# Computer Vision

Pattern Recognition for Computer Vision

Luis F. Teixeira
MAP-i 2014/15

# Goal of Computer Vision

- Provide computers with human-like **perception** capabilities so that they can sense the environment, **understand** the sensed data, take appropriate actions (**make decisions**), learn from this experience in order to enhance future performance
  - **Understand visual information** with no accompanying structural, administrative or descriptive text information

- Sources of difficulties:
  - Sensory gap
  - Semantic gap

# Why is Vision hard?

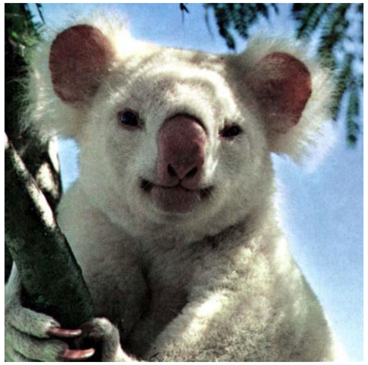
# Challenges

Illumination

Object pose

Clutter

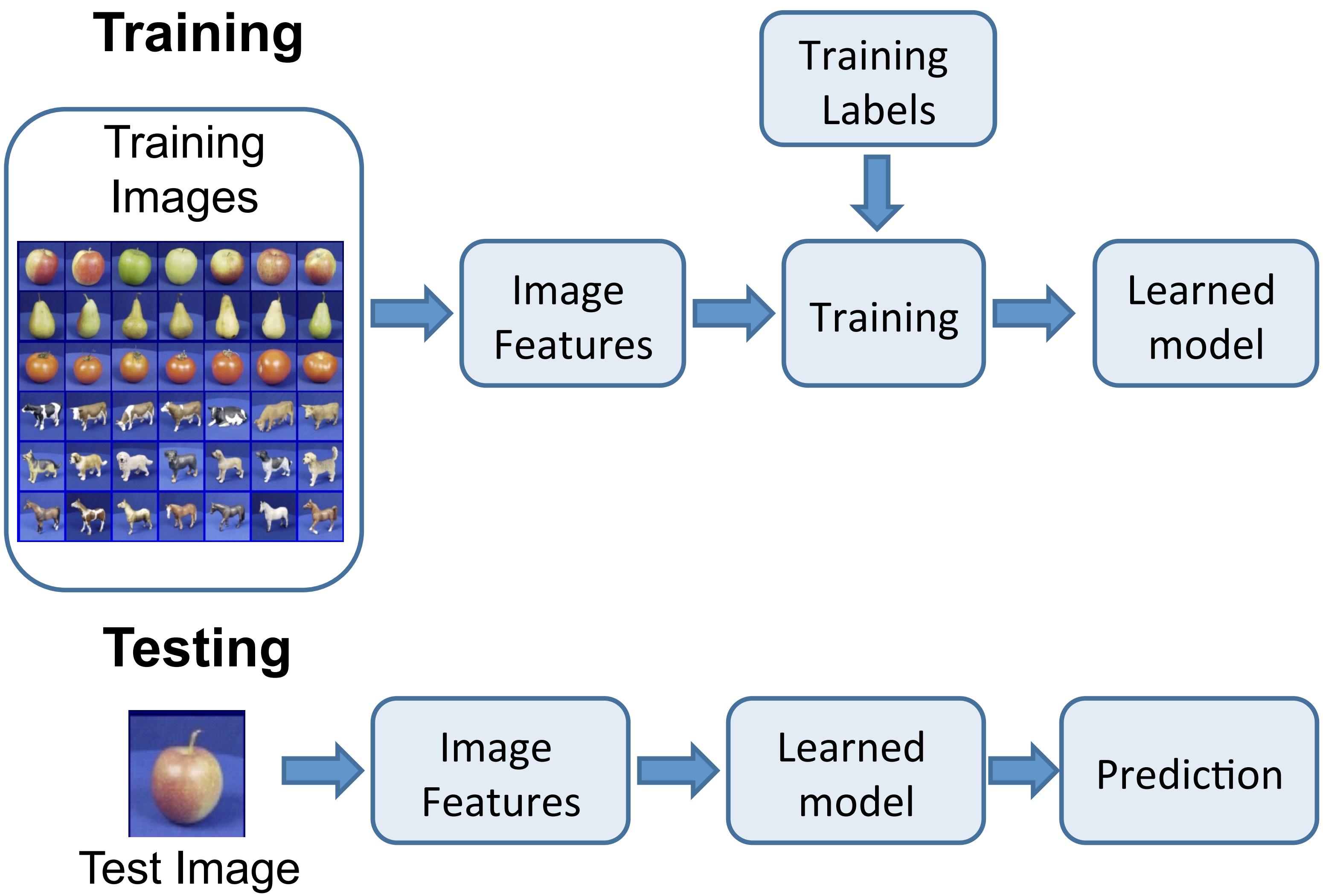
Occlusions



Intra-class appearance

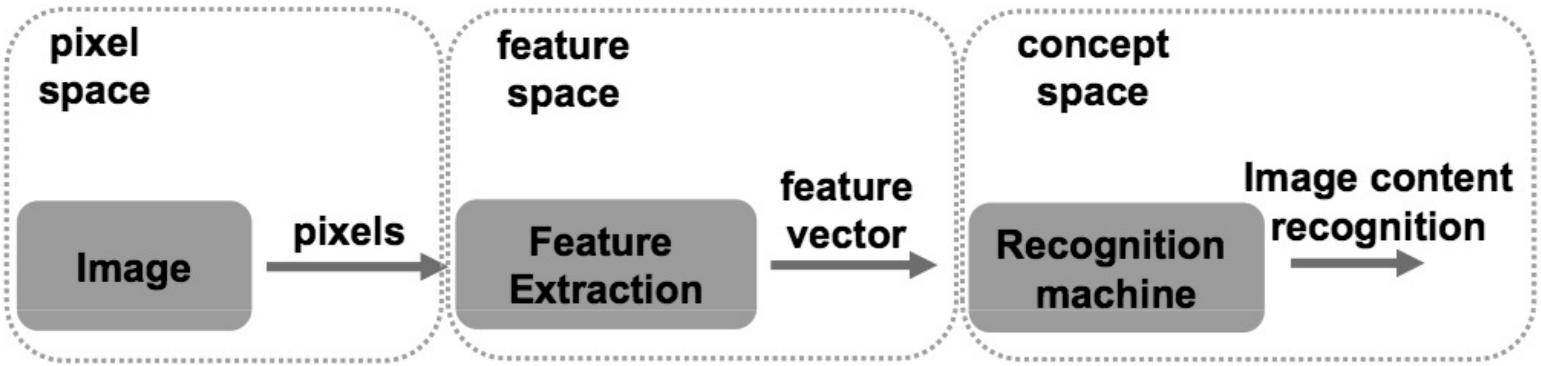
Viewpoint

# Pattern recognition in computer vision



# Pattern recognition in computer vision

- Image recognition system:
  - **Feature extraction**: captures meaningful information from the image (for the specific task at hand), reducing dimensionality.
  - **Pattern recognition**: does the actual job of classifying or describing observations, relying on the extracted features.
- System diagram




  - How can we find meaningful features?

# Features

- Raw pixels
  - Use directly the color values captured by the sensor
- Low level features
  - These features are very objective features
- Middle level features
  - Features resulting from a decision process (related to the existence of some subjective details)
    - Segmentation of certain shapes
    - Identification of certain objects, types of content
- High level features
  - Features with some semantic content information, highly contextual and based on prior knowledge.
    - Person A is talking to person B

# Features

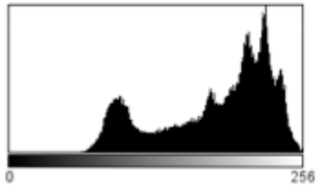
- Types of features
  - Low-level: Color, texture, shape, motion, ...
  - Middle-level: Pedestrian in the image, visible sky, existence of trees
  - High-level: Car moving fast, person smiling
- From low-level to high-level
  - While decisions must be made at each level we must always **start from the low-level**, as that is the information readily available to us.
  - The fundamental problem is **how to reach high-level knowledge** from initial low-level features.

# Features

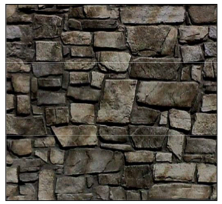
- Features can also be classified based on extent: **Global**, **Region** or **Local**
  - Global features:
    - These features highly summarize the image content enabling good description of global content or context but missing fine detail.
    - These can also be used at a semi-global level by subdividing the image into regions.
  - Region features:
    - These features describe boundary-based properties of an object or they can describe region-based properties.

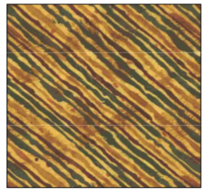
# Features

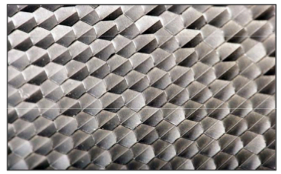
- Classic features
  - Global **colour** and **edge** histograms




  - **Texture** through co-occurrence matrices and fractal analysis







  - **Shape** through measures of area, perimeter, eccentricity, orientation, etc.
    - Very high computational cost
    - Features are very complex
- Other features such as the ones described in the MPEG-7 standard can also be useful

# Features

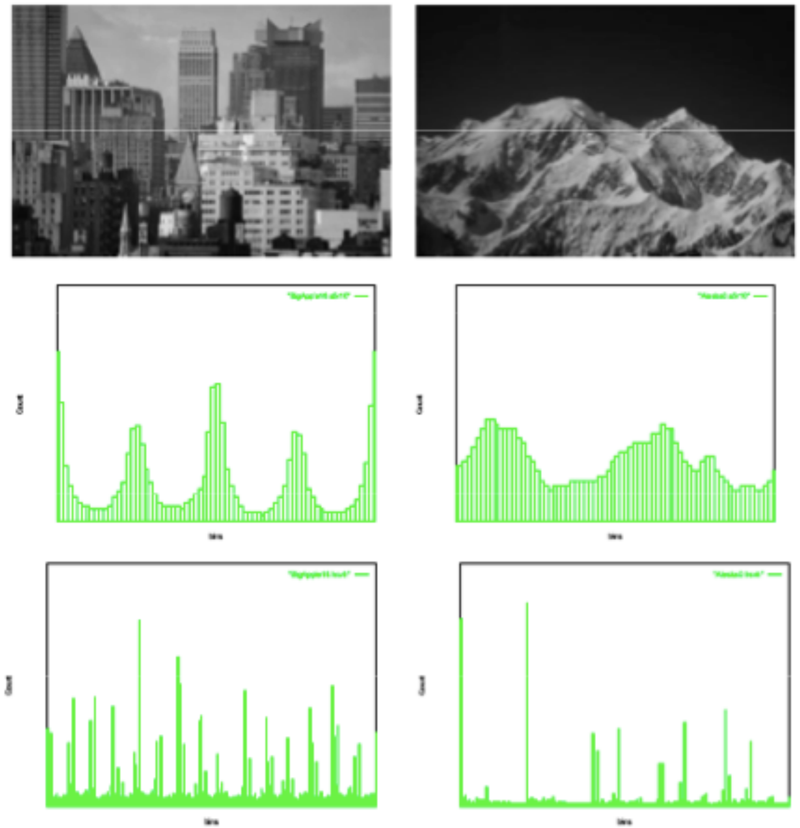
- The best feature to use for an image's description depends on what is its content:
  - For detecting different objects, different features may be required.
  - We do not know the content (we are trying to find it).
- Features can be **combined** by concatenation into a larger feature vector:
  - However, the features may have different "importance" for the image recognition system.

$$F_{fusion} = \alpha \times F_1 + (1-\alpha) \times F_2$$

  - α is the fusion weighting, an additional hyper-parameter in the system which must be validated experimentally.

# Features

- Example: using global features to classify city/landscape images
  - Based on colour and edge histograms
  - KNN classifier
  - Features fusion using weighted concatenation



**On Image Classification: City vs. Landscape**. Vailaya, A. and Jain, A. and Zhang, H. IEEE Workshop on Content - Based Access of Image and Video Libraries,1998

# Features

- Global features rarely have the descriptive power to capture all information in an image
- This leaves global features usable only for some limited image recognition tasks
- An image often requires a **part based analysis**
  - Context is global, but object are defined locally.
  - Most image content is described at a local level.
  - By dividing an image into parts we simplify recognition.
  - Separating objects from context makes recognition more robust

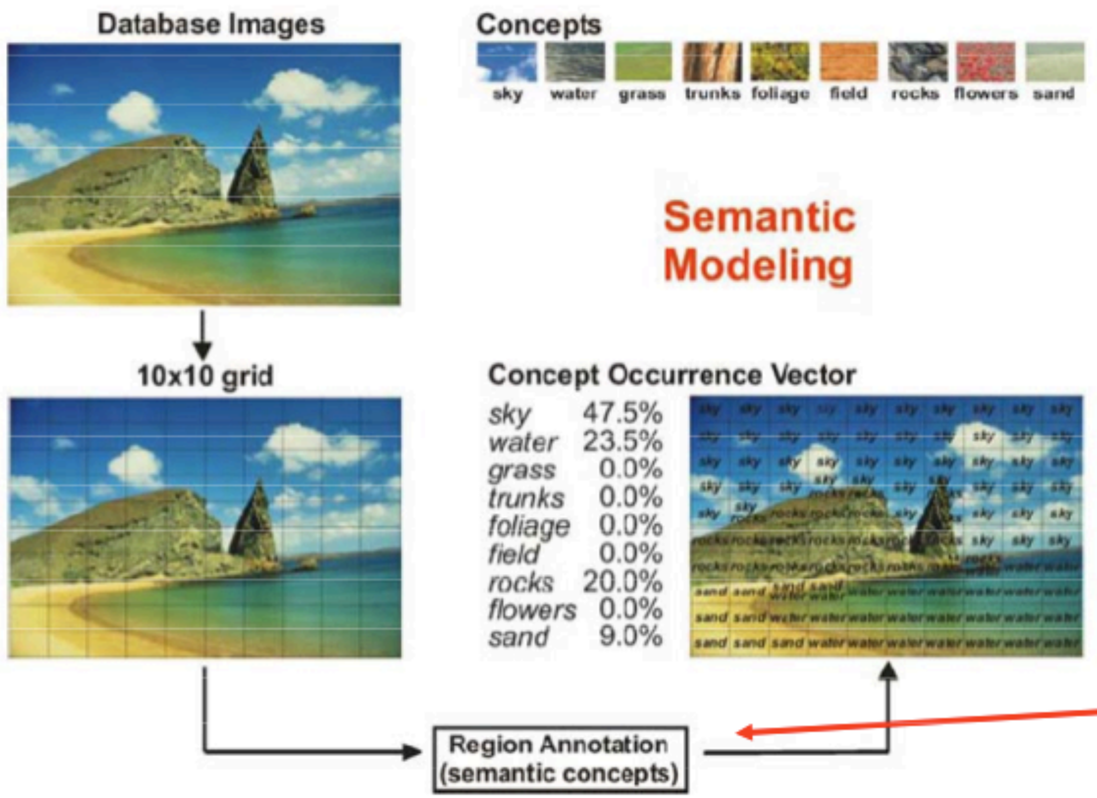
# Image subdivision

- How can we subdivide an image?
  - Object segmentation, not always an easy task
    - When the background can be modelled, we can perform background subtraction

  - Grid subdivision
  - Exhaustive search
  - Local interest points

# Image subdivision

- **Exhaustive grid division**: the whole image is divided into blocks with **no overlap**




Concepts modeled by color and texture features as in the global case. However, we can extract more information per pixel as the area under analysis is smaller

Classification for each image subdivision obtained by SVM classifier

**Semantic Scene Modeling and Retrieval for Content-Based Image Retrieval.** Julia Vogel and Bernt Schiele. *International Journal of Computer Vision*. Vol. 72, No. 2, pp. 133-157, April 2007.

