinear based on cells:
    - distances from centers: left – 2, right – 6, top – 2, down – 6
    - partial contributions: 6/8 = ¾ - left, 2/8 = ¼ right
    - partial contributions: 6/8 = ¾ - top, 2/8 = ¼ down
    - total contributions:
      - 6/8 * 6/8 = 9/16 left – top
      - 6/8 * 2/8 = 3/16 left – down
      - 2/8 * 6/8 = 3/16 right – top
      - 2/8 * 2/8 = 1/16 right -down




Slide adapted after N. Dalal

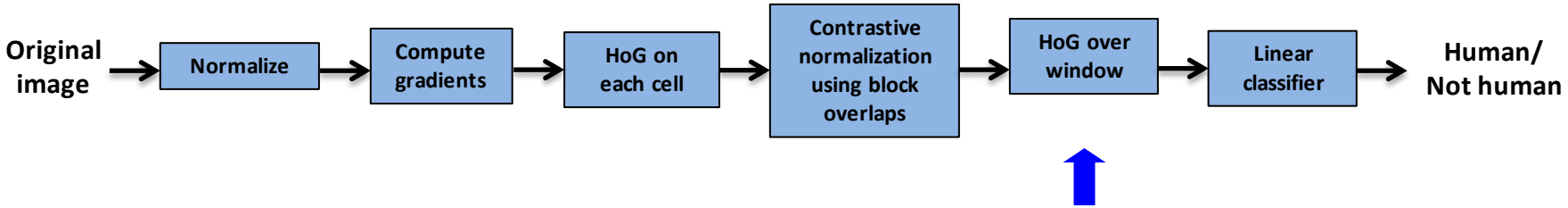



- Block = group of 2x2 cells
- Block histogram = concatenate histograms of the 4 cells inside the block (histograms are normalized with respect to the entire block)
- A cell belongs to multiple blocks
- Normalization
  - there are different approaches

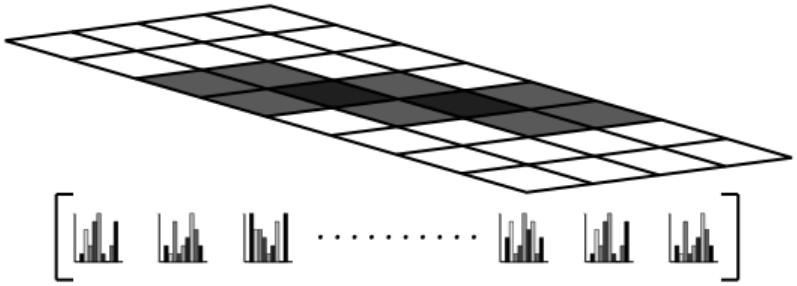



$$\mathbf{v} \rightarrow \frac{\mathbf{v}}{\|\mathbf{v}\|_1 + \epsilon} \qquad \mathbf{v} \rightarrow \frac{\mathbf{v}}{\sqrt{\|\mathbf{v}\|_2^2 + \epsilon}} \qquad \mathbf{v} \rightarrow \sqrt{\frac{\mathbf{v}}{\|\mathbf{v}\|_1 + \epsilon}}$$

Slide adapted after N. Dalal




- Concatenated block histograms

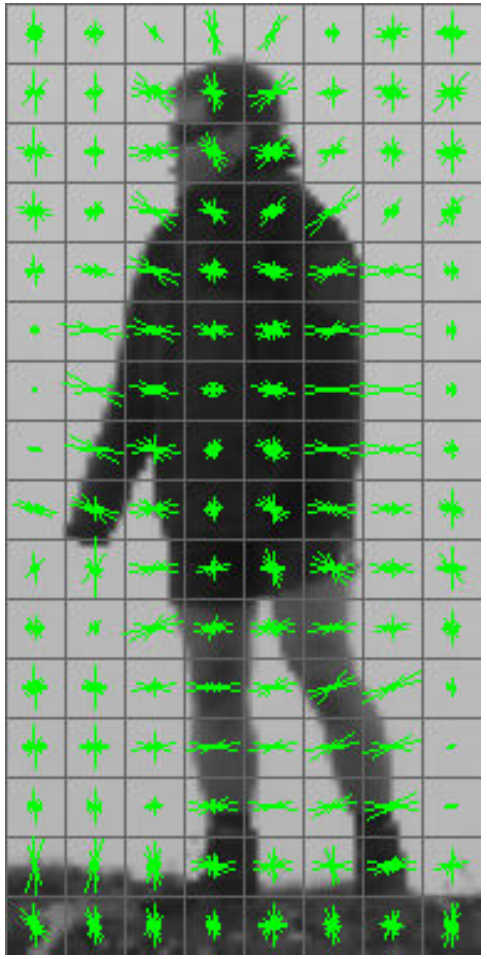


# orientations

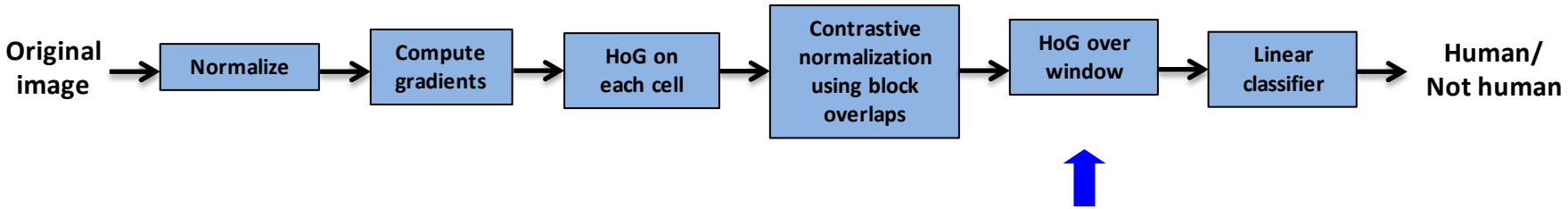
HoG size = 15 x 7 x 4 x 9 = 3780

# blocks

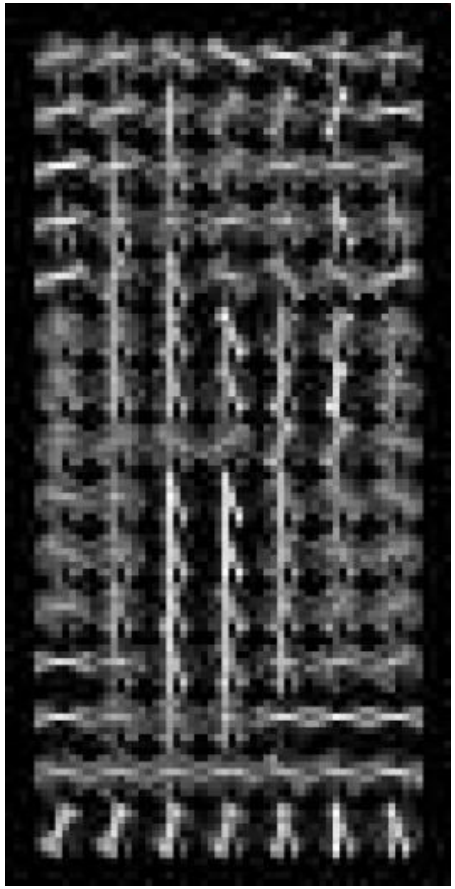
# cells in block



Slide adapted after N. Dalal




- Visualizing the descriptor

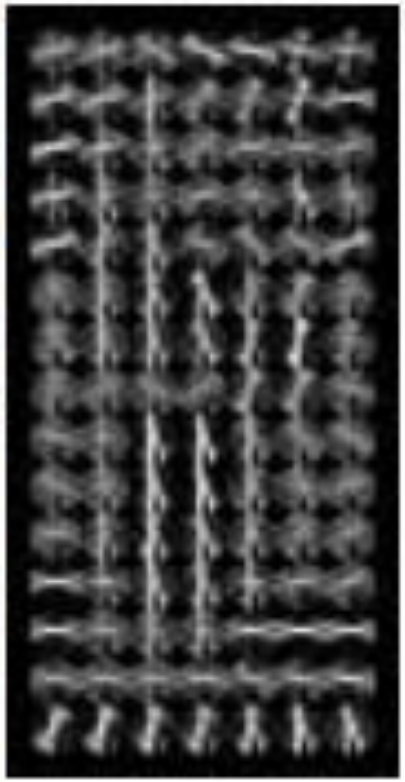


Slide adapted after N. Dalal

# HoG + SVM

- Visualization

Test image



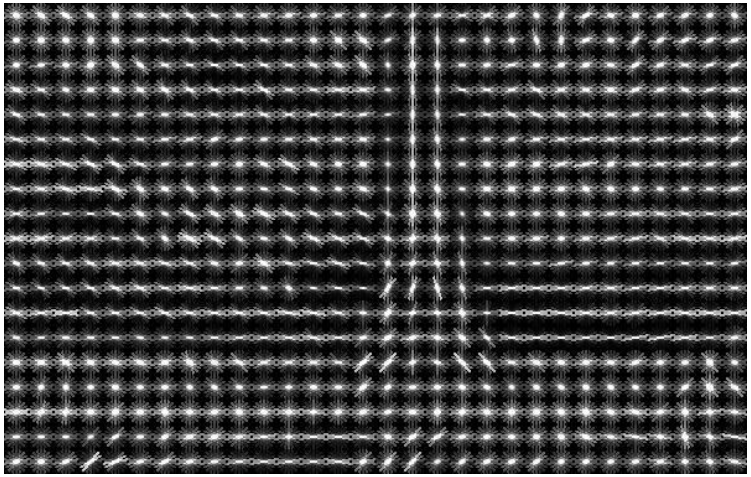
HoG descriptor

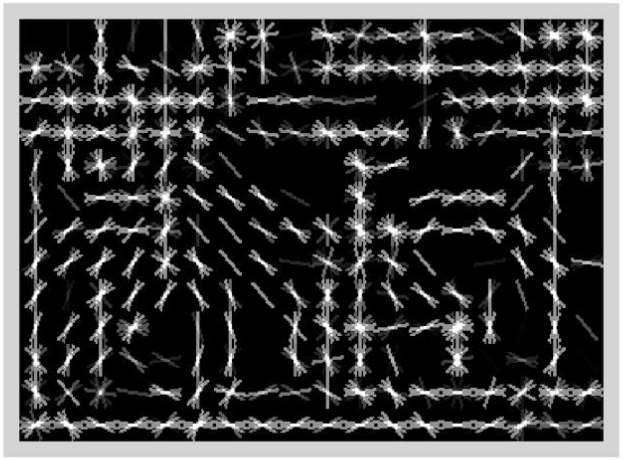
HoG descriptor with orientations weighted as positive by the SVM