# Image subdivision

- We may miss some objects if these are split over several image blocks
- **Over-sampling grid division**: the whole image is divided into blocks **with overlap**
- Redundant, but less prone to miss objects.


**Scene Classification via PLSA**. A.Bosch, A.Zisserman, X.Muñoz European Conference on Computer Vision, vol. IV, pp. 517-530. Graz, Austria. May 2006.

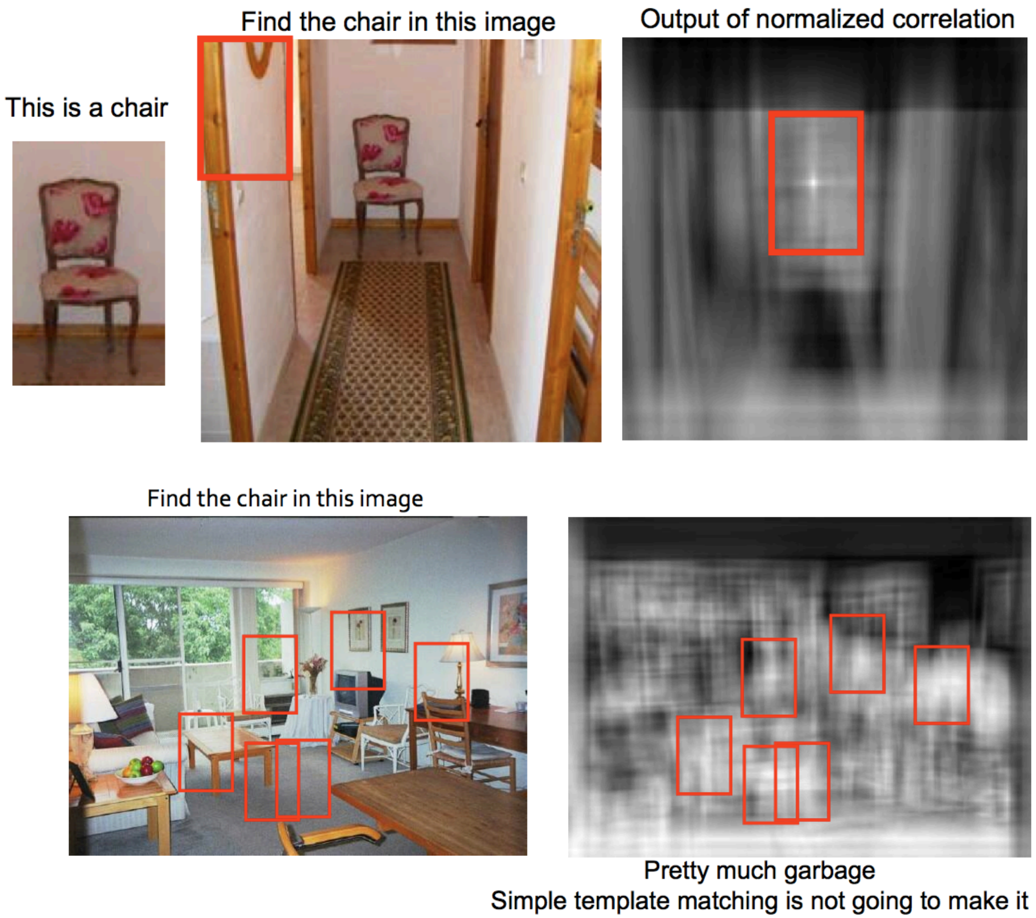
# Image subdivision

- **Scanning image division**: the image is scanned with a fine regular sampling into block (very redundant).
  - Similar to a grid division. However, it is more exhaustive.
  - To detect object at several scales several passes have to be made with variable window size (same applies to rotation).

  - We can do this using **template matching**: cross-correlate the pixels in each area with a model template

# Image subdivision

- **Template matching -** sensitive to noise and **computationally expensive**, i.e. requires presented image to be correlated with every image in the database (no generalization power)



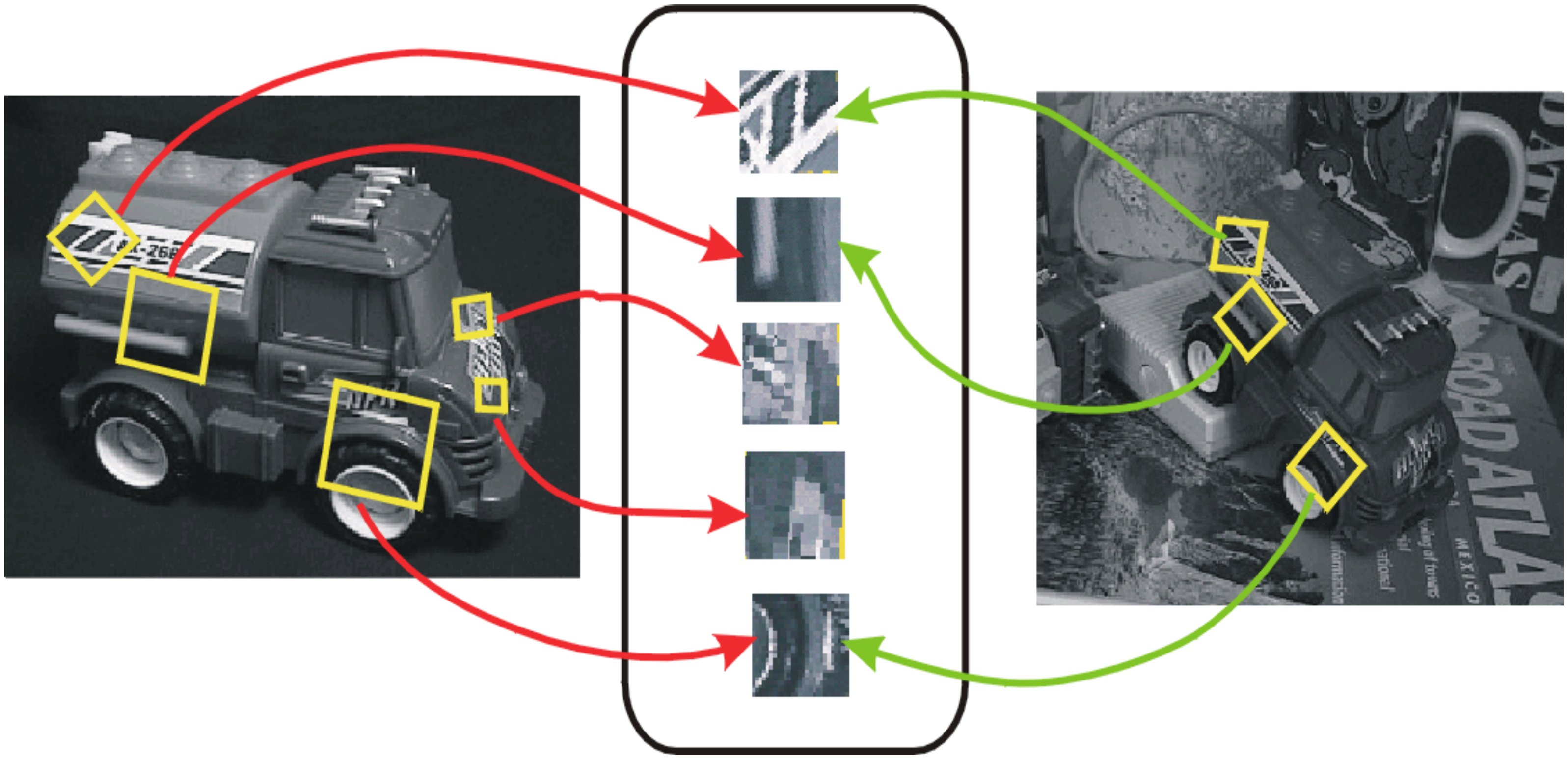
# Local interest points

- Local point detectors were first created to help solve the wide-baseline matching problem.
- **Local point detectors**
  - Detectors that identify specific locations in the image
  - Define areas that are **invariant** to certain transformations
- **Local descriptors**
  - Highly specific, must describe a local area with high **discriminative** power.
- Invariance to transformation can come from either the point detector or the local descriptor.

# Invariant local features

Find features that are invariant to transformations

- geometric invariance:  translation, rotation, scale
- photometric invariance:  brightness, exposure, …



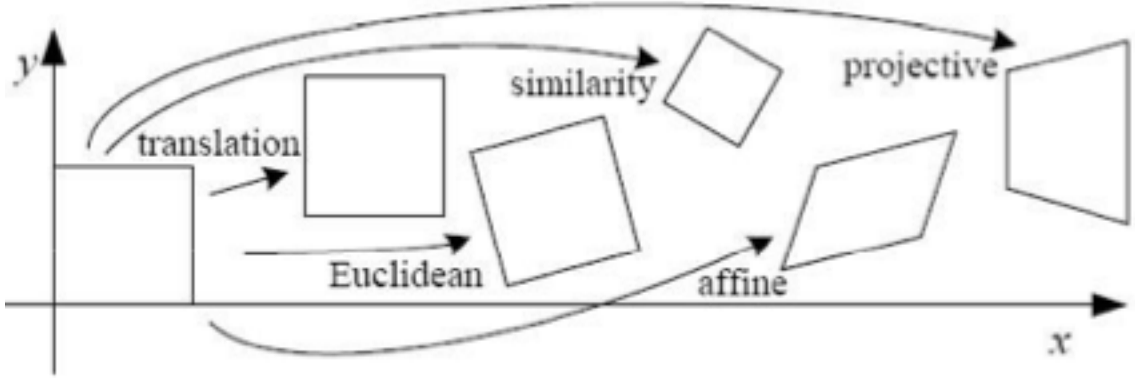

**Feature Descriptors**

# Invariant local features

## Geometric transformations

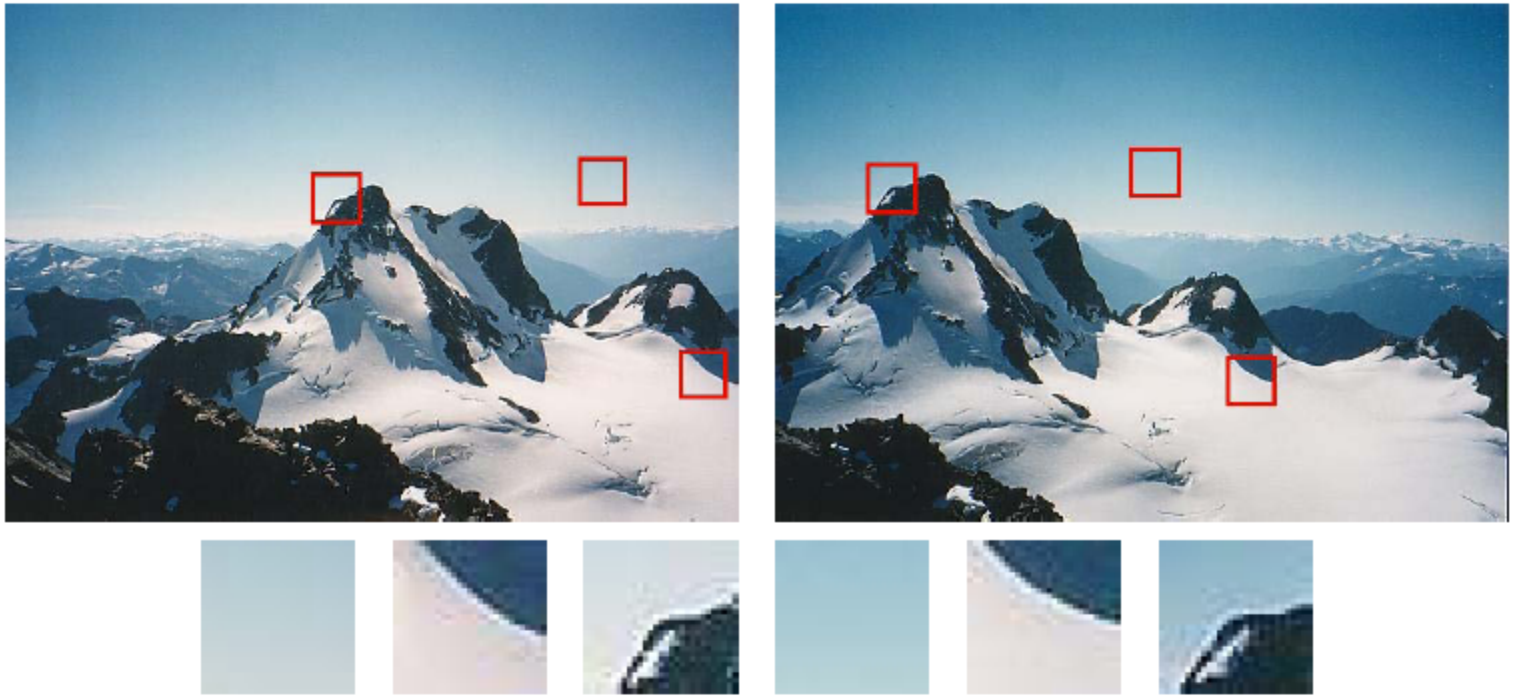
- Translation
- Euclidean (translation + rotation)
- Similarity (translation + rotation + scale)
- Affine transformations
- Projective transformations

**Only holds for planar patches**



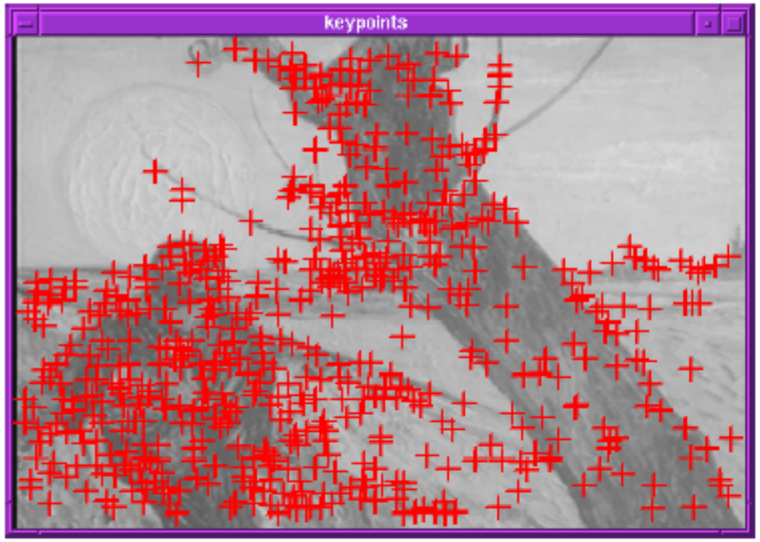
# Local point detectors

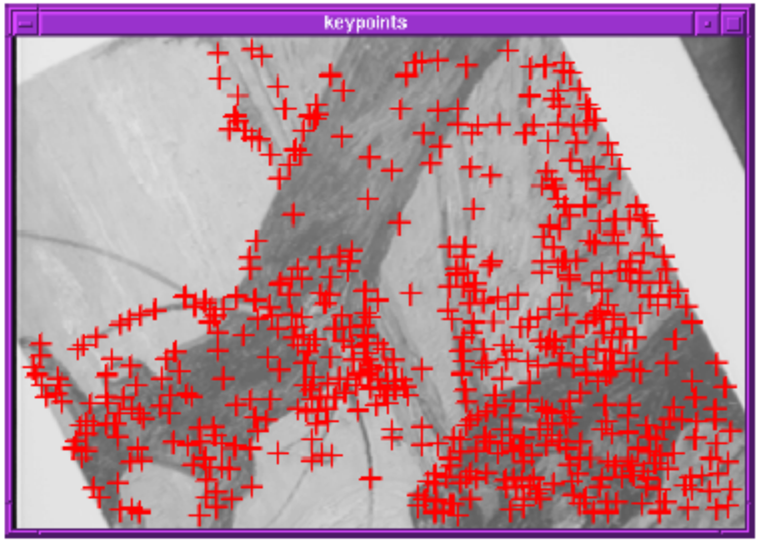
- What are salient features that can be *detected* in multiple views?



# Corners

- Key property: in the region around a corner, image gradient has two or more dominant directions
- Corners are **repeatable** and **distinctive**

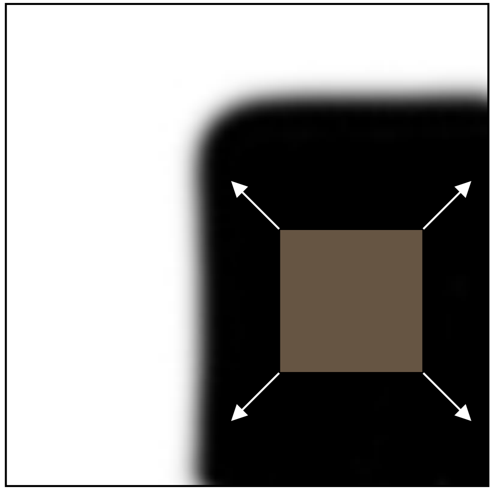




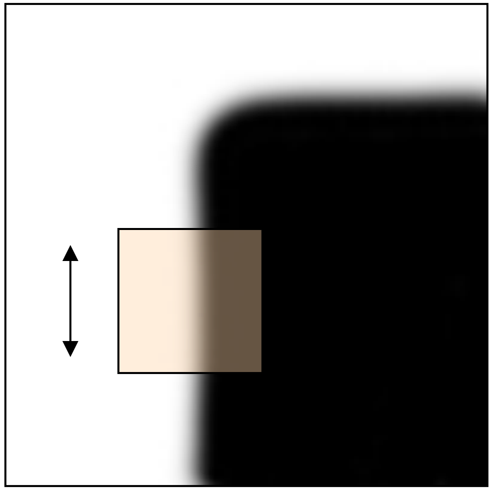

**A Combined Corner and Edge Detector**. C.Harris and M.Stephens. *Proceedings of the 4th Alvey Vision Conference*: pages 147-151, 1988

# Corners

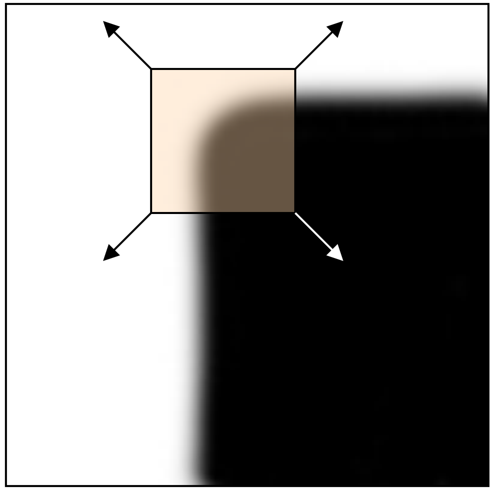
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"**flat**" region:
no change in all directions



"**edge**":
no change along the edge direction



"**corner**":
significant change in all directions

# Harris corner detector

Change of intensity for the shift [*u,v*]:

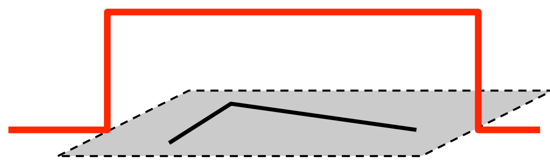
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

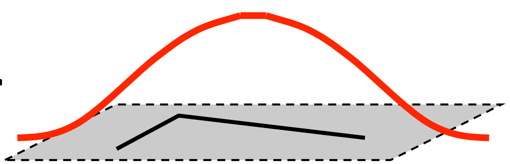
Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

**or**



Gaussian

# Harris corner detector

This measure of change can be approximated by:

$$E(u,v) \approx [u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

where *M* is a 2×2 matrix computed from image derivatives:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Gradient with respect to x, times gradient with respect to y

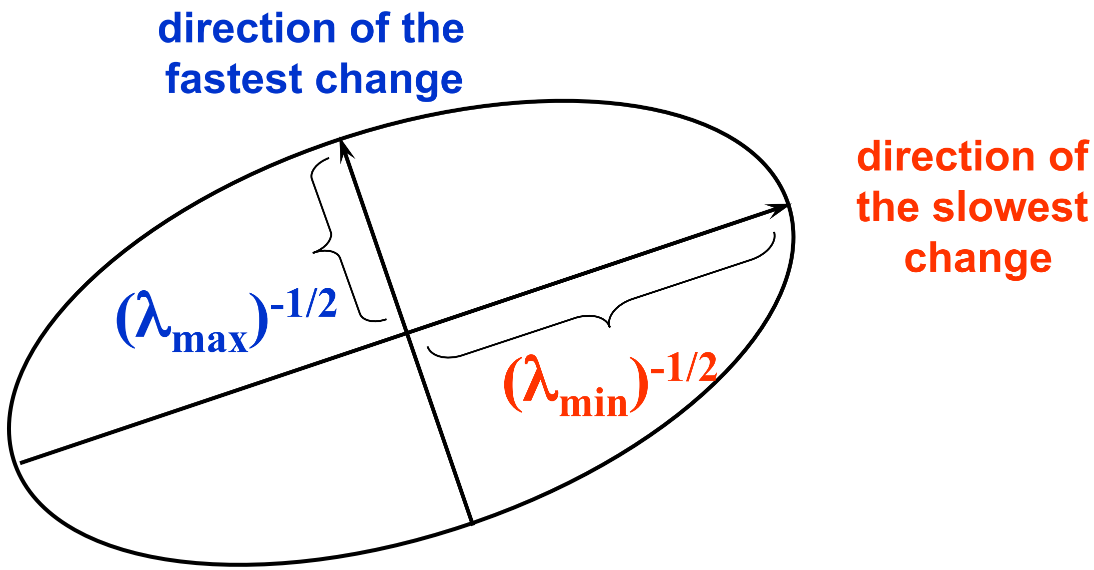
Sum over image region – area we are checking for corner

$$\boldsymbol{M} = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y]$$

# Harris corner detector

Since M is symmetric, we have $M = R^{-1}\begin{bmatrix}\lambda_1 & 0 \\ 0 & \lambda_2\end{bmatrix}R$

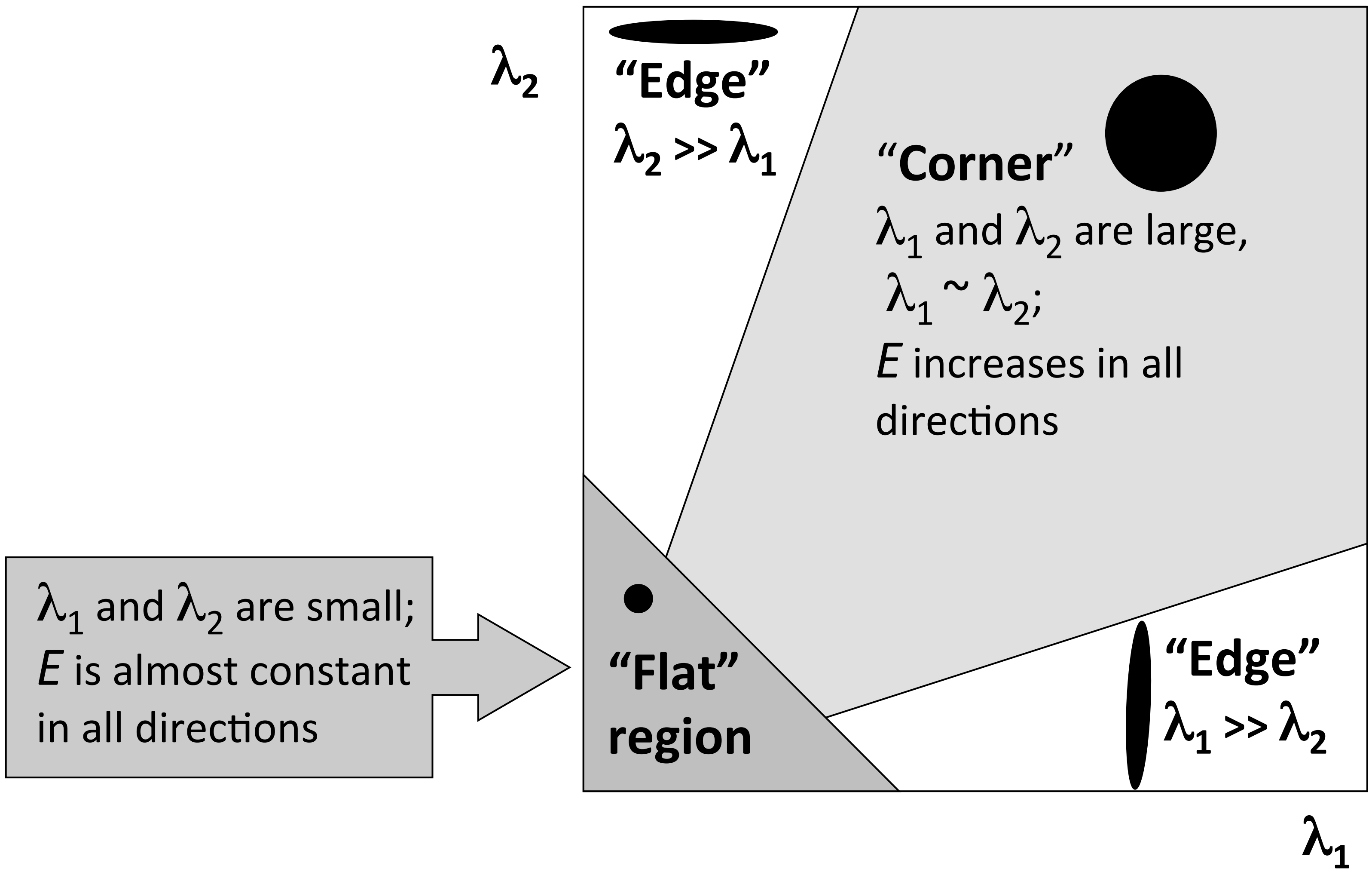
We can visualize *M* as an ellipse with axis lengths determined by the eigenvalues and orientation determined by *R*




Slide adapted form Darya Frolova, Denis Simakov.

# Harris corner detector

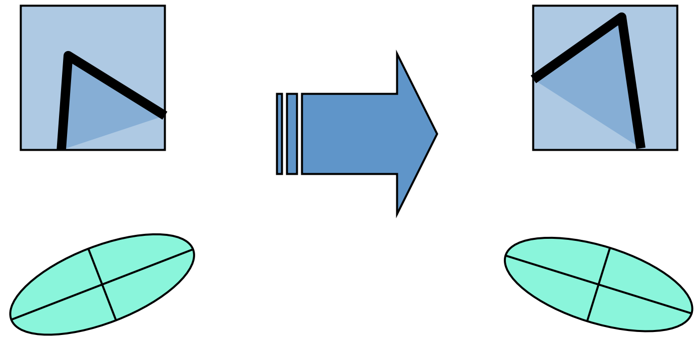
Classification of image points using eigenvalues of *M*:



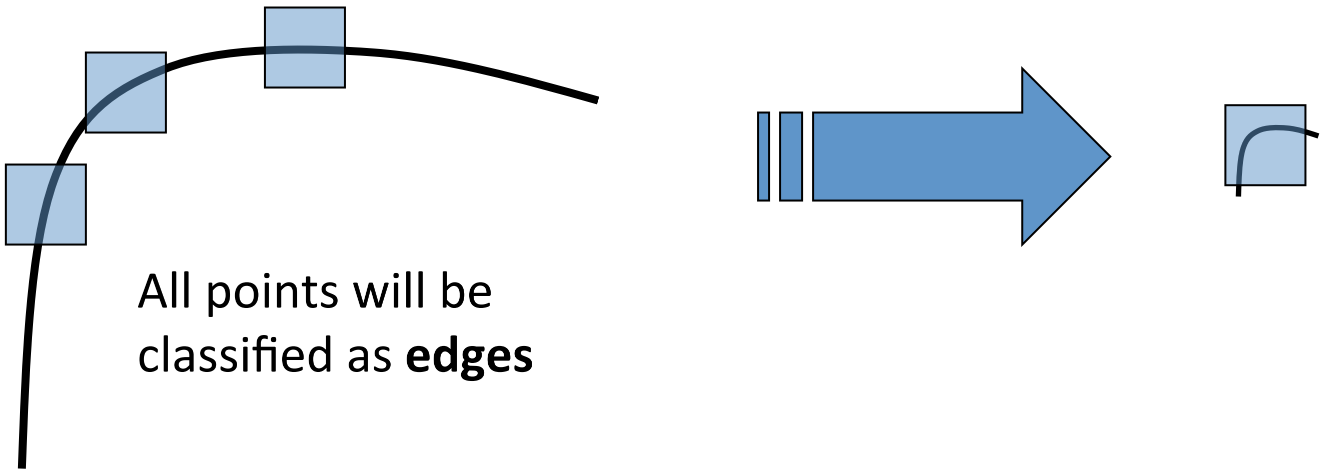
# Harris corner detector

- Properties of the Harris corner detector
  - Rotation invariance

    Ellipse rotates but its shape (i.e. eigenvalues) remains the **same**



  - Not invariant to image scale



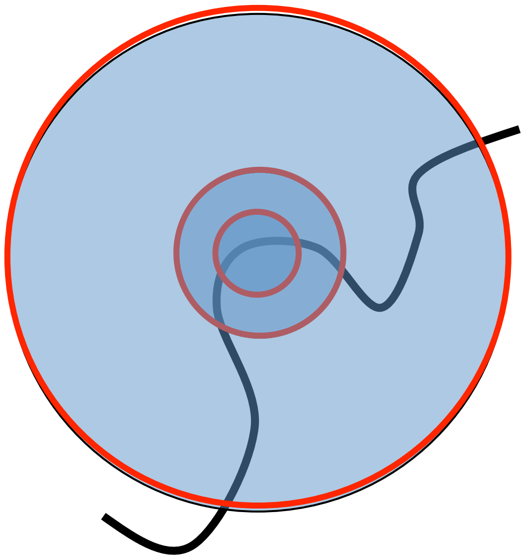
All points will be classified as **edges**