# HoG + SVM

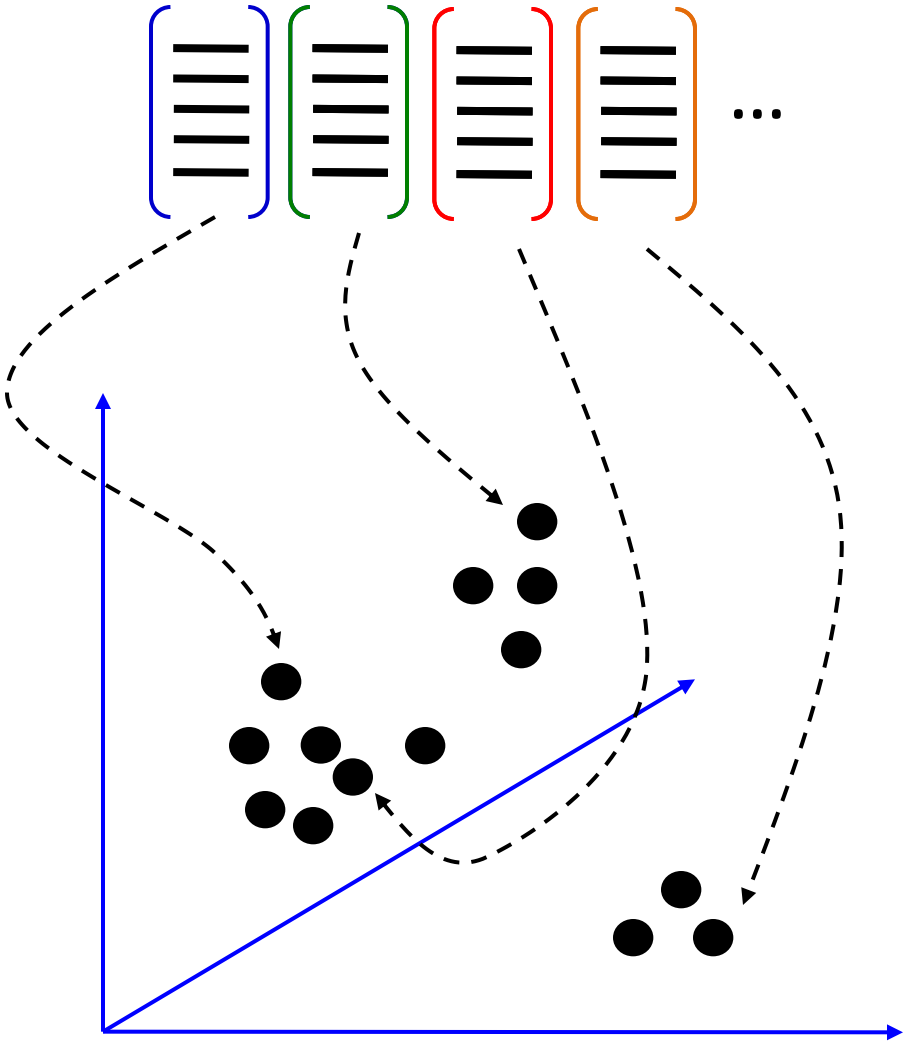
- Visualization

# HoG for other object class



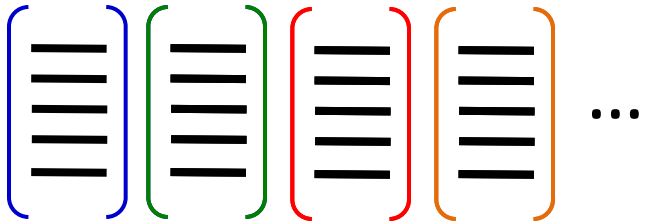


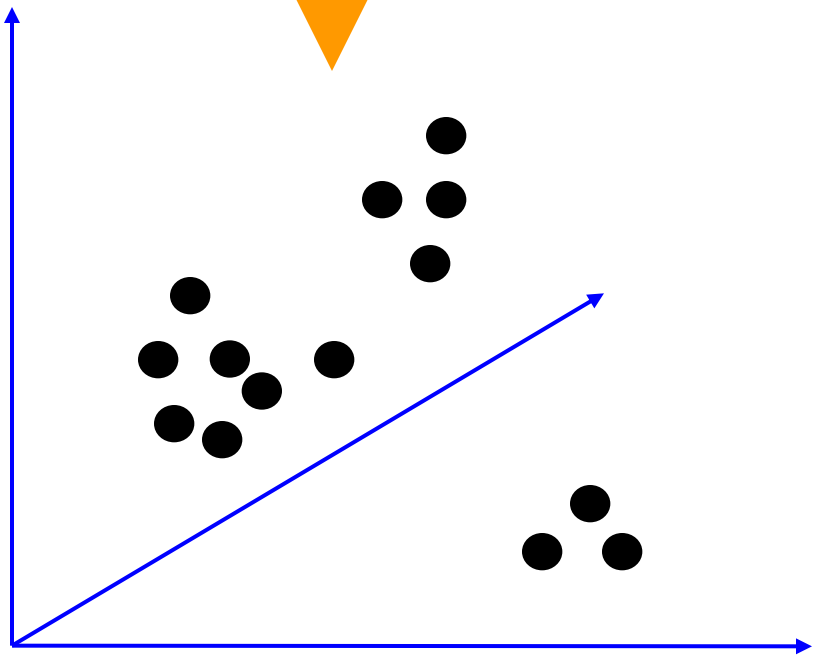
# 3. Learning the visual vocabulary

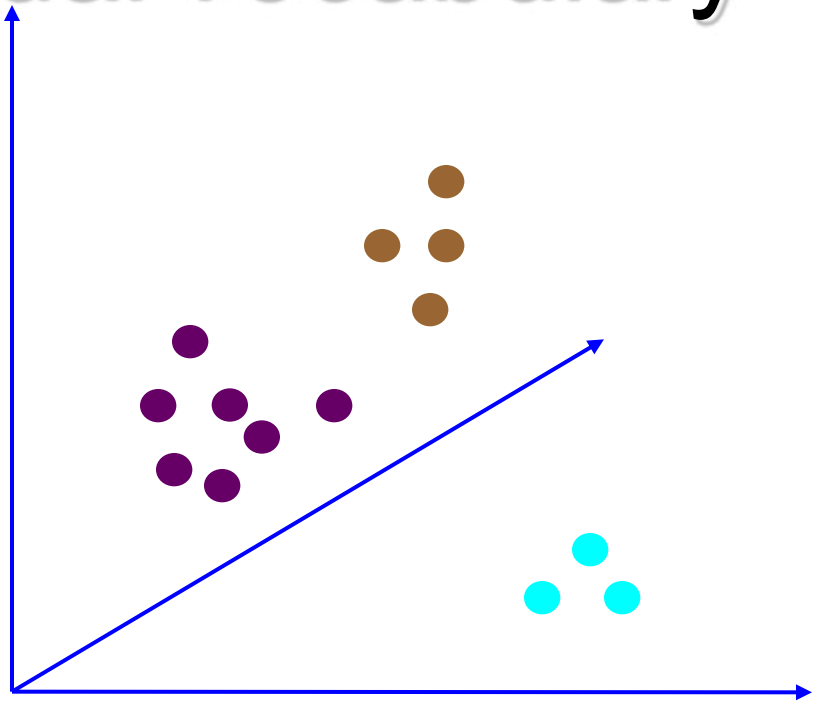


Slide adapted after J. Sivic

# 3. Learning the visual vocabulary



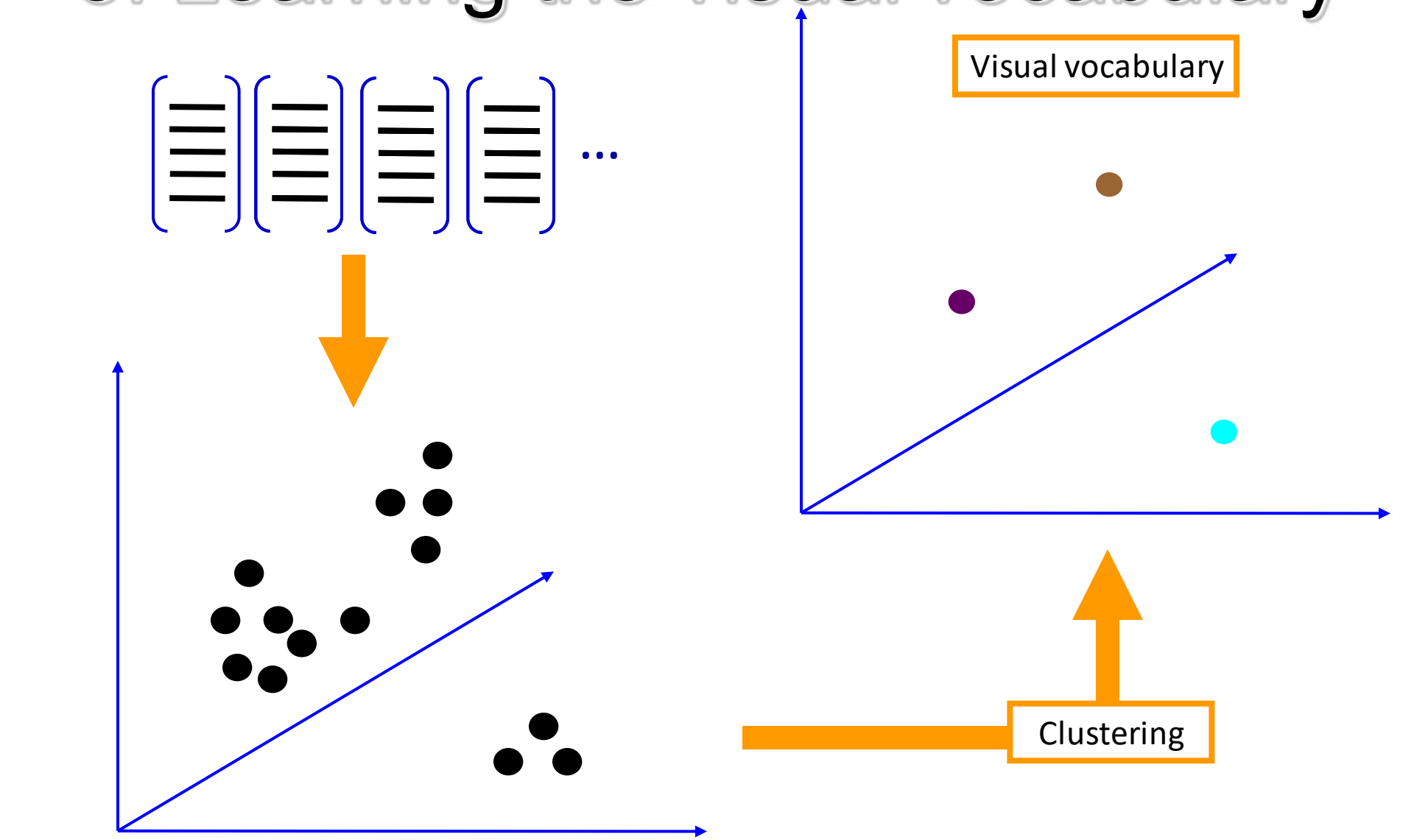




Clustering

Slide adapted after J. Sivic

# 3. Learning the visual vocabulary




Slide adapted after J. Sivic

# K-means clustering

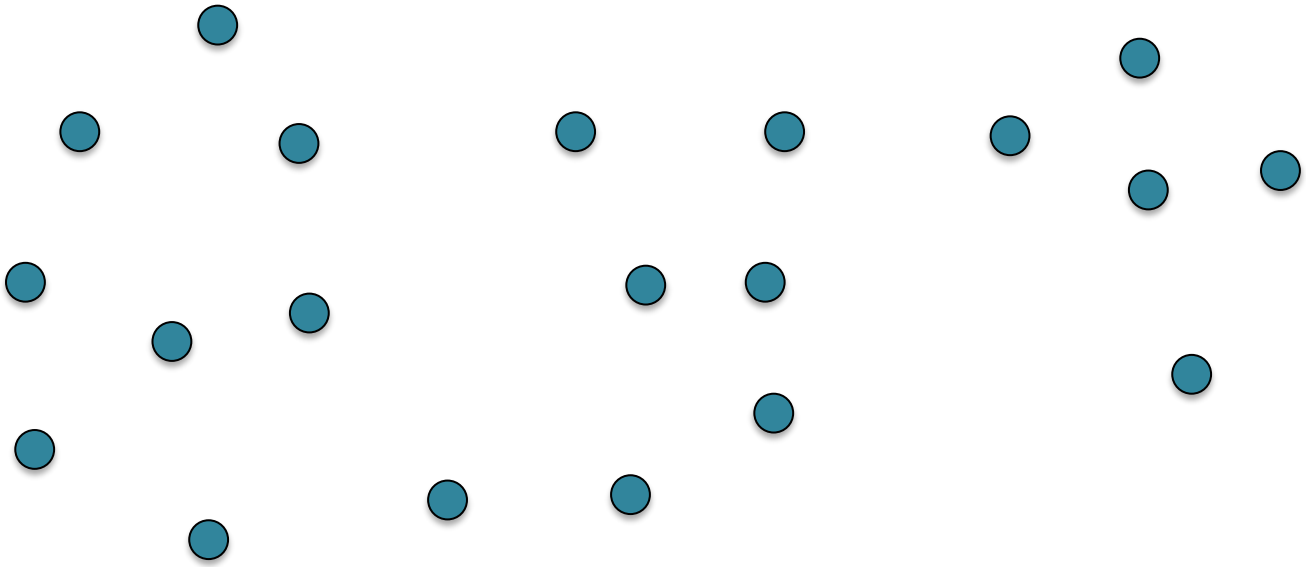
- Want to minimize the sum of Euclidean distances between feature vectors $\mathbf{x}_i$ and the nearest cluster centroids $\mathbf{m}_k$:

$$L(X, M) = \sum_{k=1}^{K} \sum_{x_i \in C_k} (x_i - m_k)^2$$

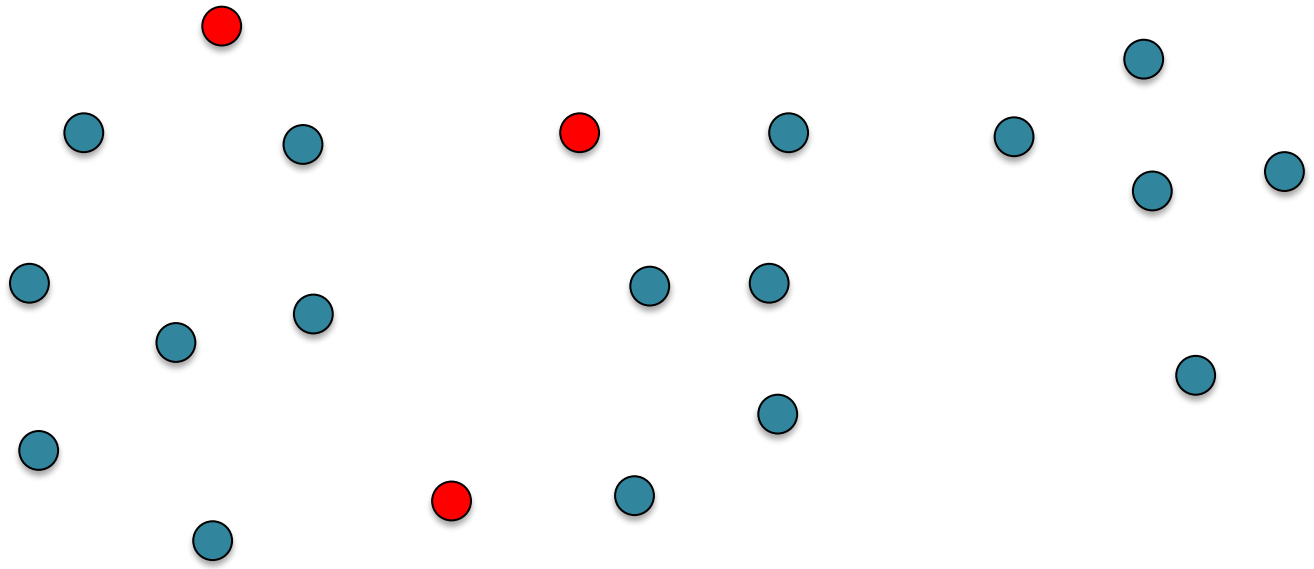
- Algorithm (Expectation-Maximization):