# Harris corner detector

# Harris corner detector

# Scale invariance detection

Suppose you are looking for corners



Key idea: find scale that gives local maximum of f

- f is a local maximum in both position and scale
- Common definition of f: Laplacian
  (or difference between two Gaussian filtered images with different sigmas)

# Scale invariance detection

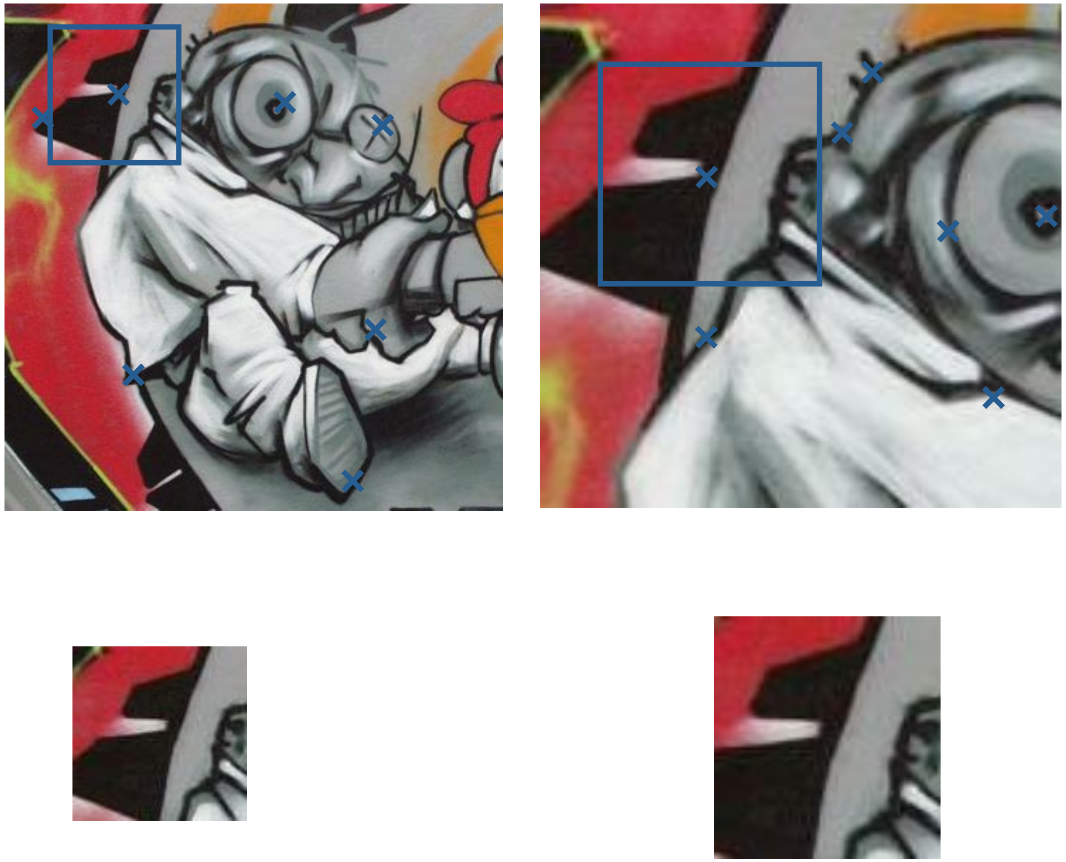
- Multi-scale approach

# Scale invariance detection

- Multi-scale approach

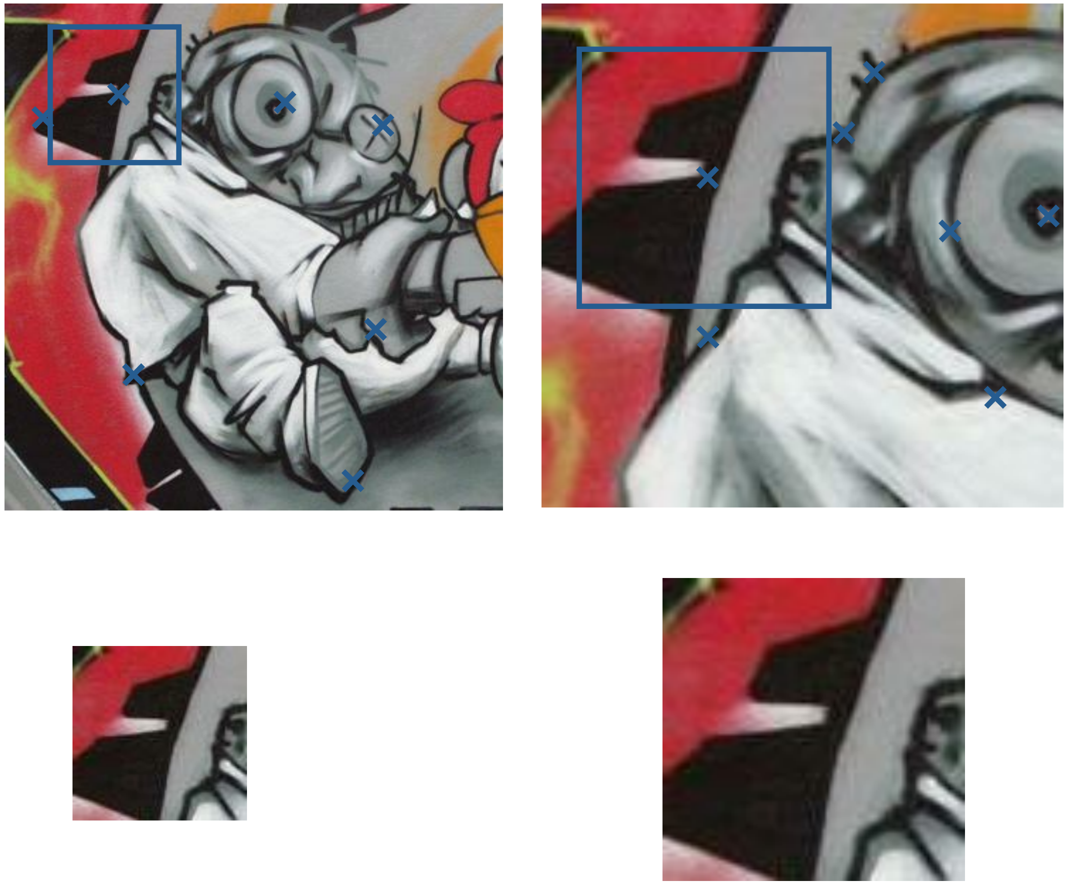
# Scale invariance detection

- Multi-scale approach



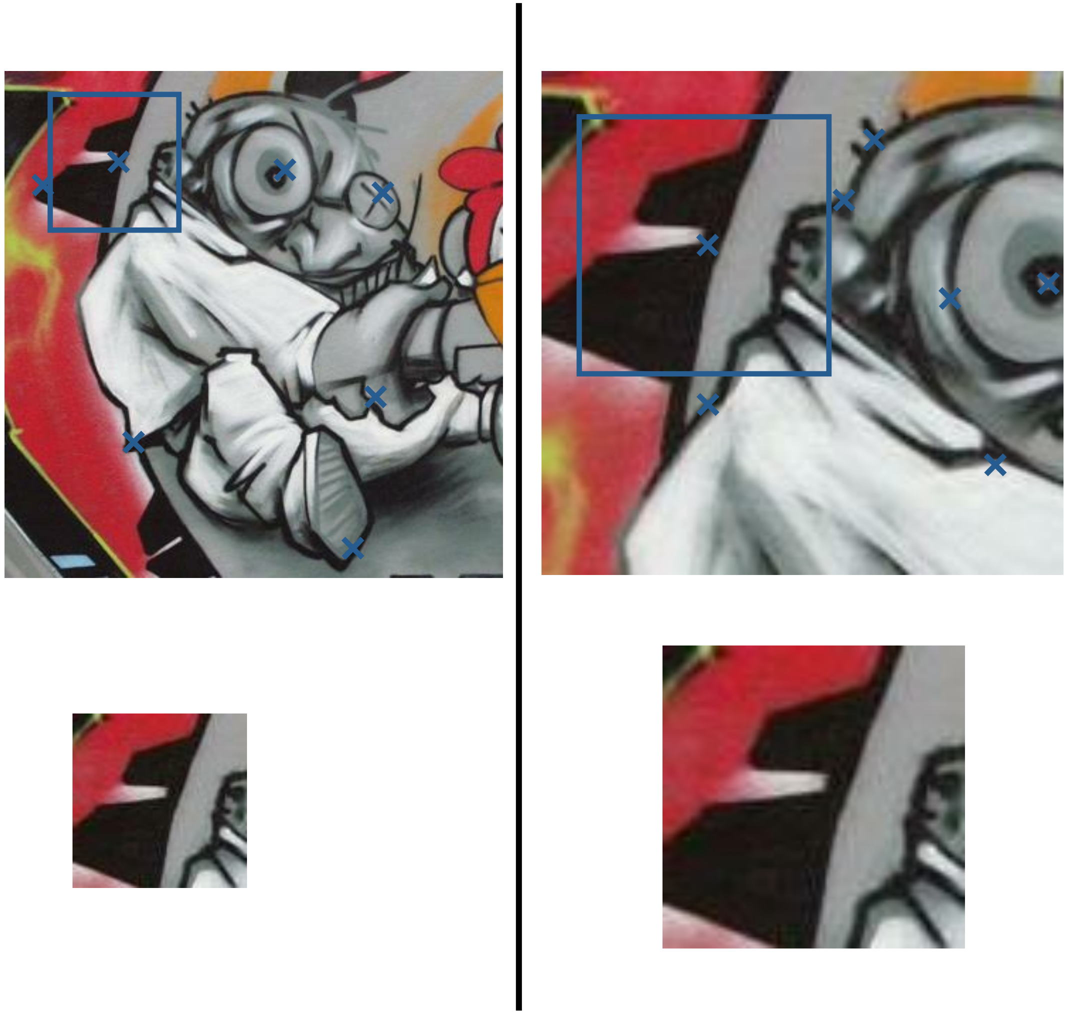
# Scale invariance detection

- Multi-scale approach



# Scale invariance detection

- Extract patch from each image individually

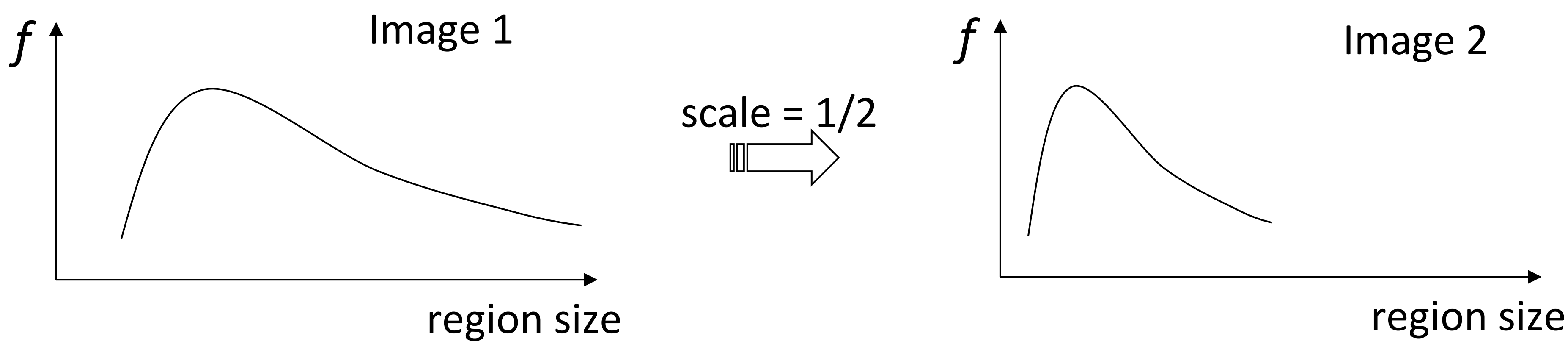


# Scale invariance detection

- Solution for an automatic scale detection:
  - Design a function *f* on the region, which is "scale invariant" (*the same for corresponding regions, even if they are at different scales*)

    Example: average intensity. For corresponding regions (even of different sizes) it will be the same.

  - For a point in one image, we can consider it as a function of region size (patch width)

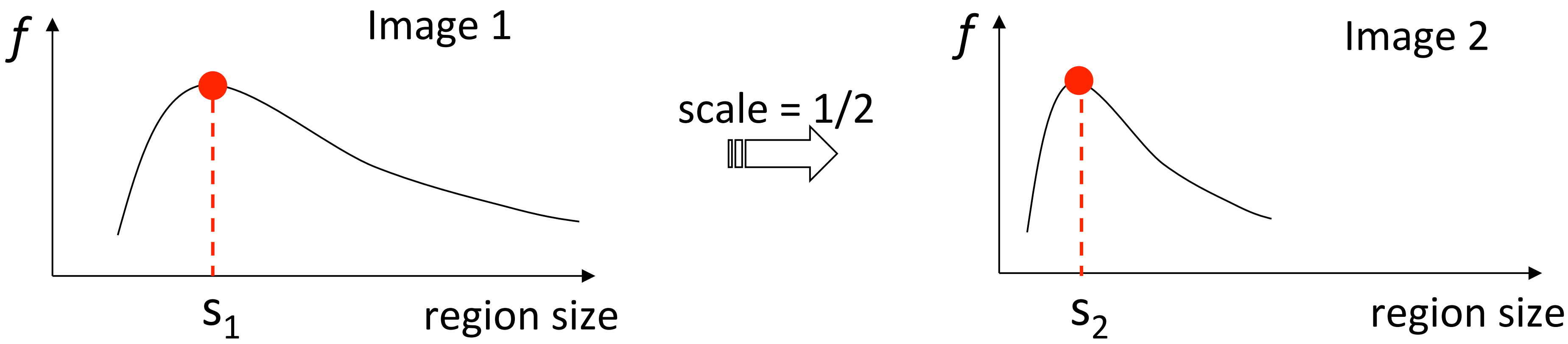


# Scale invariance detection

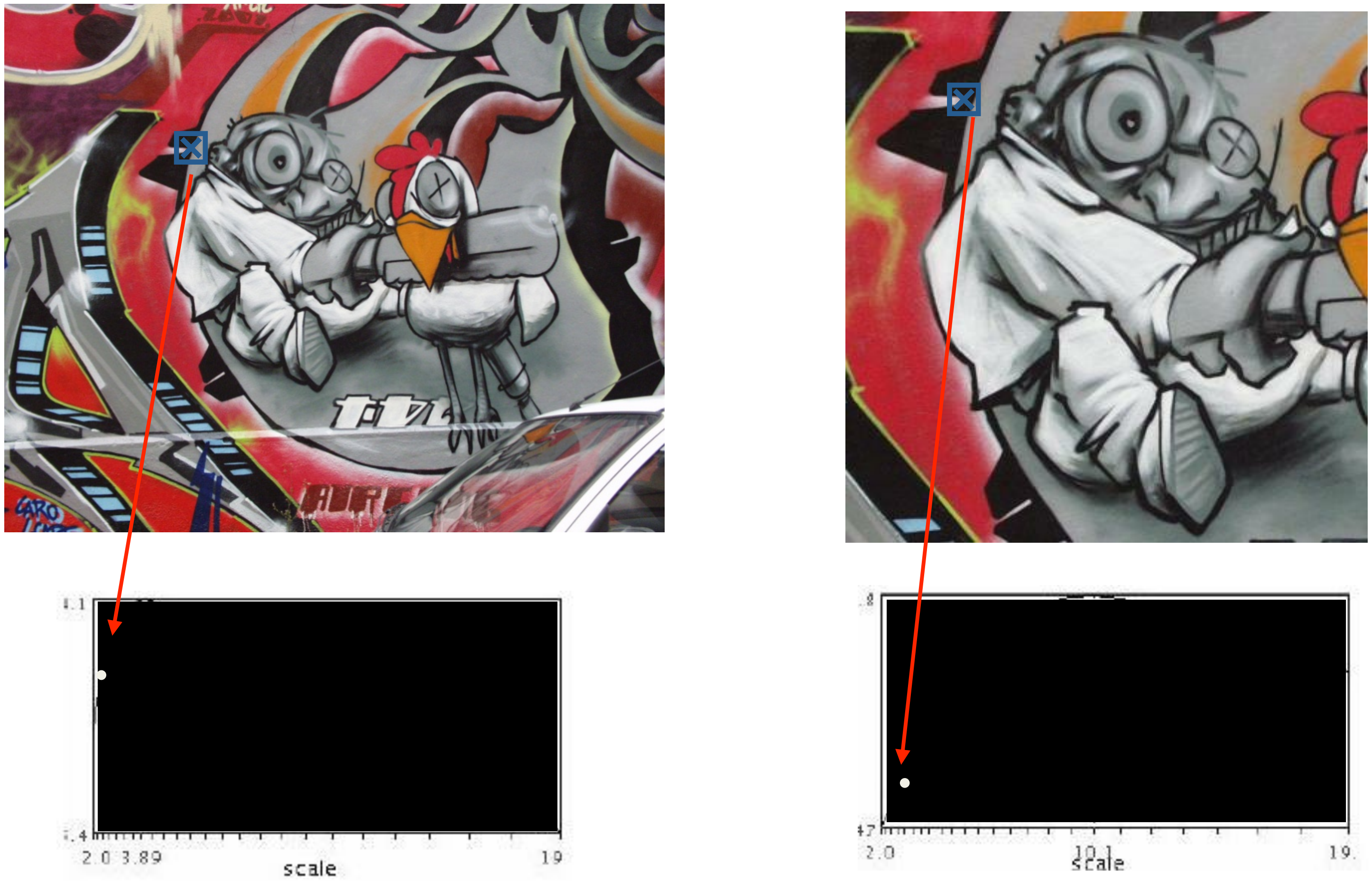
- Common approach:

  Take a local maximum of this function

  Observation: region size, for which the maximum is achieved, should be *invariant* to image scale.