1. Initialize the K cluster centroids randomly
2. Iterate until clusters converge:
   - a. Label each vector based on the nearest cluster centroid
   - b. Recompute the centroid of each cluster = mean of all feature vectors assigned to a cluster

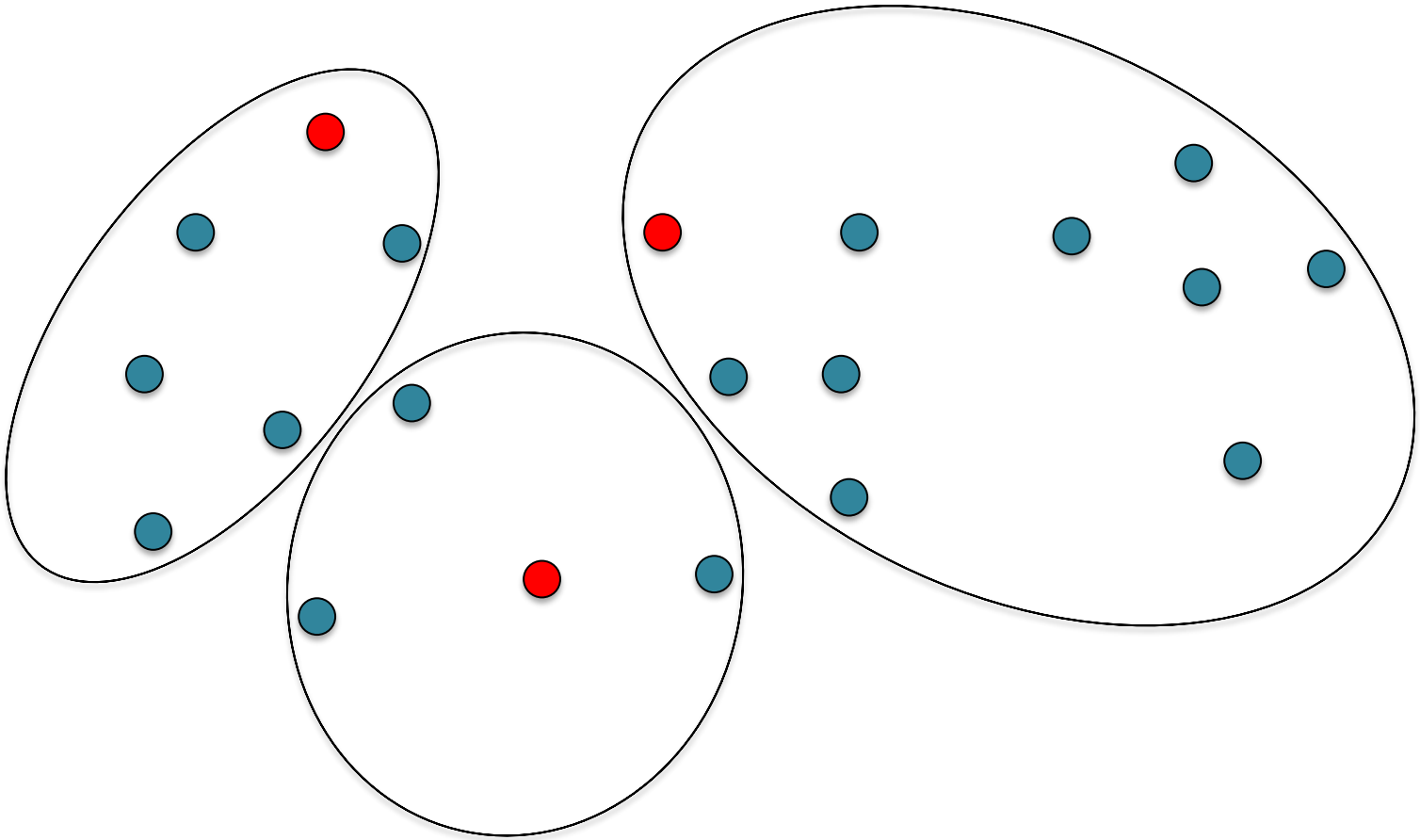
# K-means clustering – 1.



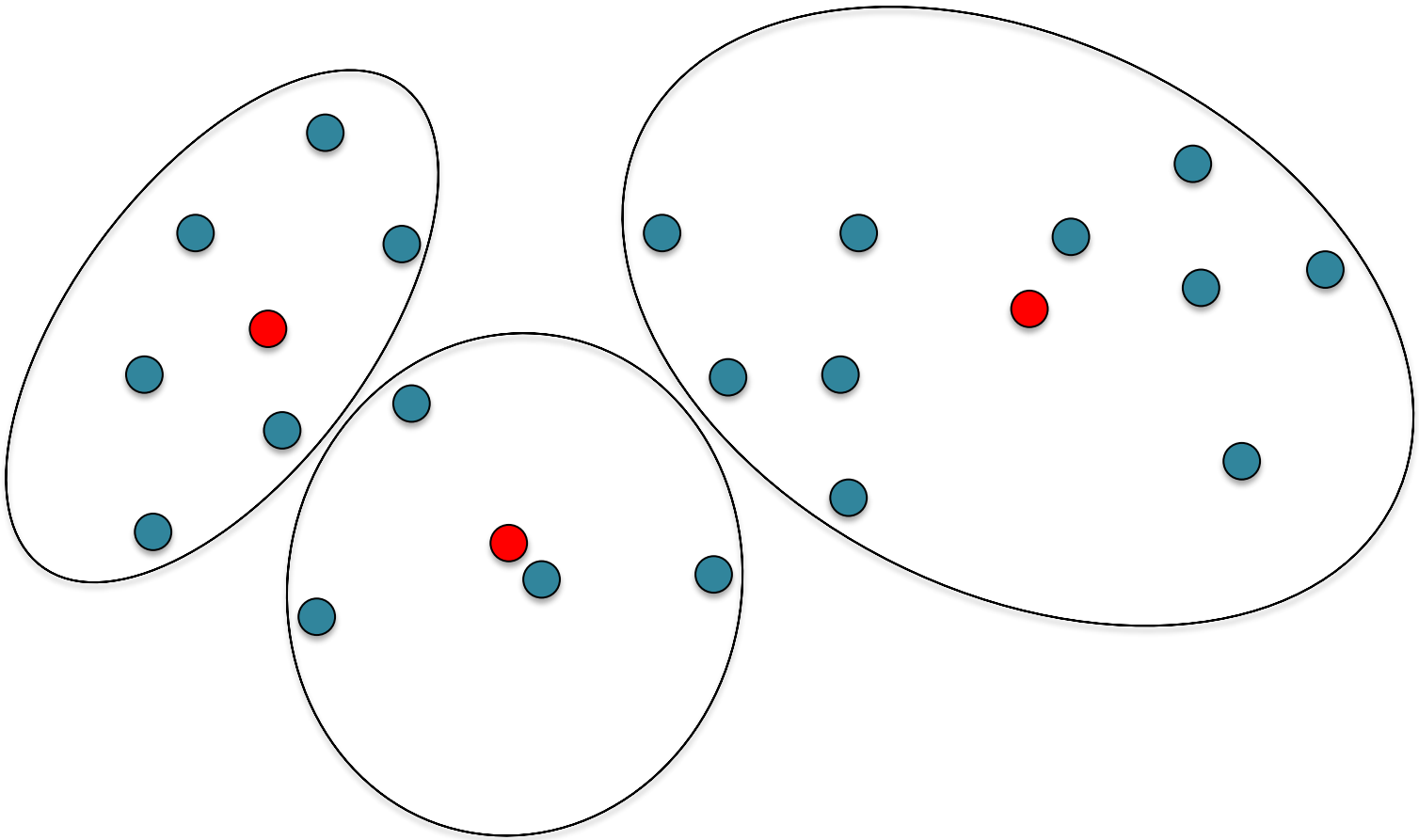
# K-means clustering – 1.



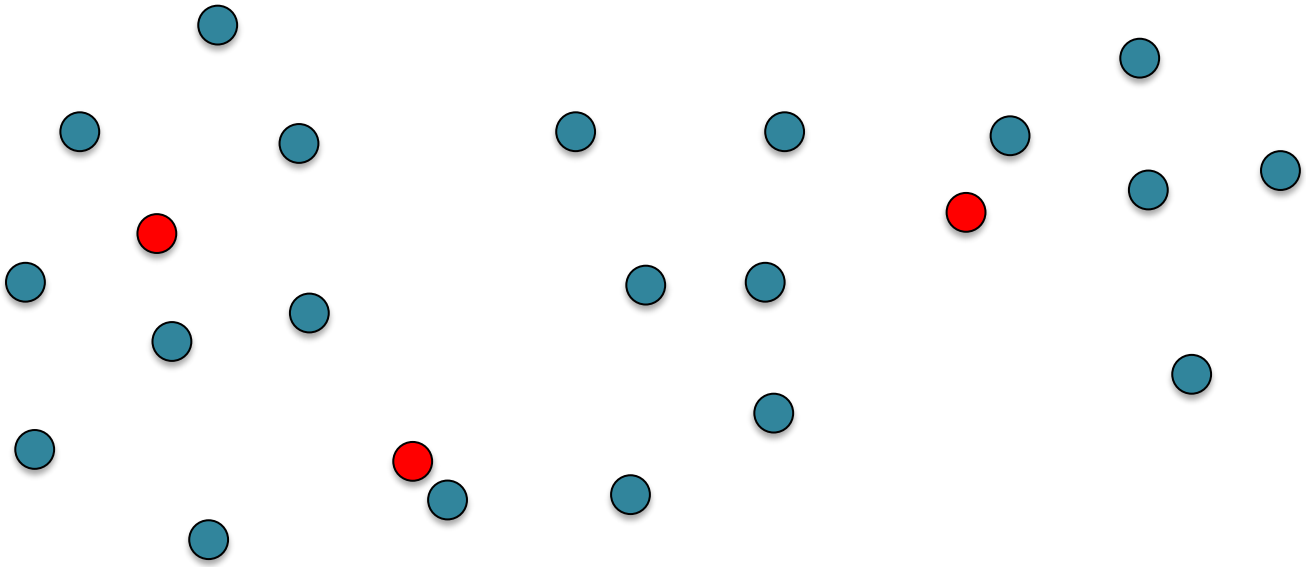
# K-means clustering – 2.a.



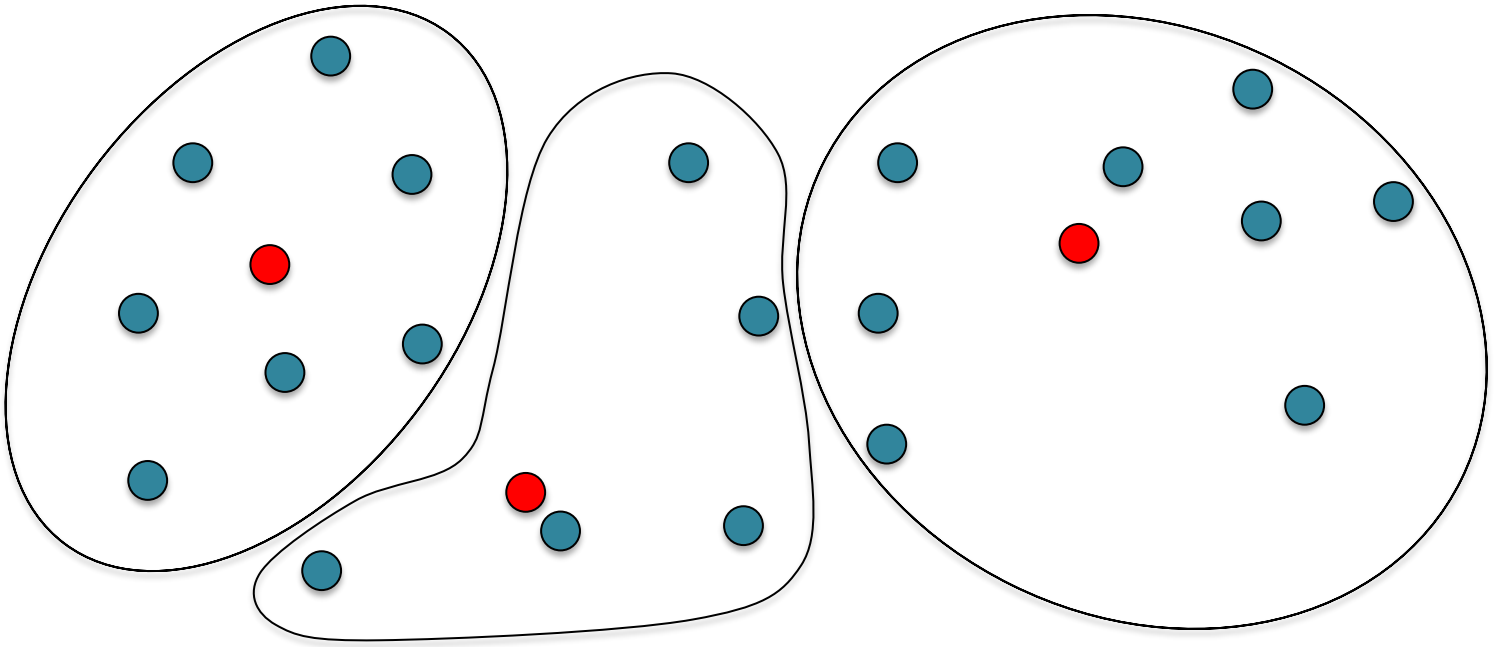
# K-means clustering – 2.b.



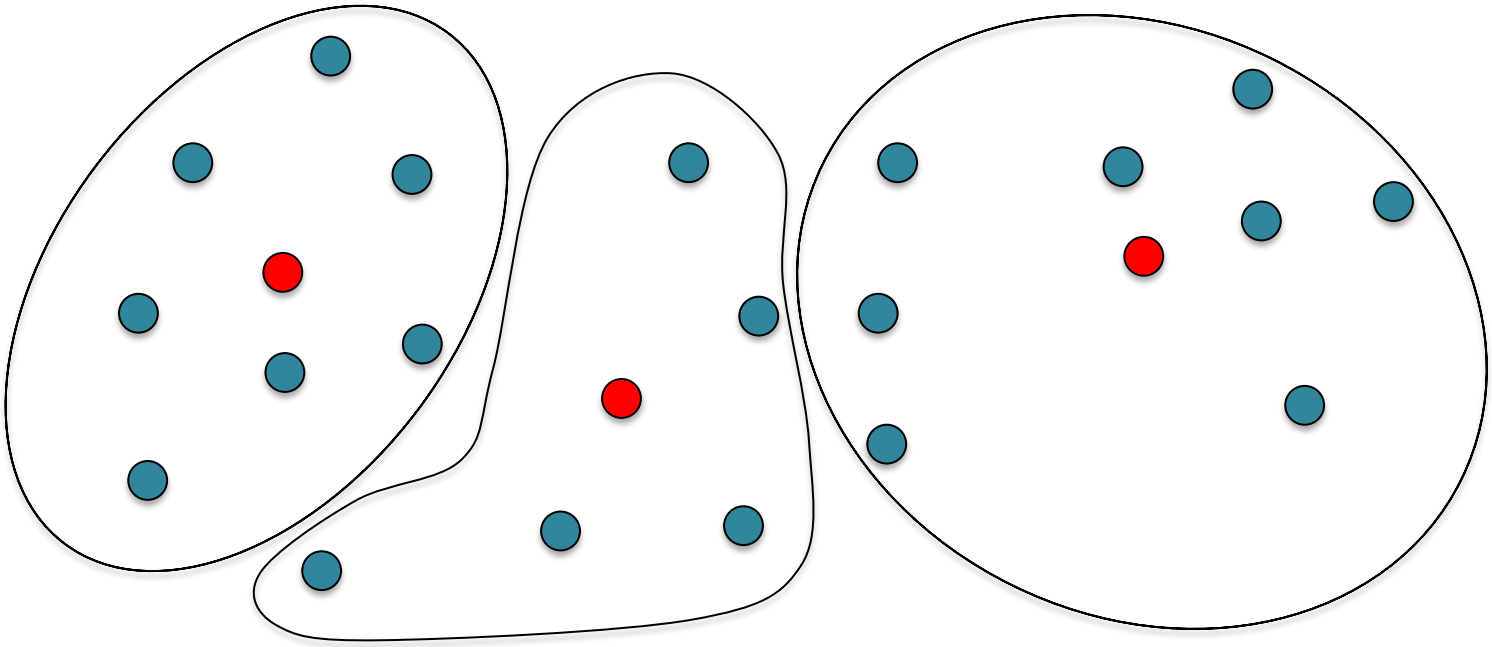
# K-means clustering – 2.b



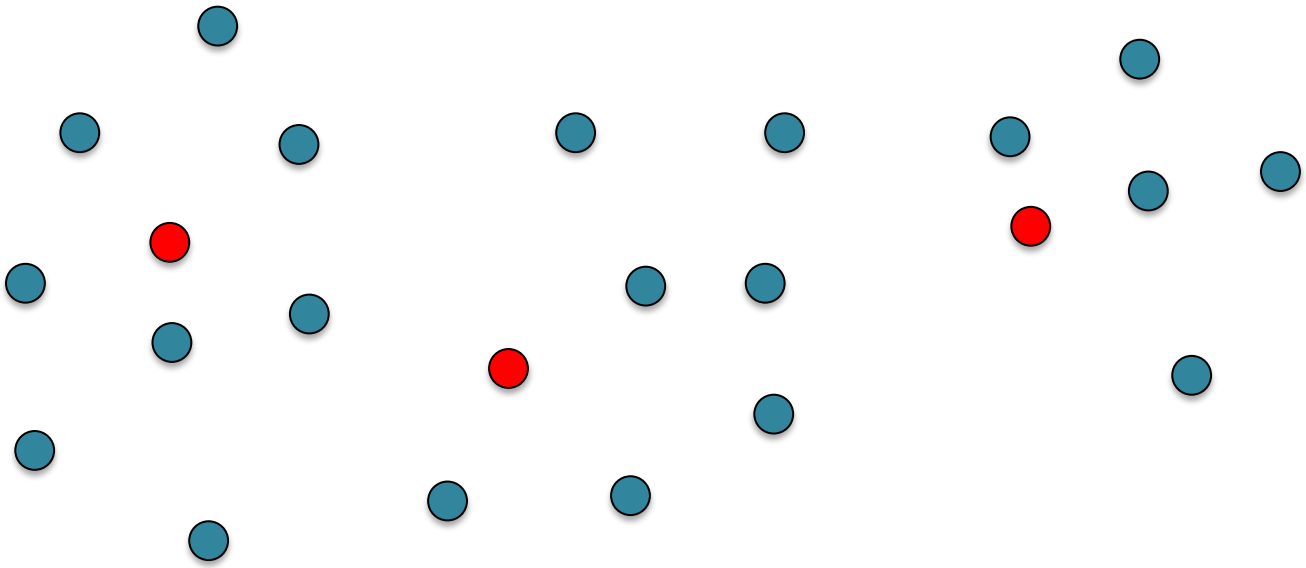
# K-means clustering – 2.a



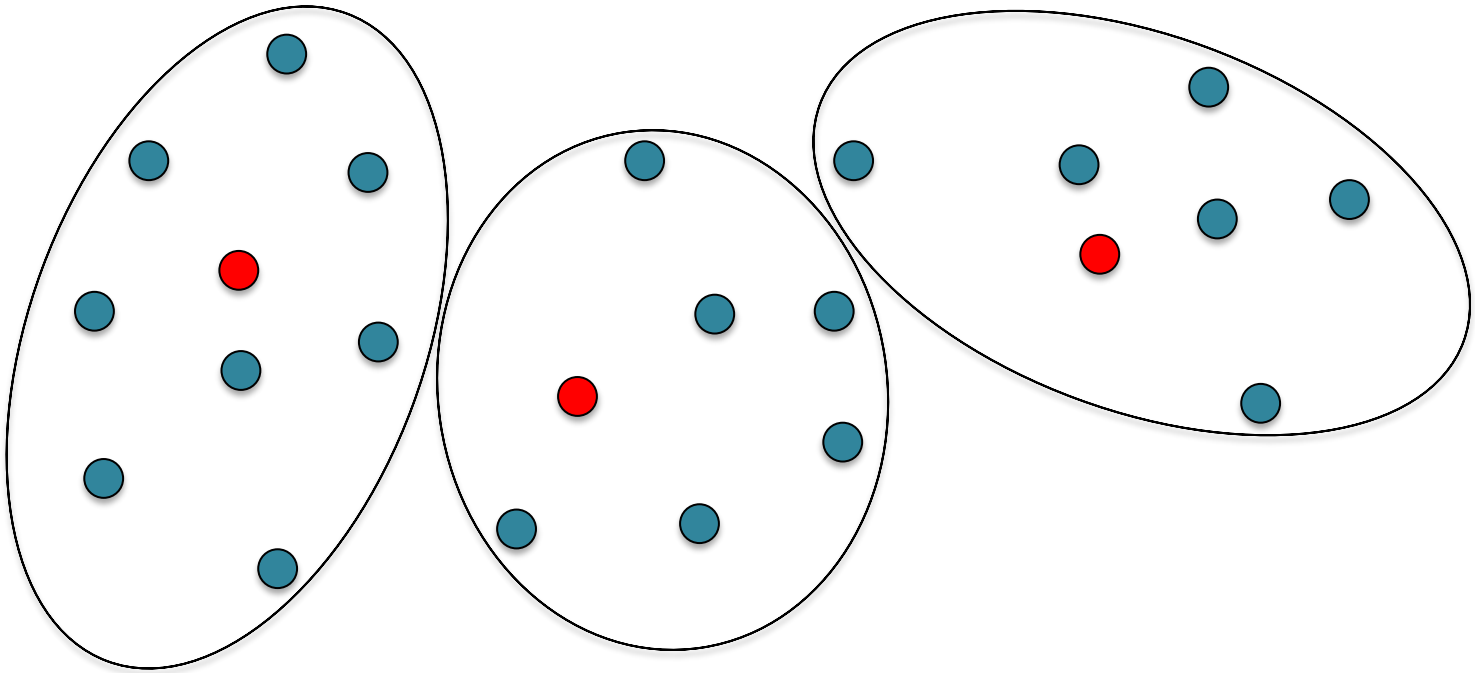
# K-means clustering – 2.b.