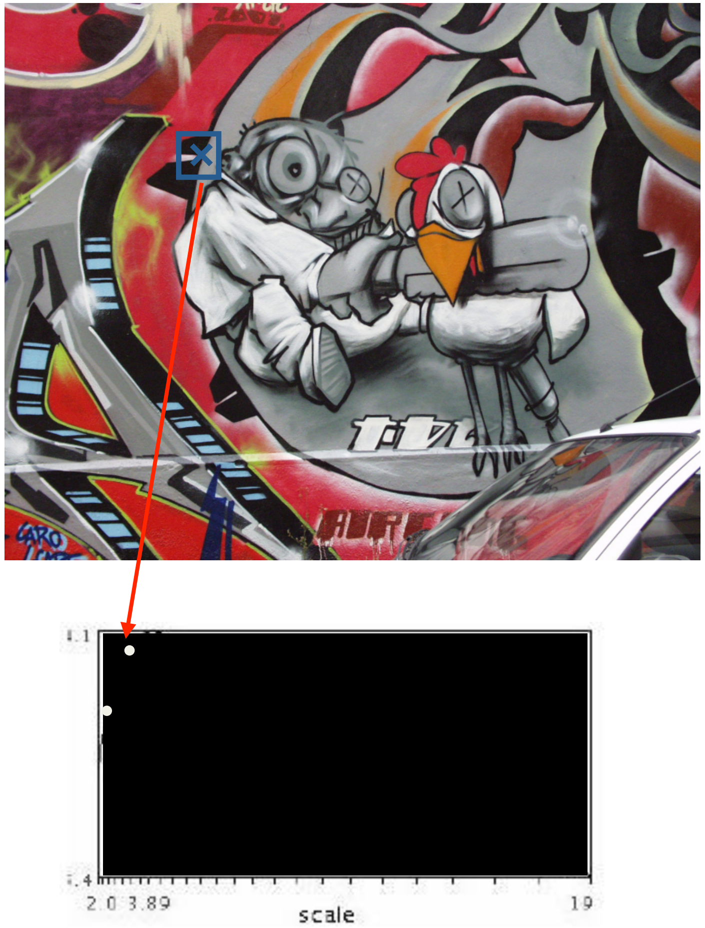


# Scale invariance detection




$f(I_{i_1 \ldots i_m}(x,\sigma))$ $f(I_{i_1 \ldots i_m}(x',\sigma))$

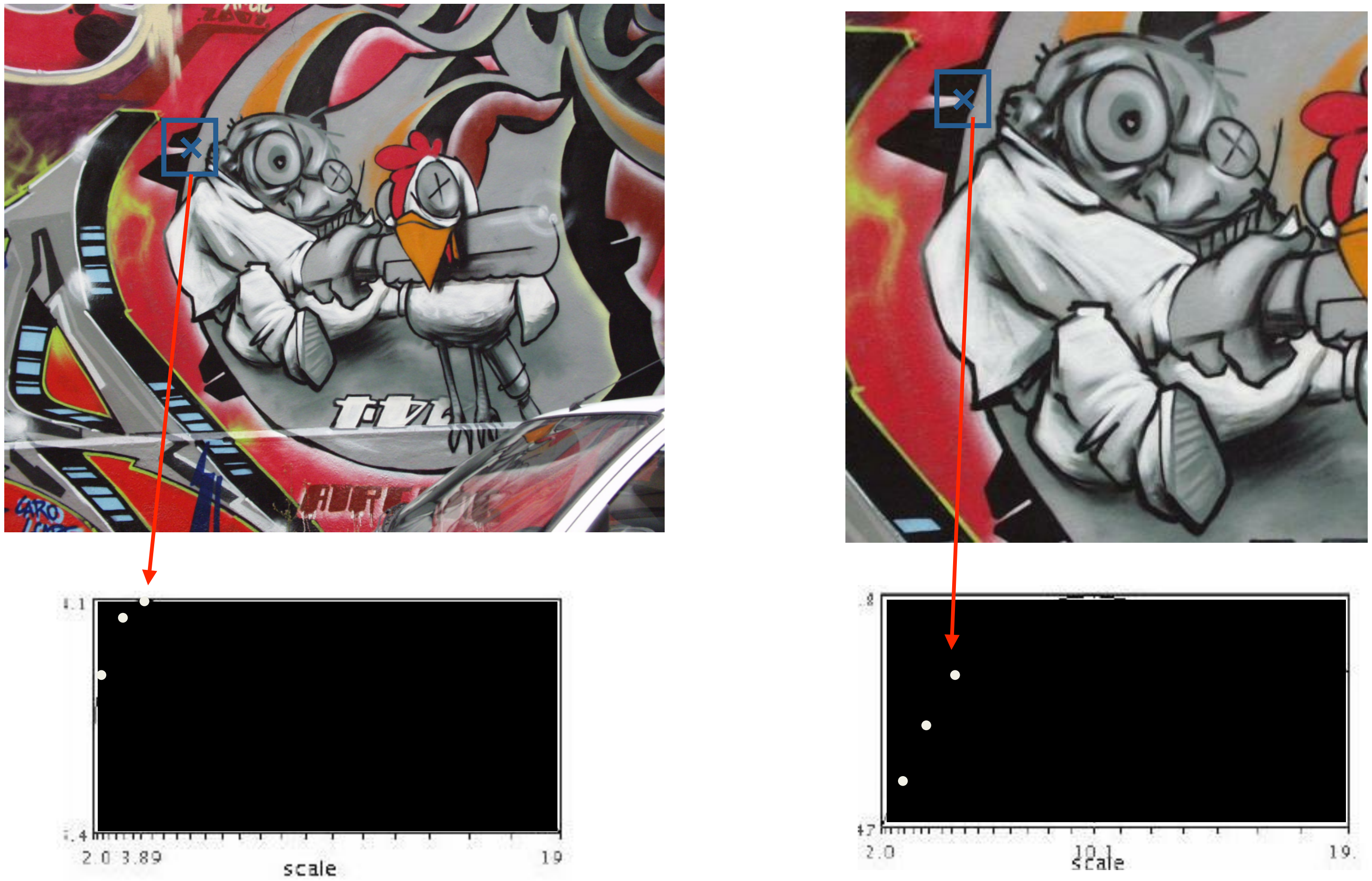
# Scale invariance detection




$f(I_{i_1 \ldots i_m}(x,\sigma))$

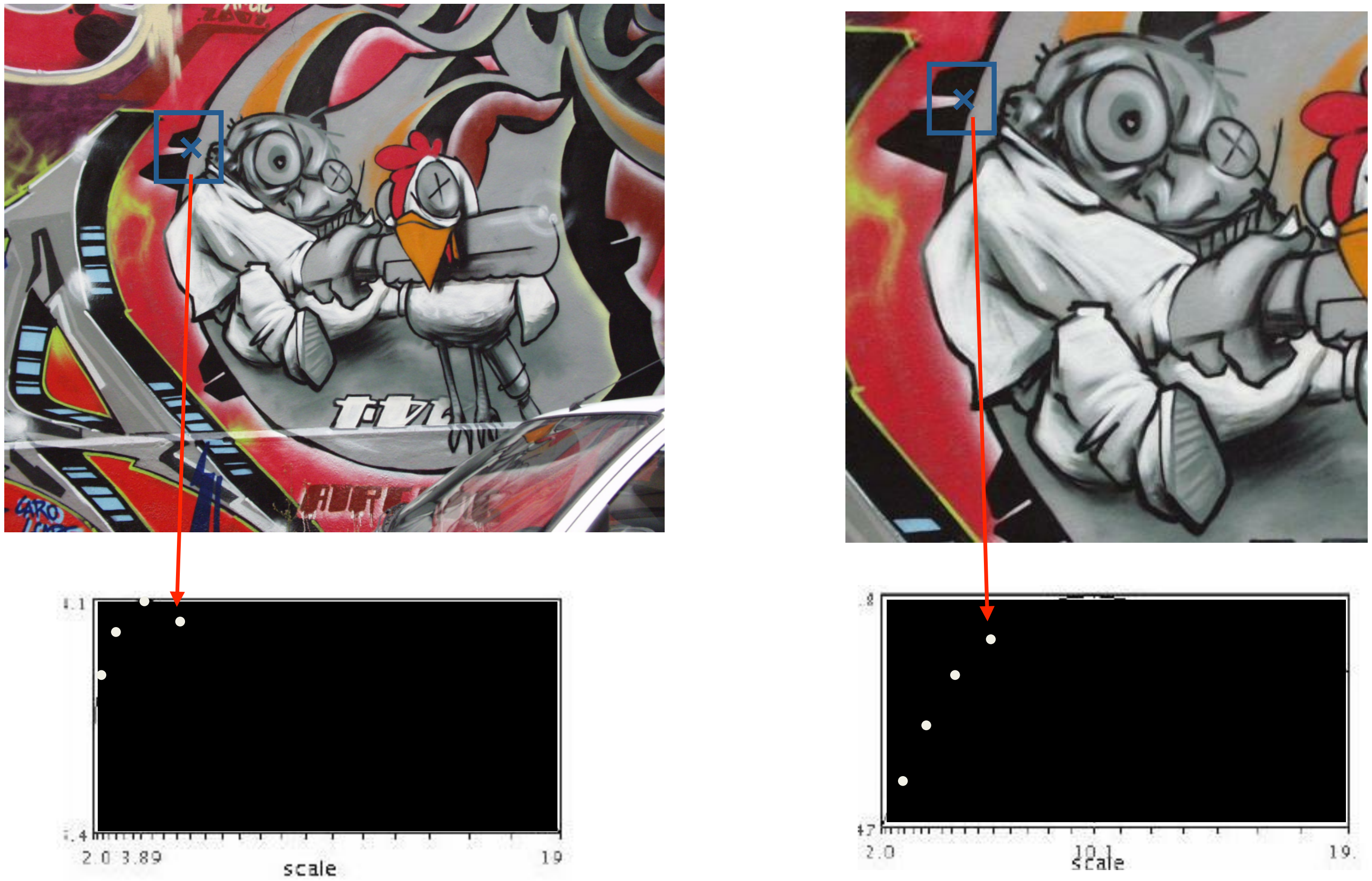

$f(I_{i_1 \ldots i_m}(x',\sigma))$

# Scale invariance detection



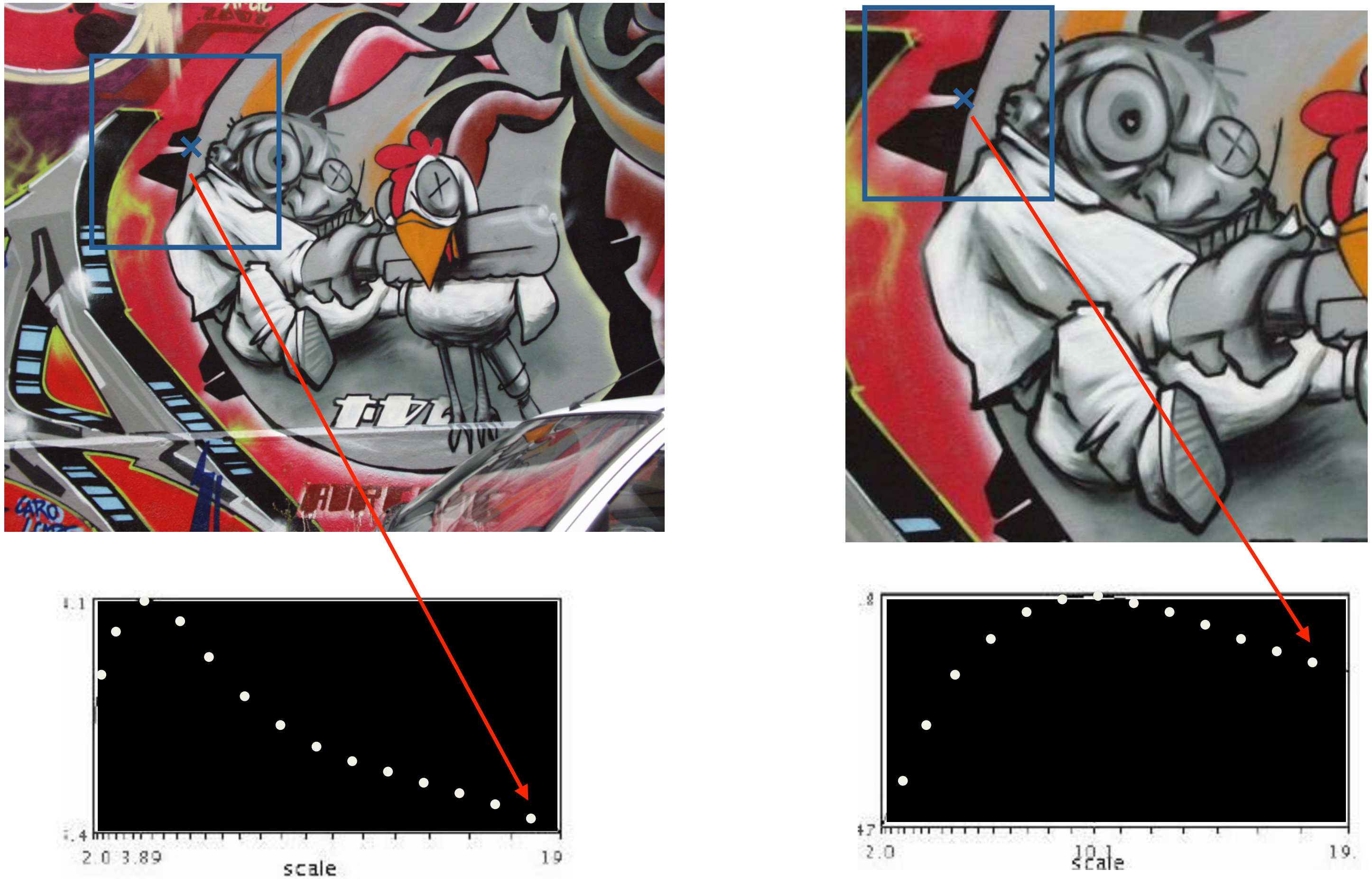

$f(I_{i_1\ldots i_m}(x,\sigma))$ $f(I_{i_1\ldots i_m}(x',\sigma))$

# Scale invariance detection




$f(I_{i_1 \dots i_m}(x, \sigma))$ $\qquad$ $f(I_{i_1 \dots i_m}(x', \sigma))$

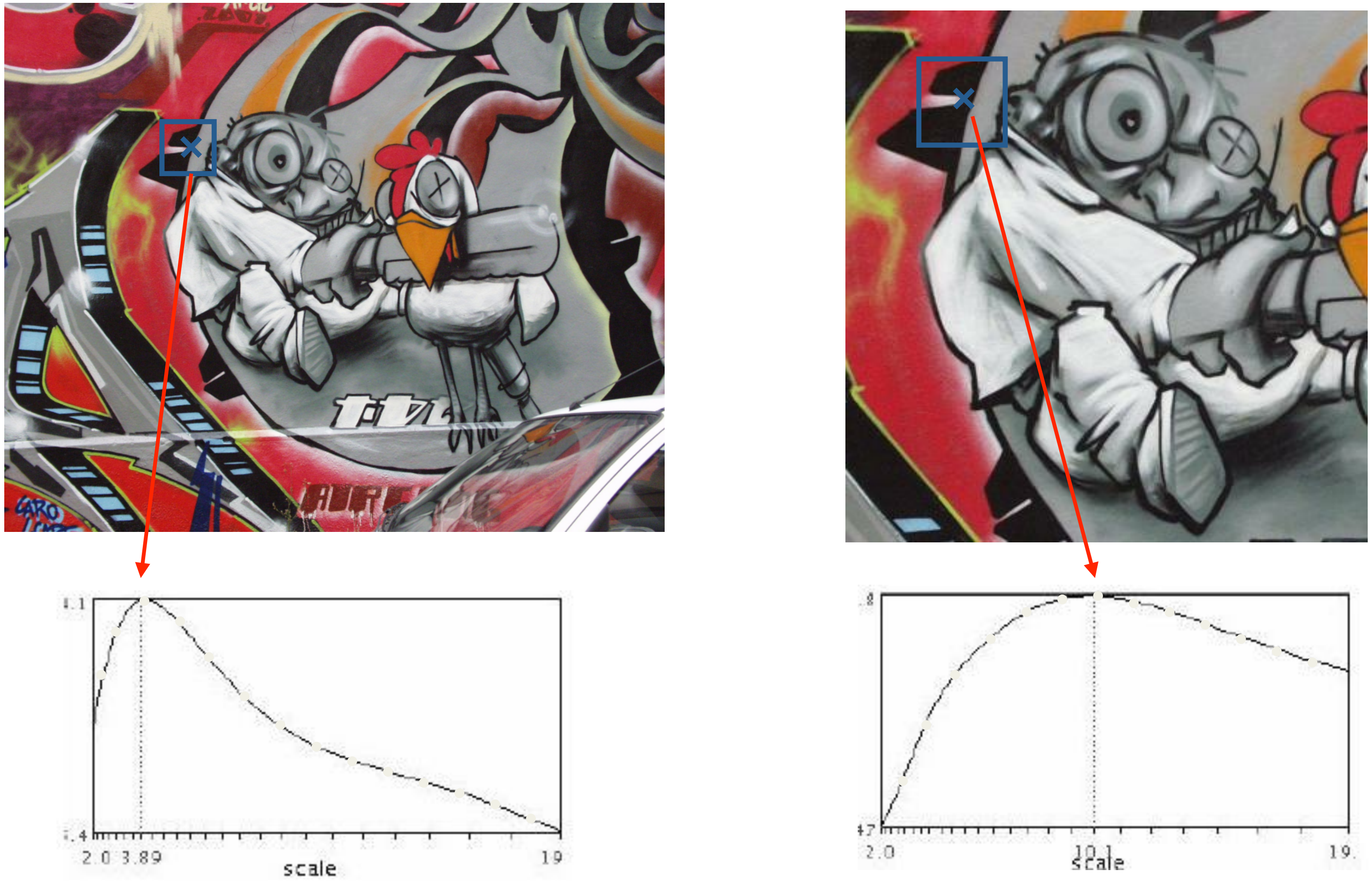
# Scale invariance detection




$f(I_{i_1 \ldots i_m}(x,\sigma))$

$f(I_{i_1 \ldots i_m}(x',\sigma))$

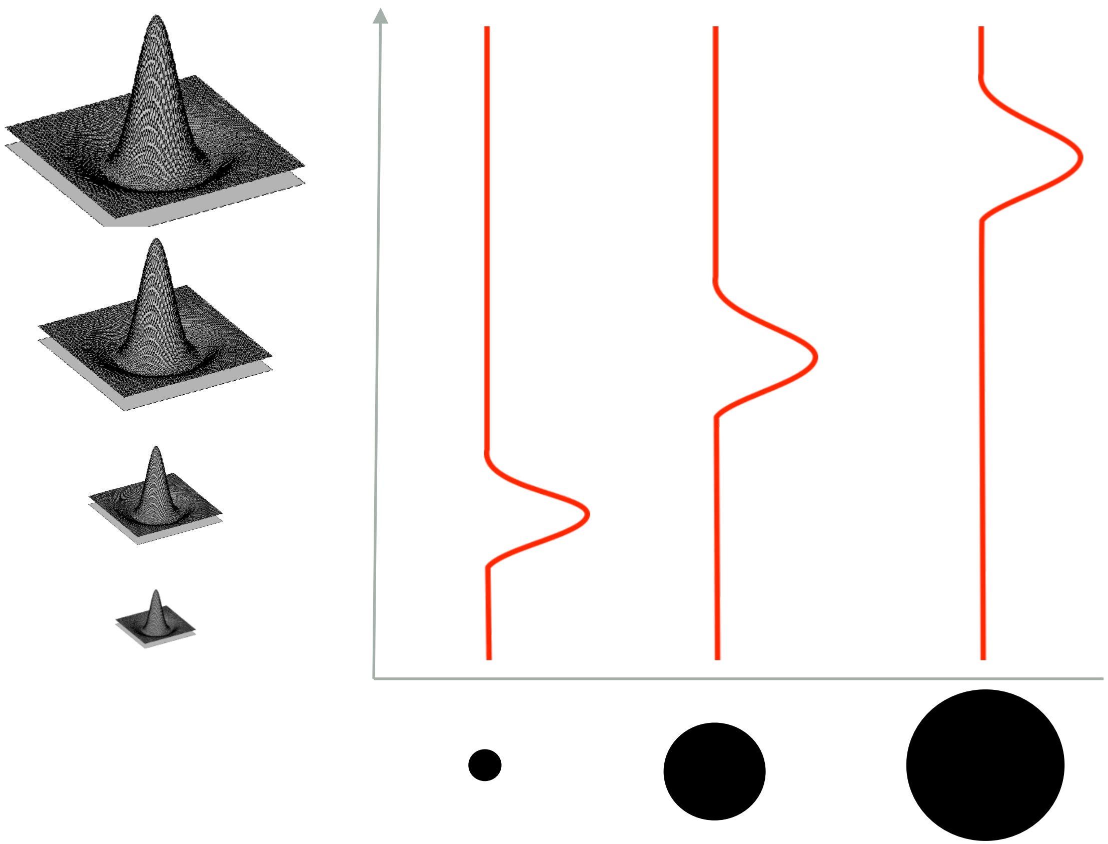
# Scale invariance detection




$f(I_{i_1 \ldots i_m}(x, \sigma))$ $f(I_{i_1 \ldots i_m}(x', \sigma'))$

# Scale invariance detection

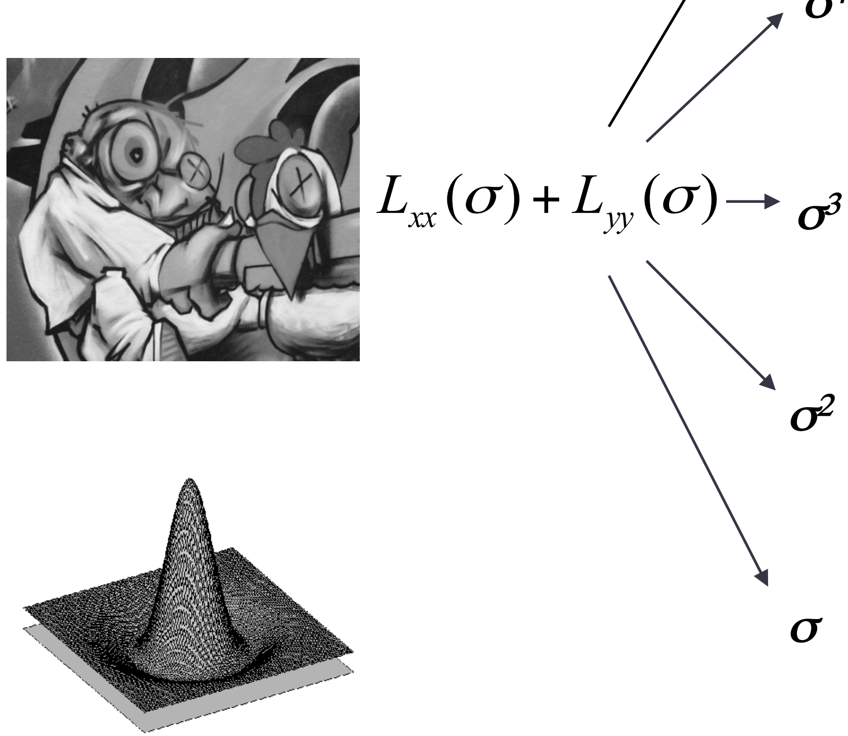
- Useful signature function
  - Laplacian-of-Gaussian = “blob” detector