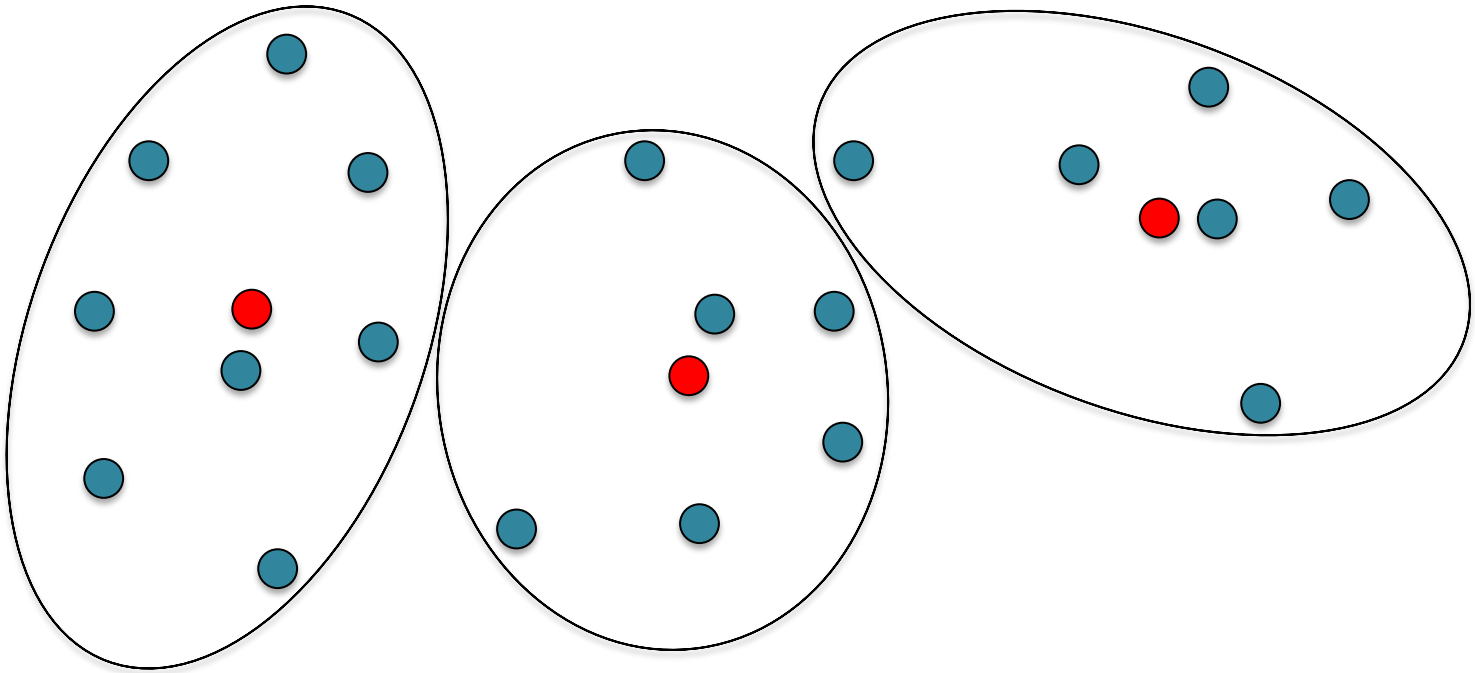
# K-means clustering – 2.b.



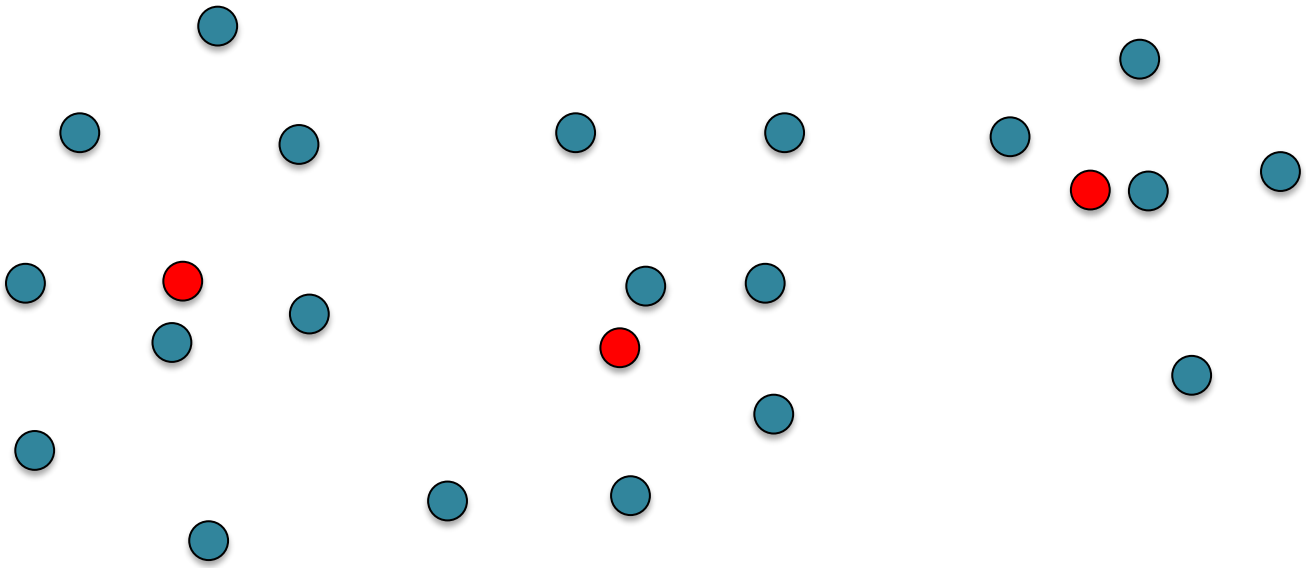
# K-means clustering – 2.a.



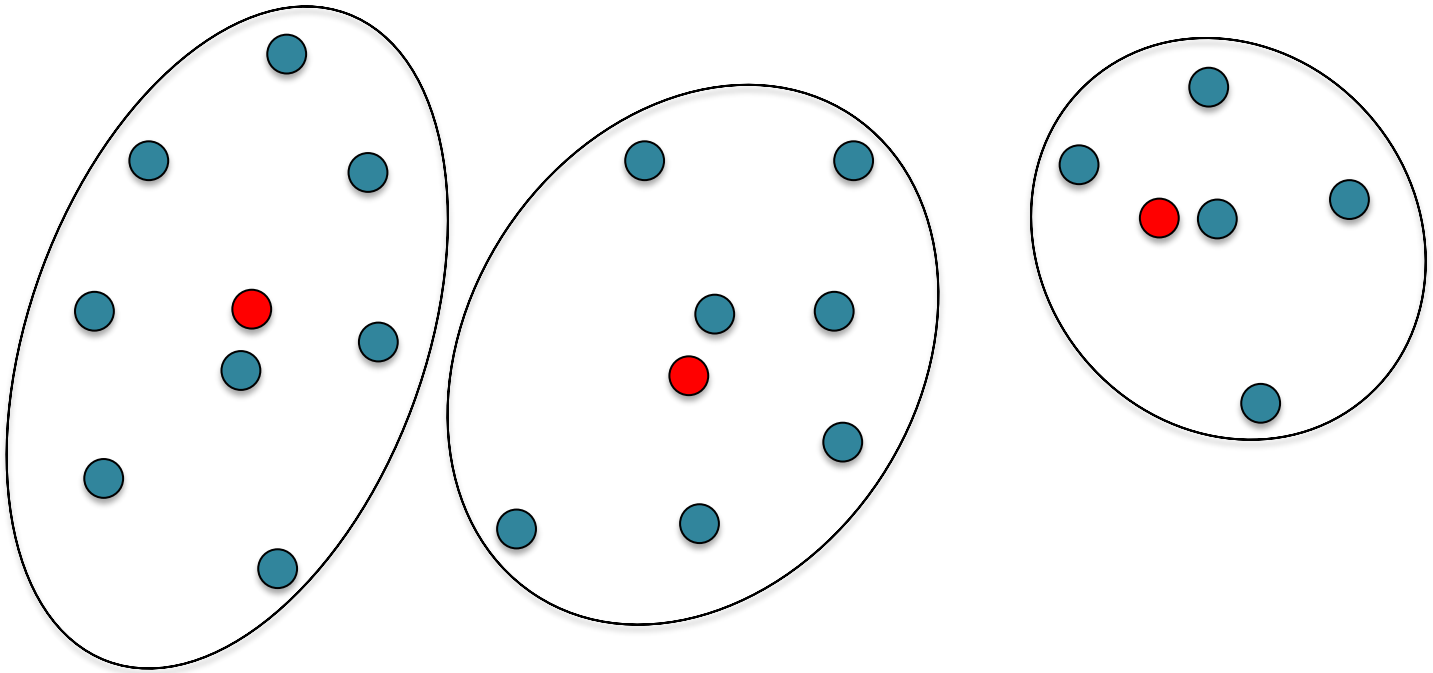
# K-means clustering – 2.b.



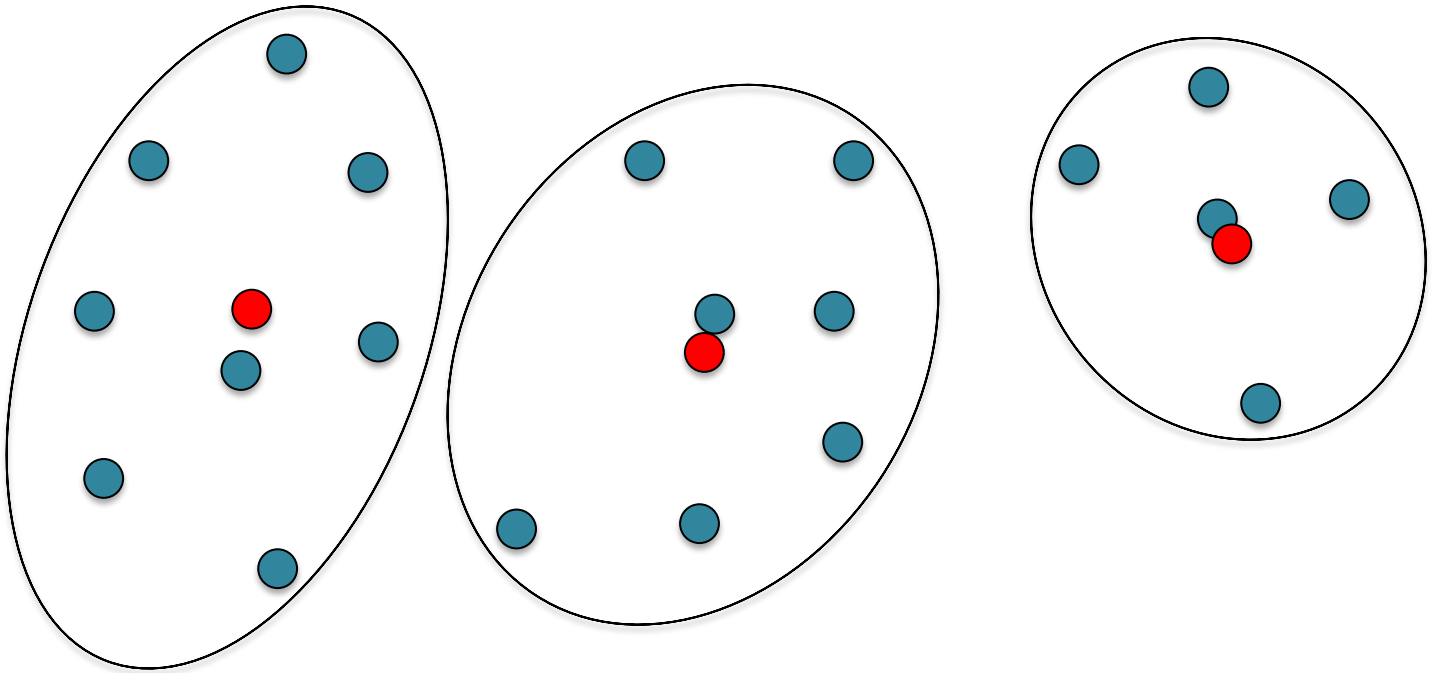
# K-means clustering – 2.b.



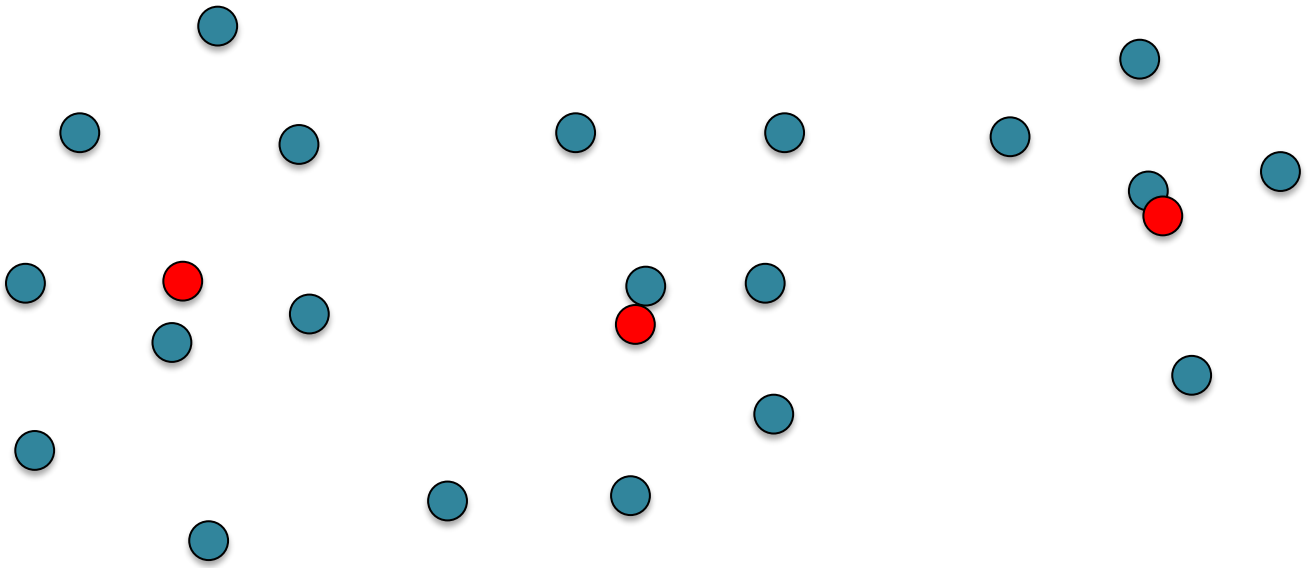
# K-means clustering – 2.a.