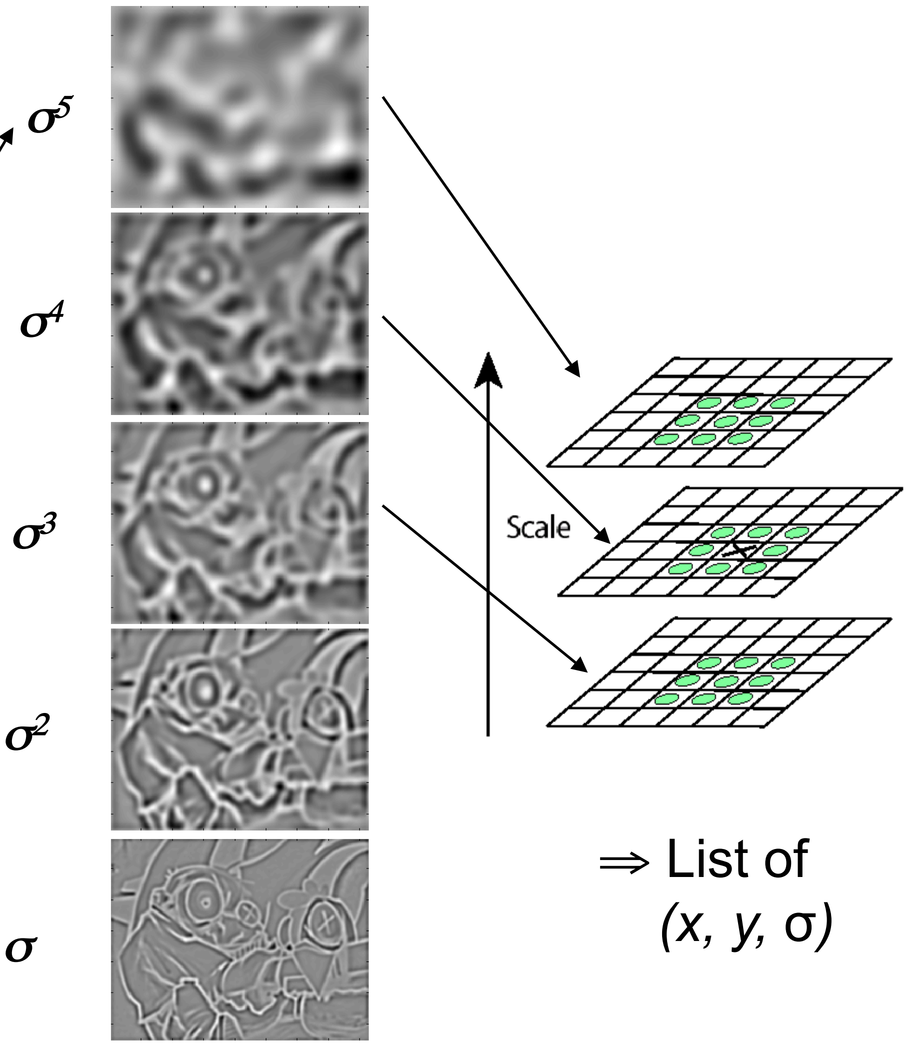


# Scale invariance detection

**Interest points:**

Local maxima in scale space of Laplacian-of-Gaussian (LoG)

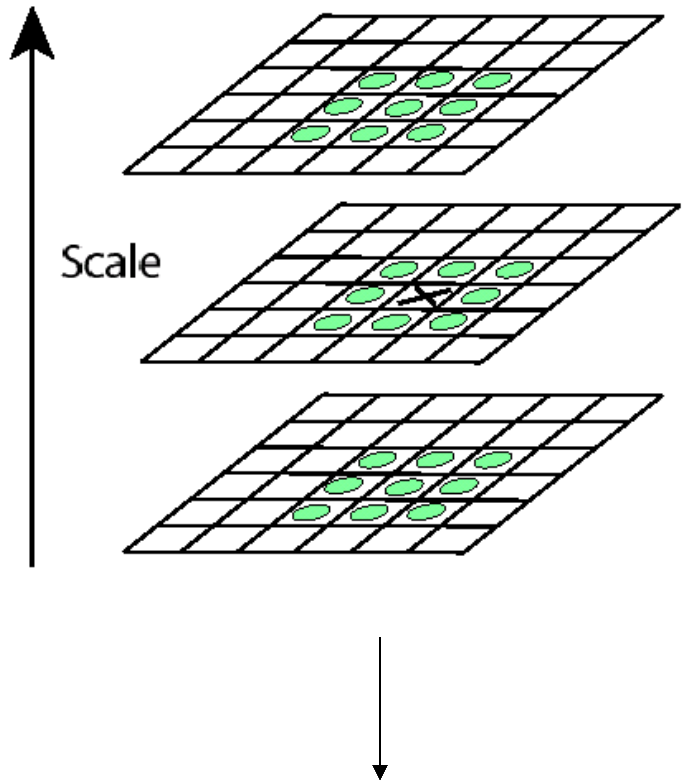



$\Rightarrow$ List of $(x, y, \sigma)$

# Scale invariance detection

- LoG can be approximated by a Difference of two Gaussians (DoG) at different scales
- Detect maxima of DoG in the scale space volume
- Reject points with low contrast (threshold)
- Reject points that are localized along an edge




Candidate keypoints: list of (x,y,σ)

# Scale-space blob detector: Example

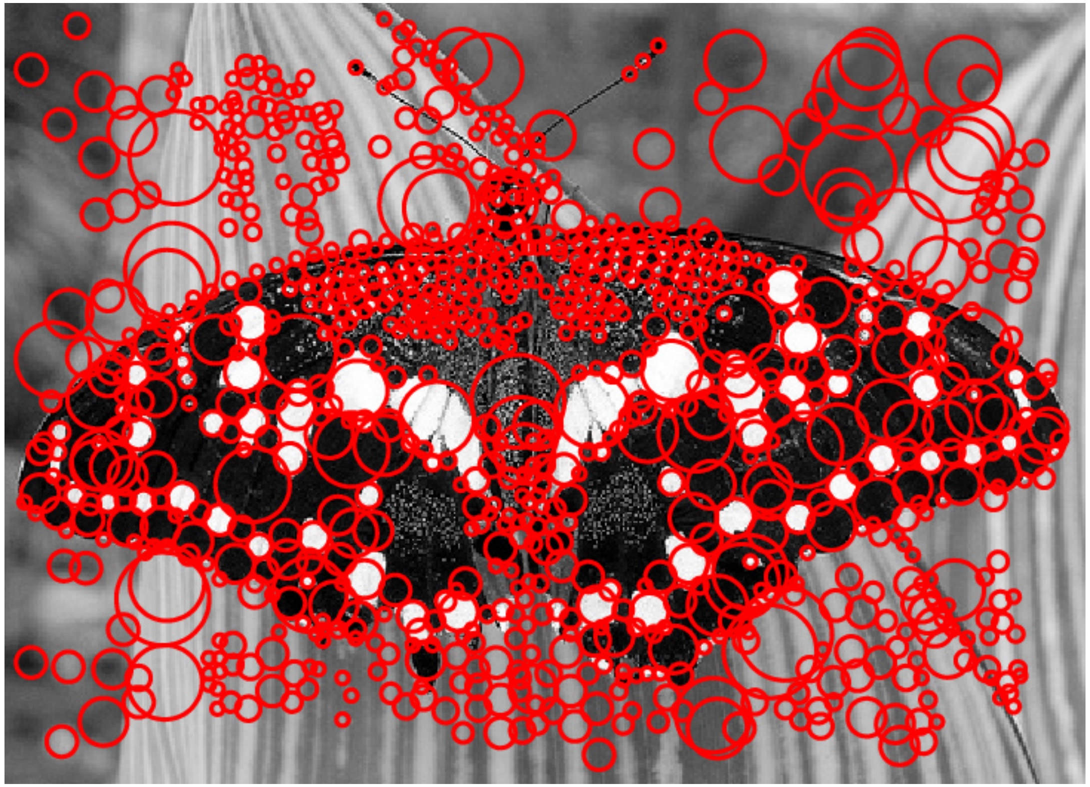
Source: Lana Lazebnik

# Scale-space blob detector: Example

sigma = 11.9912

Source: Lana Lazebnik

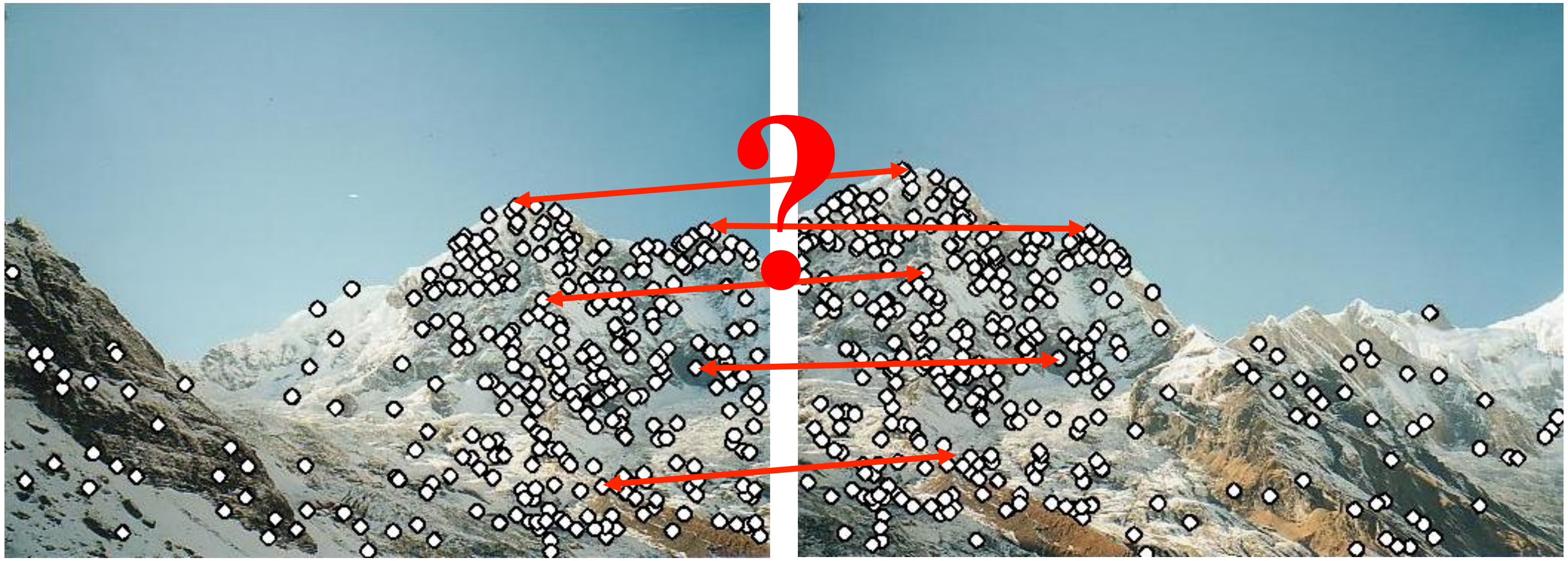
# Scale-space blob detector: Example



Source: Lana Lazebnik

# Local descriptors

- How can we describe interest points for matching?




Point descriptor should be:

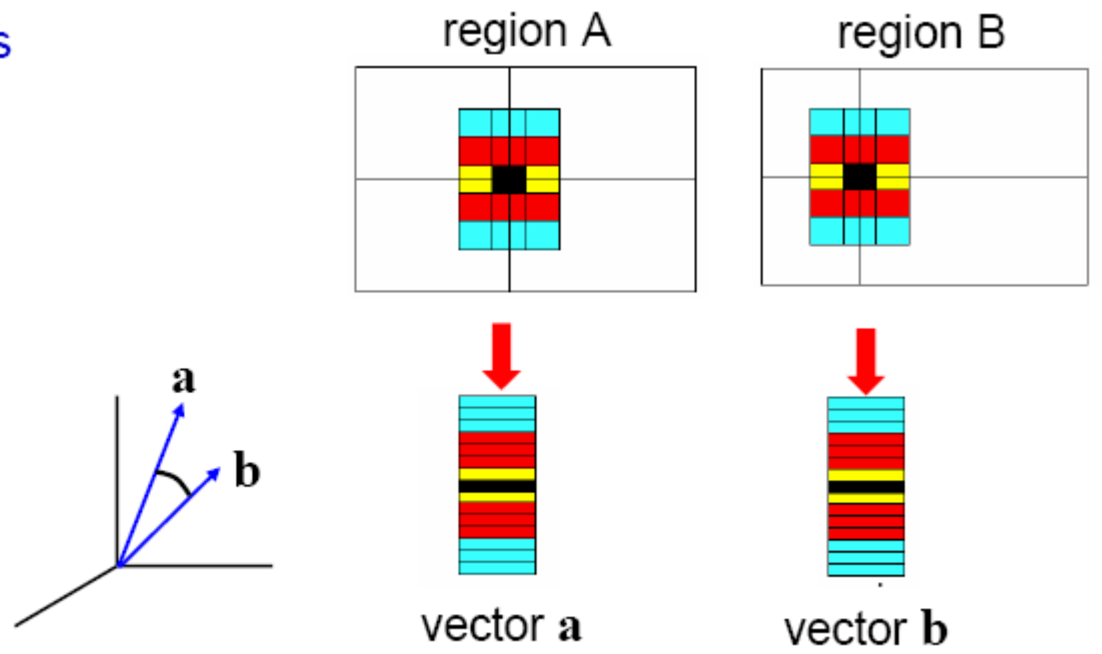
1. Invariant
2. Distinctive

# Local descriptors

- Simplest descriptor: list of intensities within a patch.
- What is this going to be invariant to?

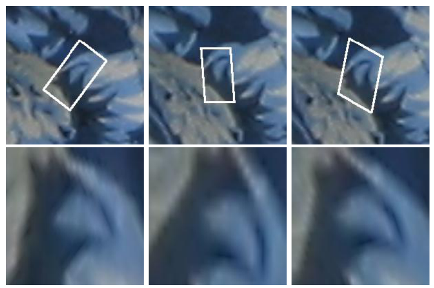
Write regions as vectors

$$A \rightarrow \mathbf{a}, \; B \rightarrow \mathbf{b}$$

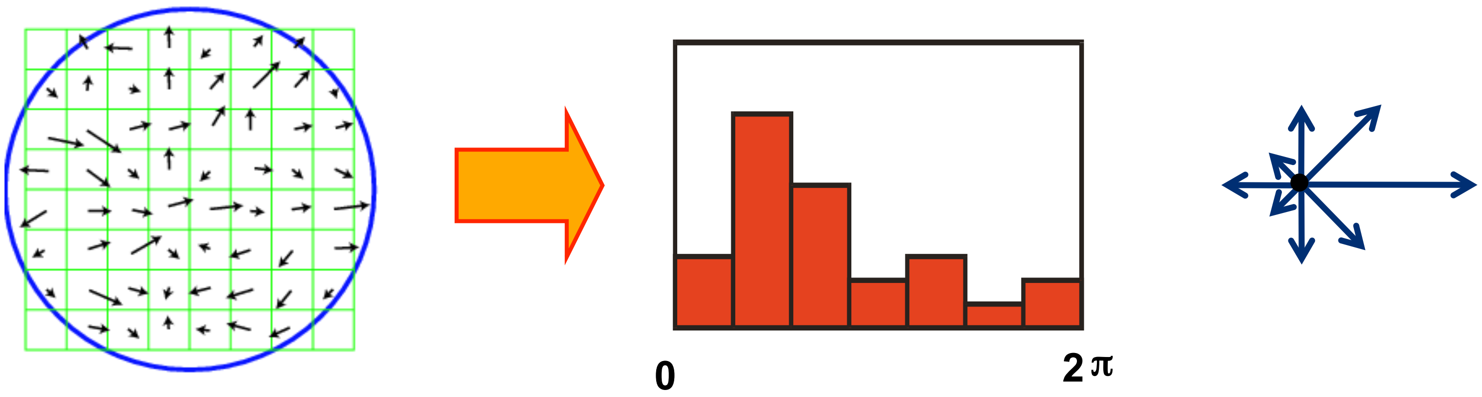


# Local descriptors

- Disadvantage of patches as descriptors:
  - Small shifts can affect matching score a lot