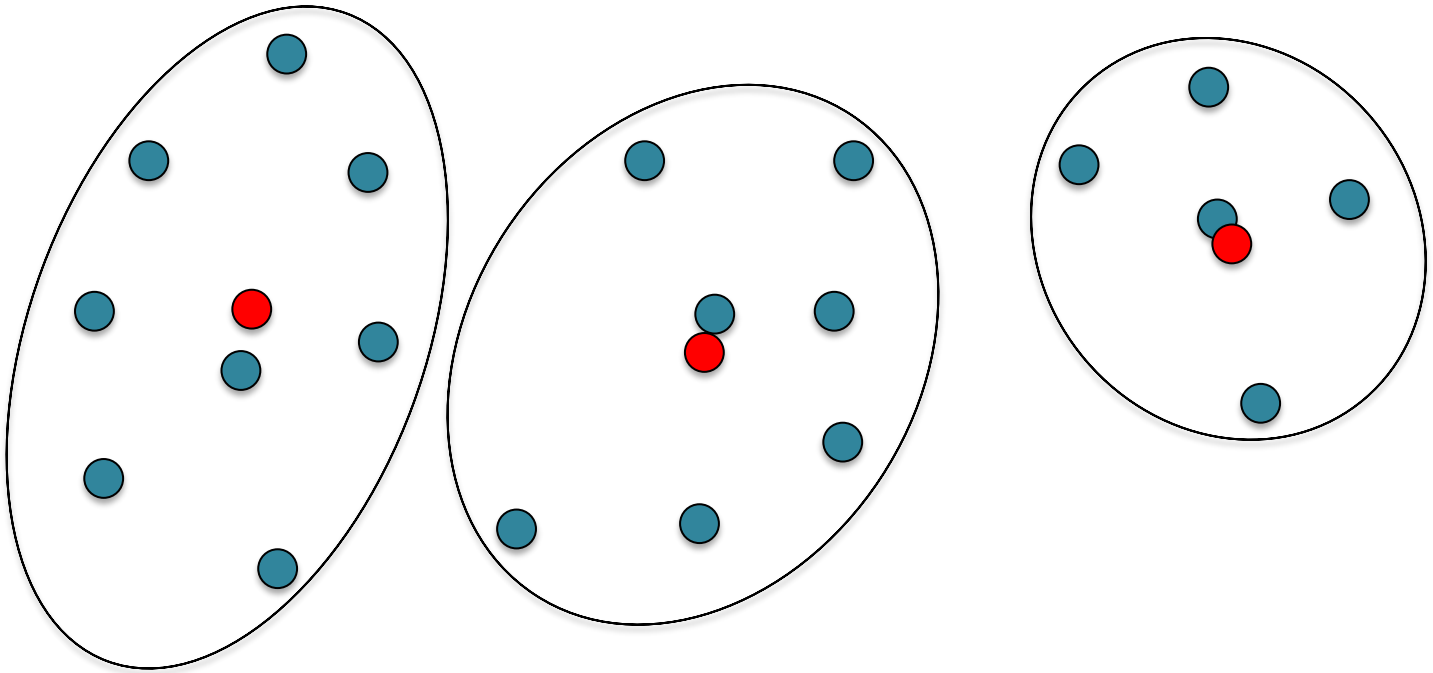
# K-means clustering – 2.b.



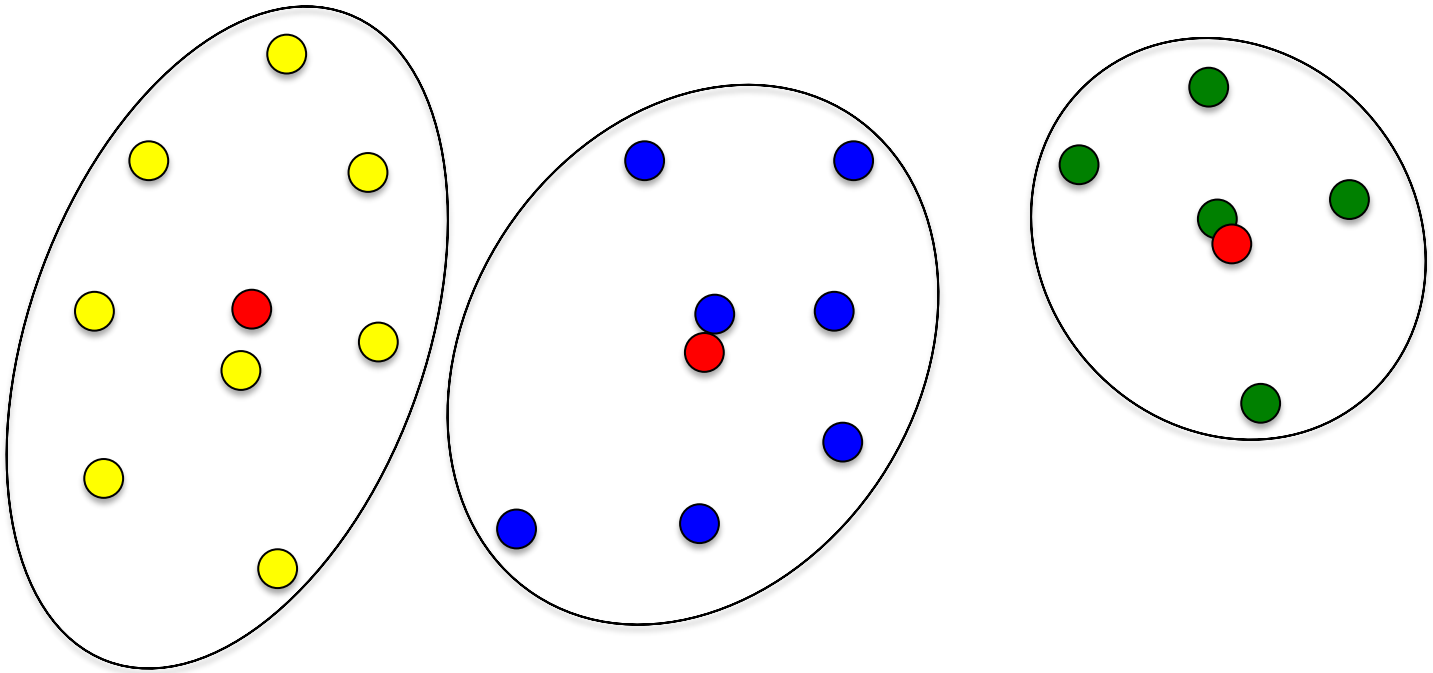
# K-means clustering – 2.b.



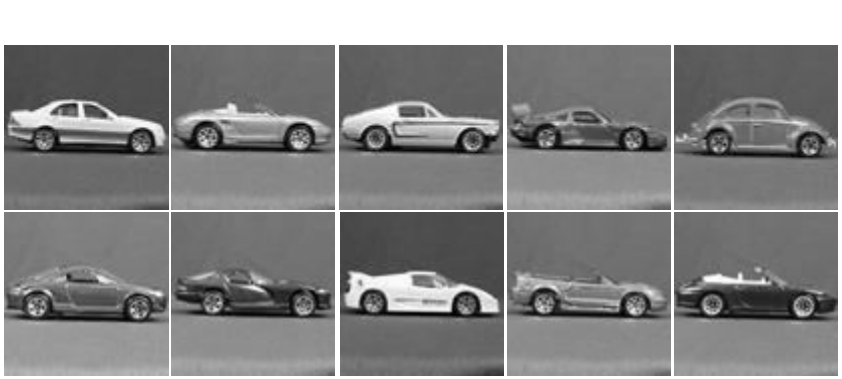
# K-means clustering – 2.a.

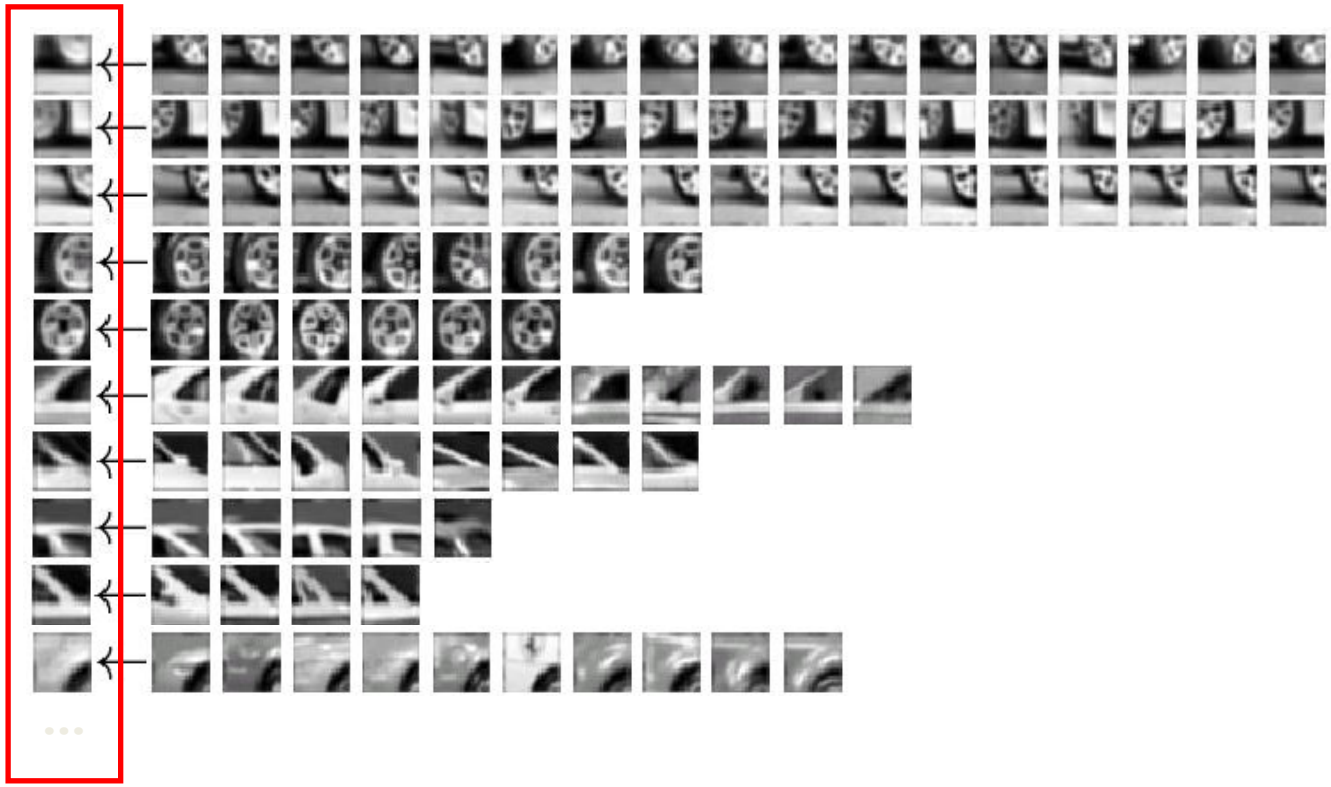


# K-means clustering - Output

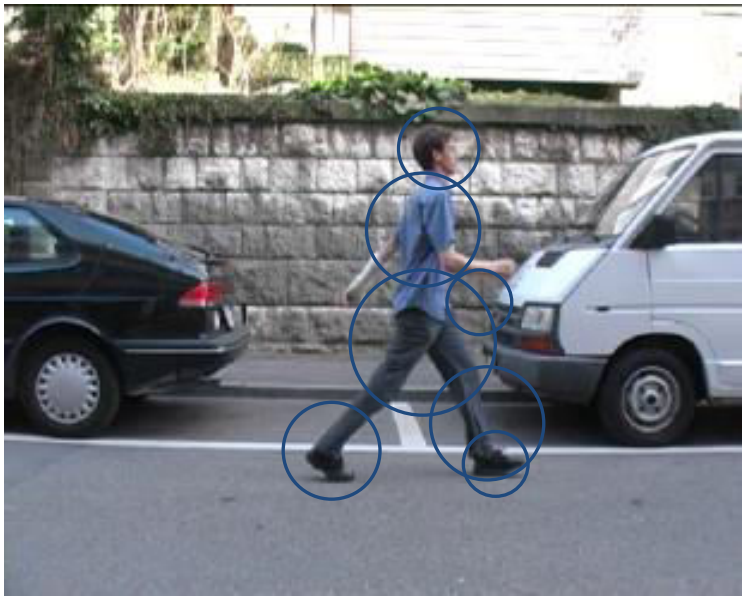


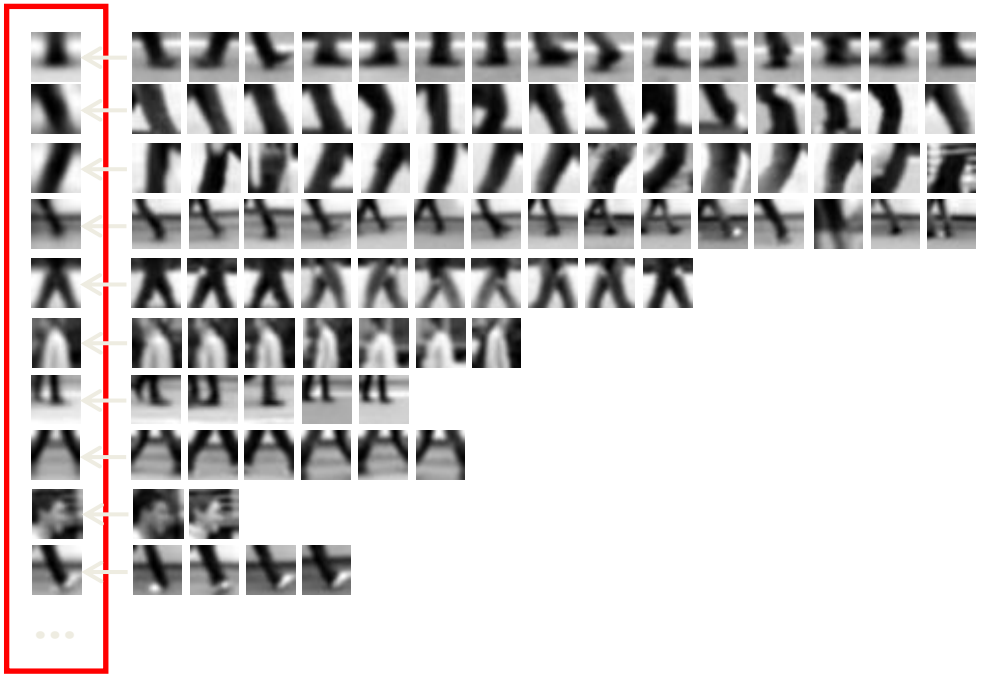
# Visual word vocabulary





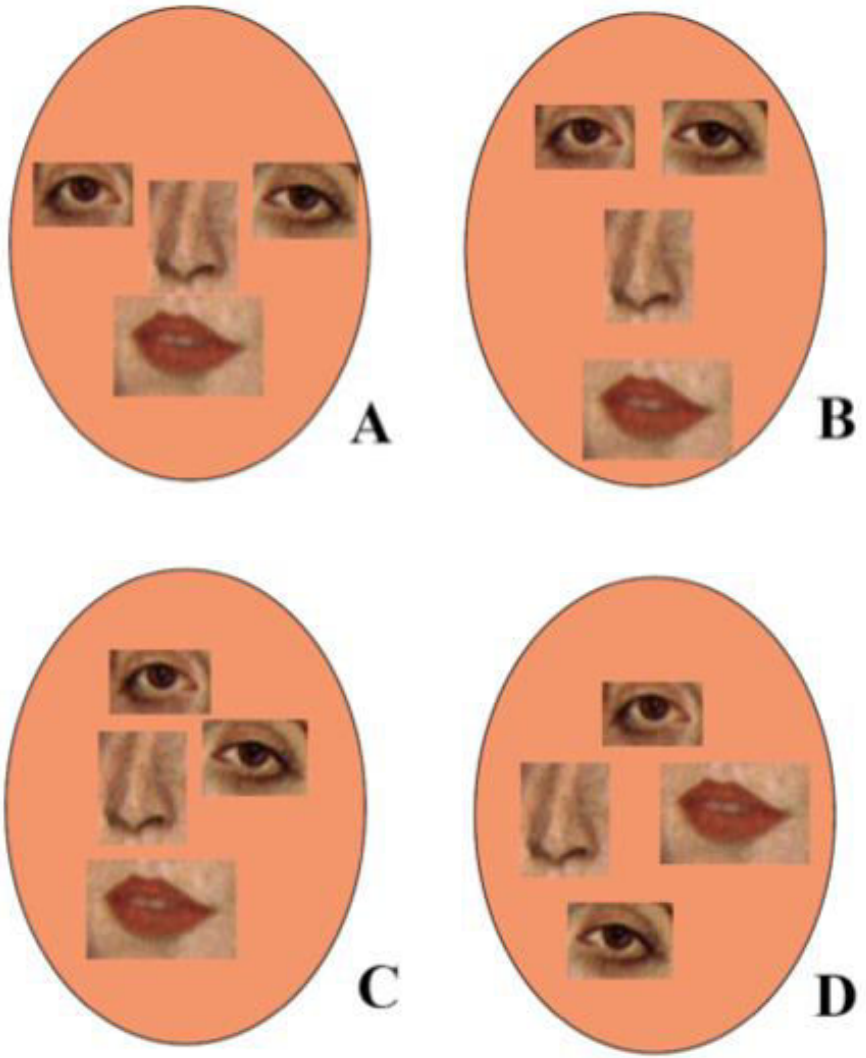
# Visual word vocabulary





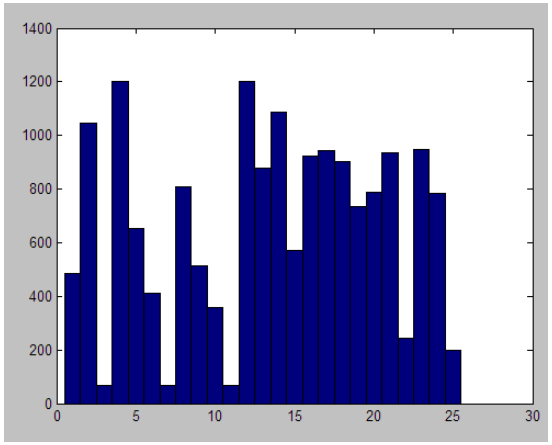
# Limitations of bag-of-visual-words

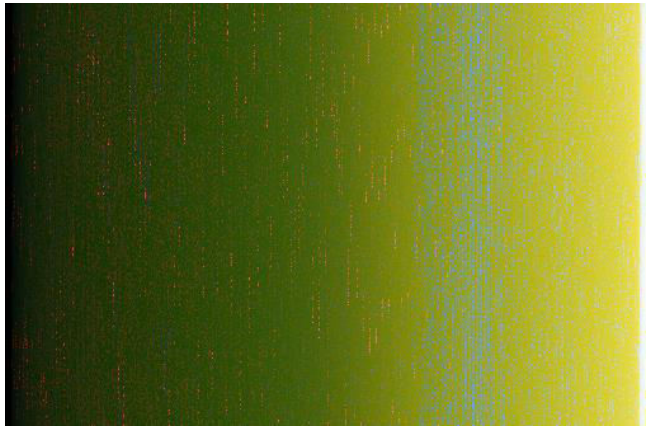
- The representation does not take into account the location of words in the image
- Advantages?
- Problems?




Slide adapted after V. Ferrari

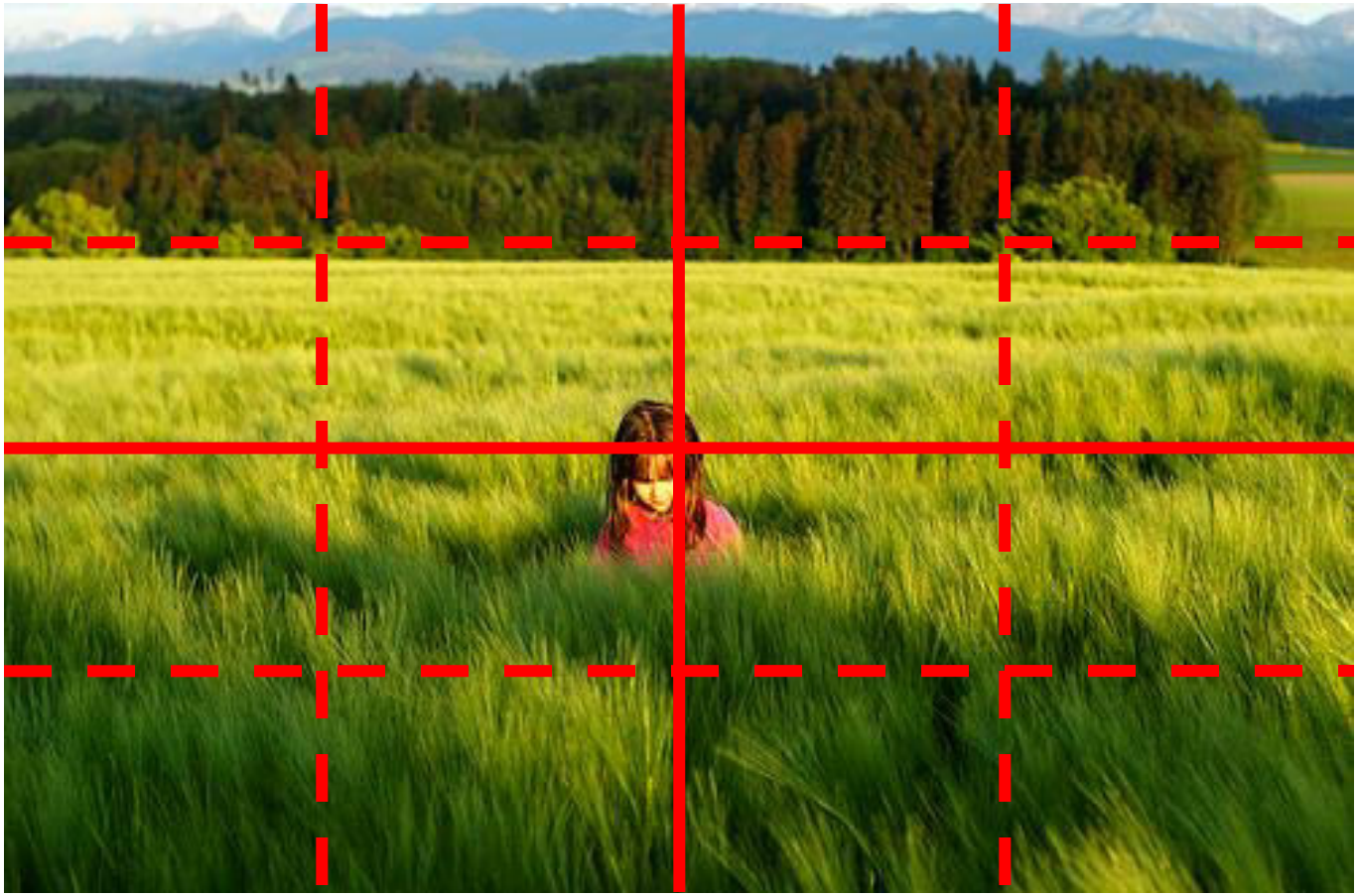
# Location of features in the image



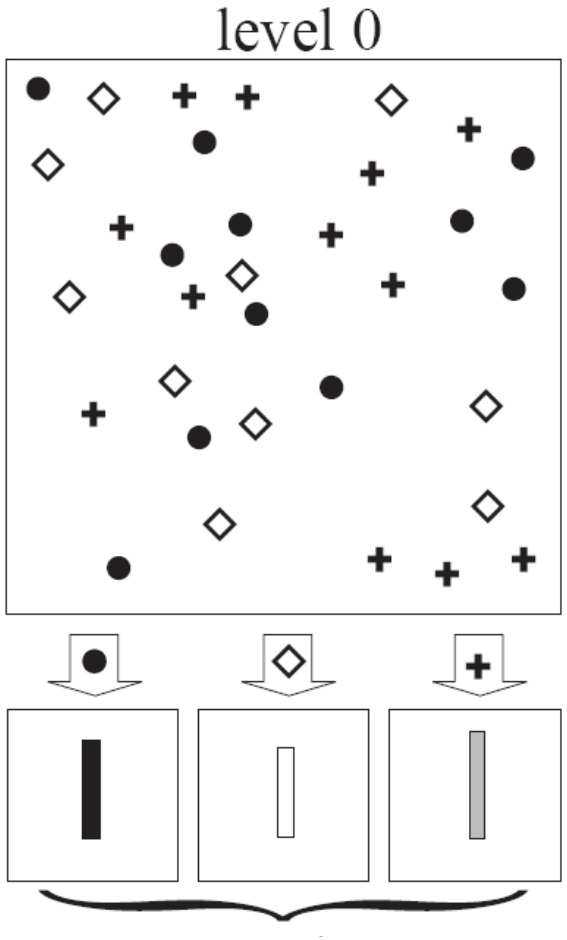
These 3 images have the same histogram

# Spatial pyramid

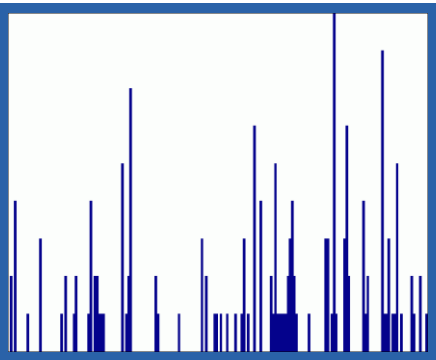


Compute histogram for each sub-region (bin)

# Spatial pyramid



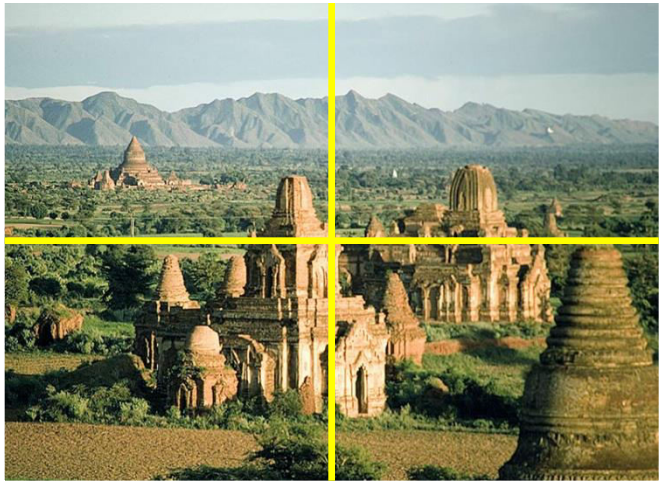
# Spatial pyramid

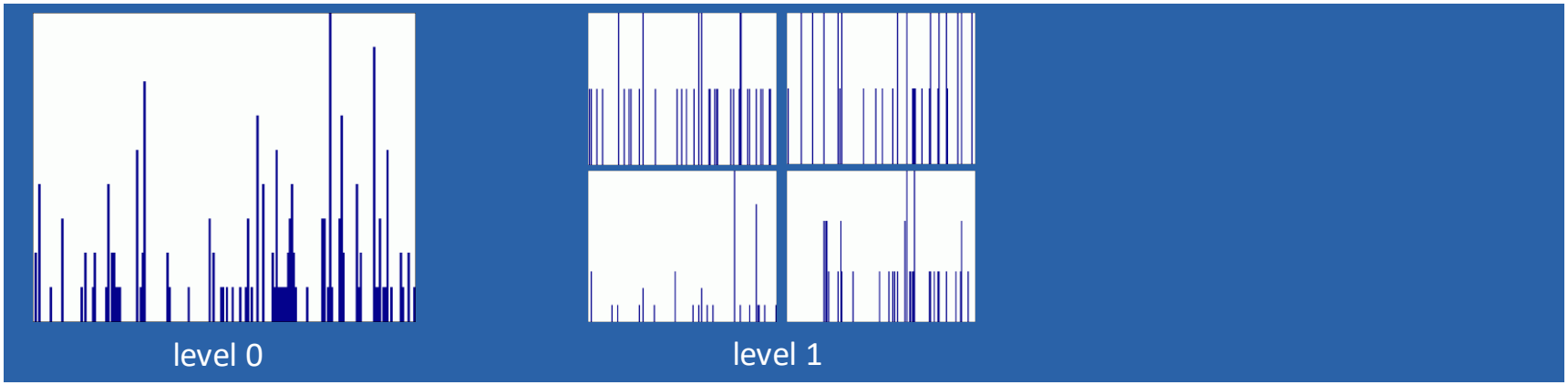


level 0

Slide adapted after S. Lazebnik

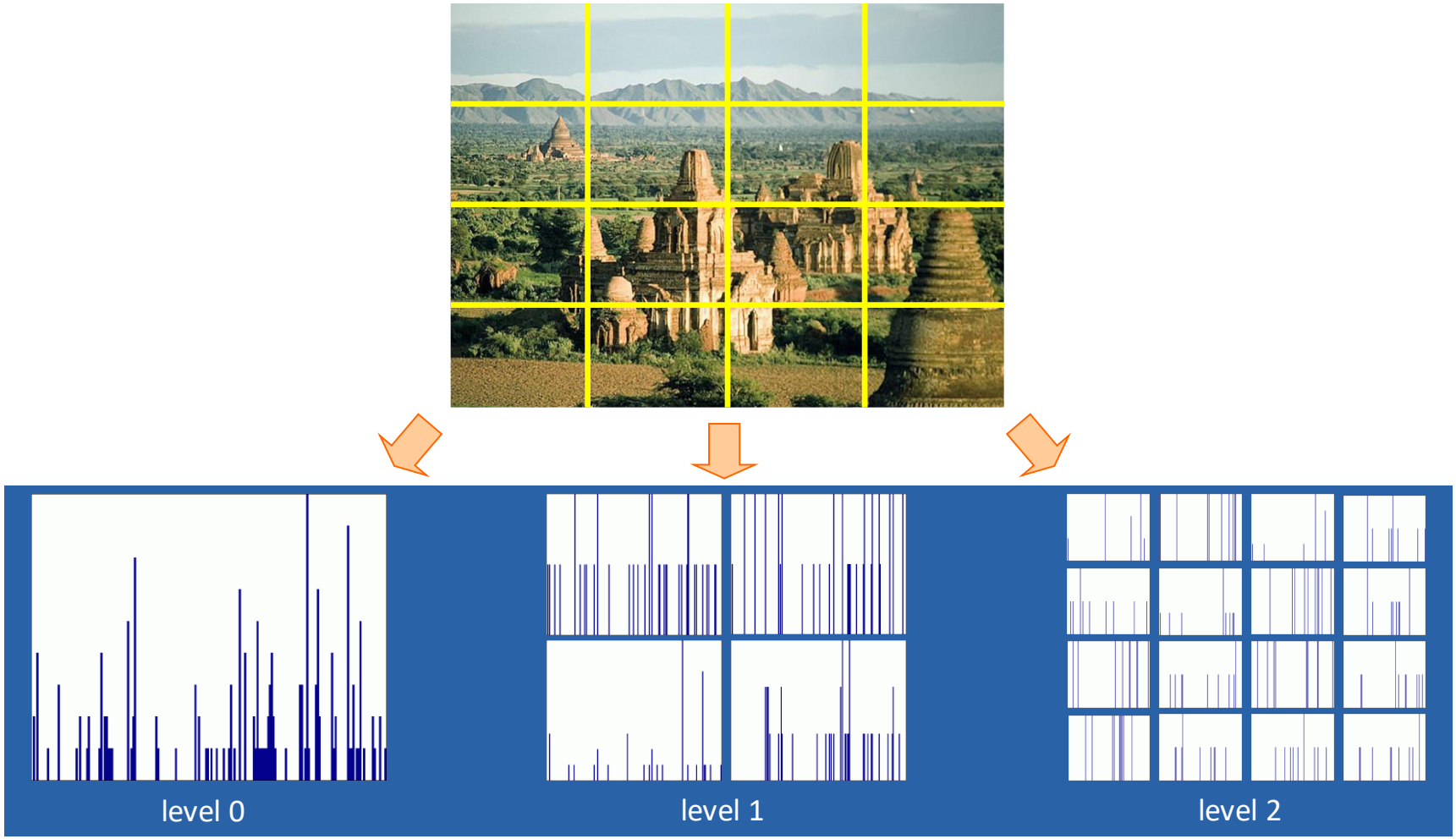
# Spatial pyramid



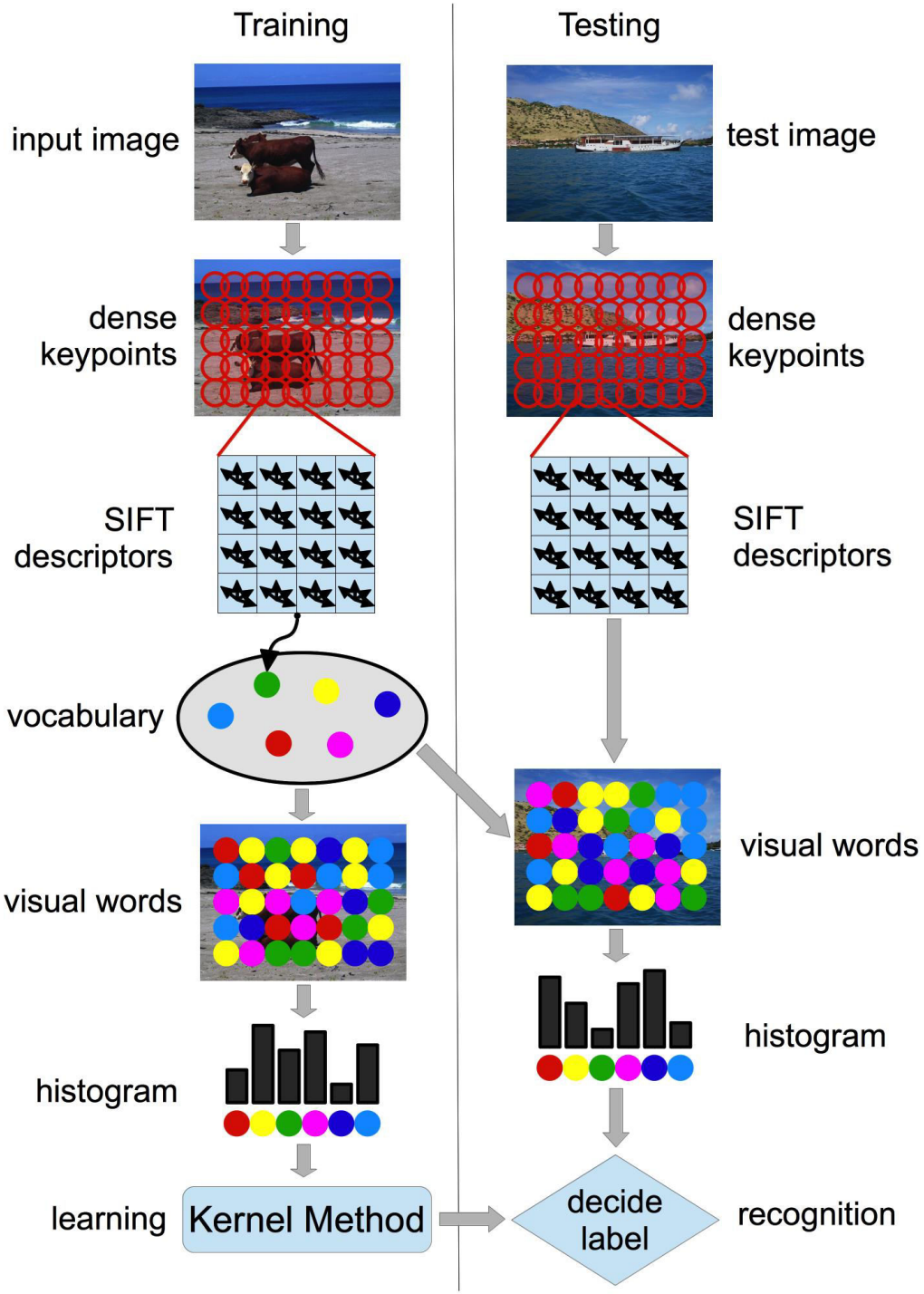



Slide adapted after S. Lazebnik

# Spatial pyramid




Slide adapted after S. Lazebnik


Training
Testing
input image
test image
dense keypoints
dense keypoints
SIFT descriptors
SIFT descriptors
vocabulary
visual words
visual words
histogram
histogram
learning
Kernel Method
decide label
recognition