- Solution: histograms




Source: Lana Lazebnik

# Local descriptors

- Histogram-based descriptors
  - Based on the histogram of oriented gradient
  - SIFT, SURF, GLOH and HOG
- Compact descriptors
  - Based on binary strings obtained comparing pairs of image intensities
  - BRIEF, ORB, BRISK and FREAK

# SIFT descriptor

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

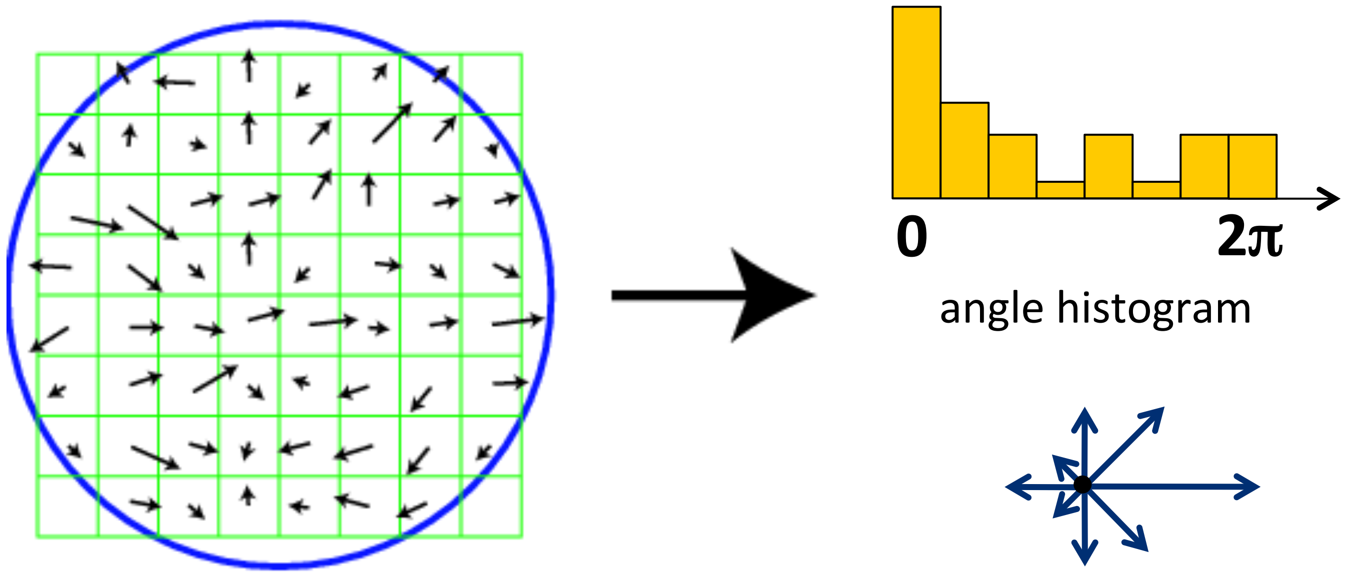


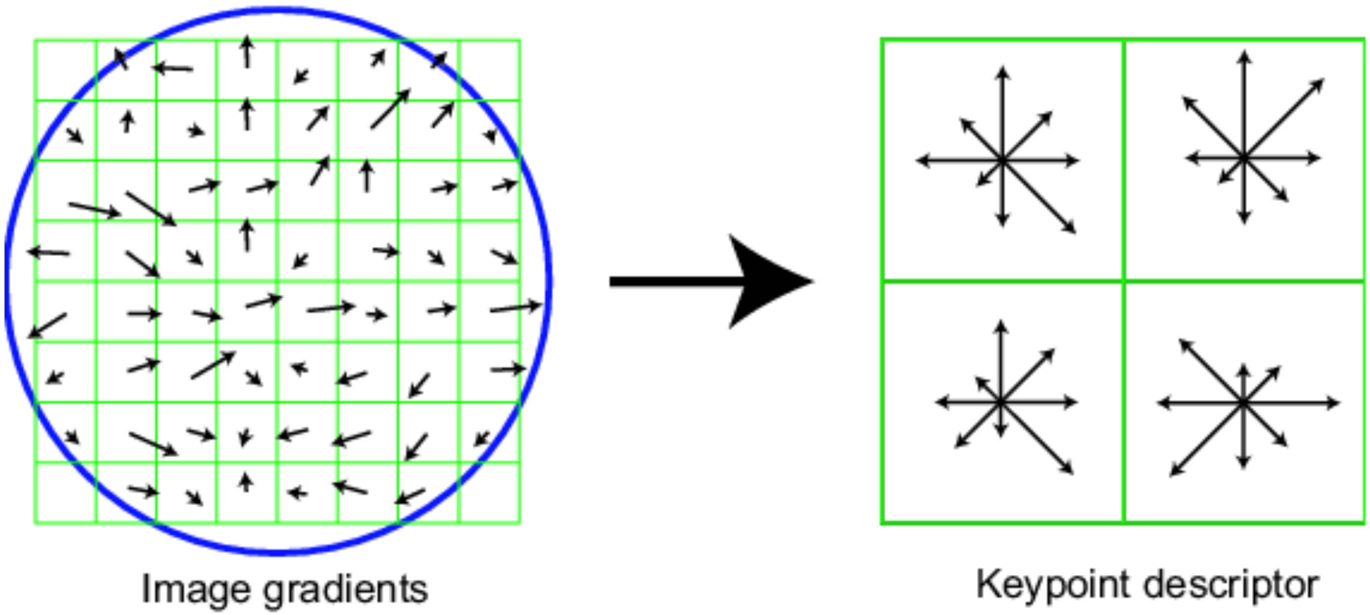

Image gradients

**Distinctive image features from scale-invariant keypoints**. David G. Lowe. *IJCV* 60 (2), pp. 91-110, 2004.

# SIFT descriptor

Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor



# SIFT descriptor

- One image yields:
  - n 128-dimensional descriptors: each one is a histogram of the gradient orientations within a patch
    - [n x 128 matrix]
  - n scale parameters specifying the size of each patch
    - [n x 1 vector]
  - n orientation parameters specifying the angle of the patch
    - [n x 1 vector]
  - n 2d points giving positions of the patches
    - [n x 2 matrix]

# Feature matching

Given a feature in $I_1$, how to find the best match in $I_2$?

1. Define distance function that compares two descriptors
2. Test all the features in $I_2$, find the one with min distance

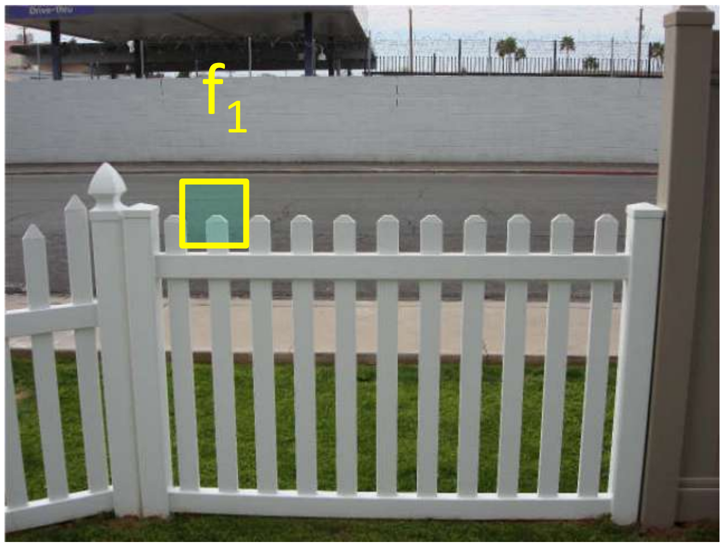
$I_1$

$I_2$

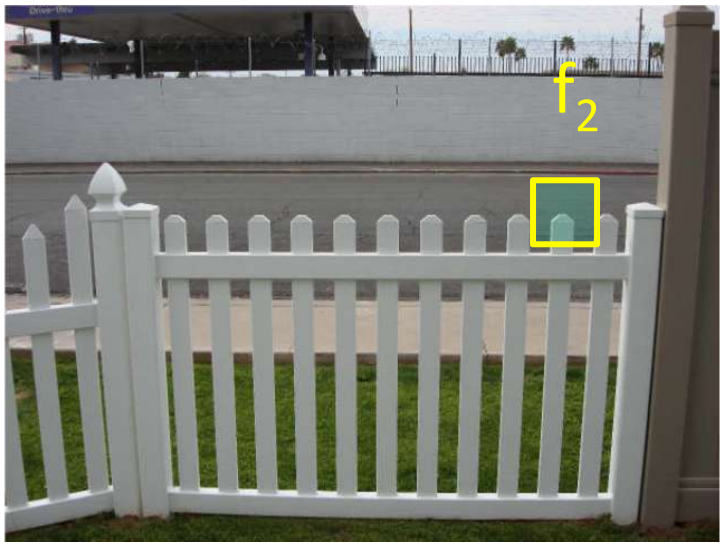
# Feature matching

How to define the difference between two features $f_1$, $f_2$?

– Simple approach is SSD($f_1$, $f_2$)
  - sum of square differences between entries of the two descriptors
  - can give good scores to very ambiguous (bad) matches




$I_1$

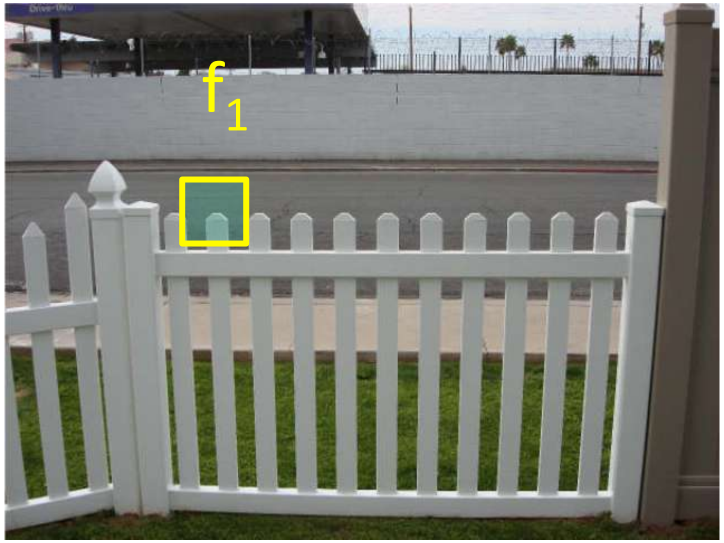



$I_2$

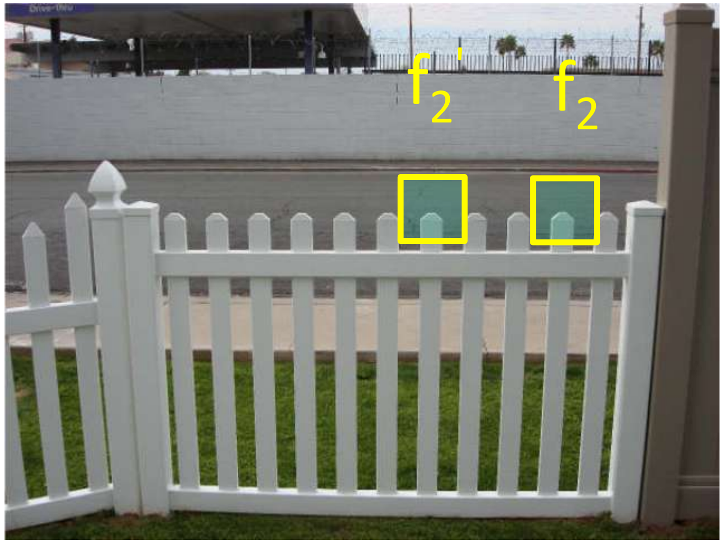
# Feature matching

How to define the difference between two features $f_1$, $f_2$?

– Ratio distance = $SSD(f_1, f_2) / SSD(f_1, f_2')$
  - $f_2$ is best SSD match to $f_1$ in $I_2$
  - $f_2'$ is $2^{nd}$ best SSD match to $f_1$ in $I_2$
  - gives large values (~1) for ambiguous matches

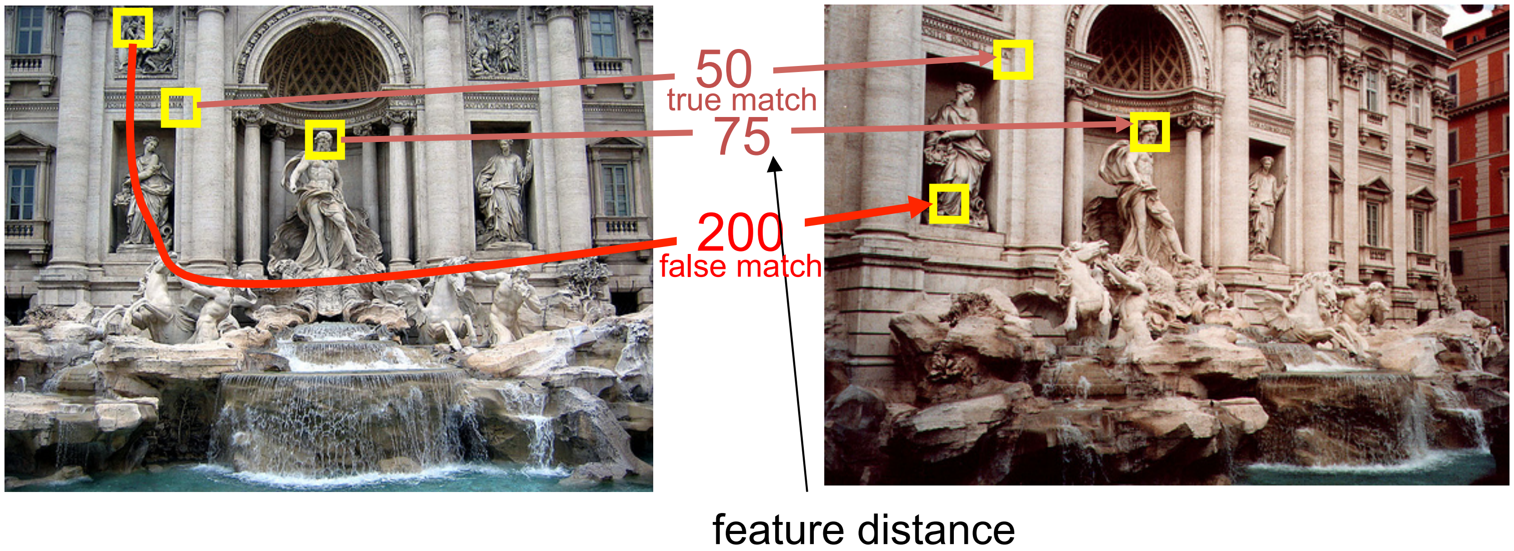



$I_1$




$I_2$

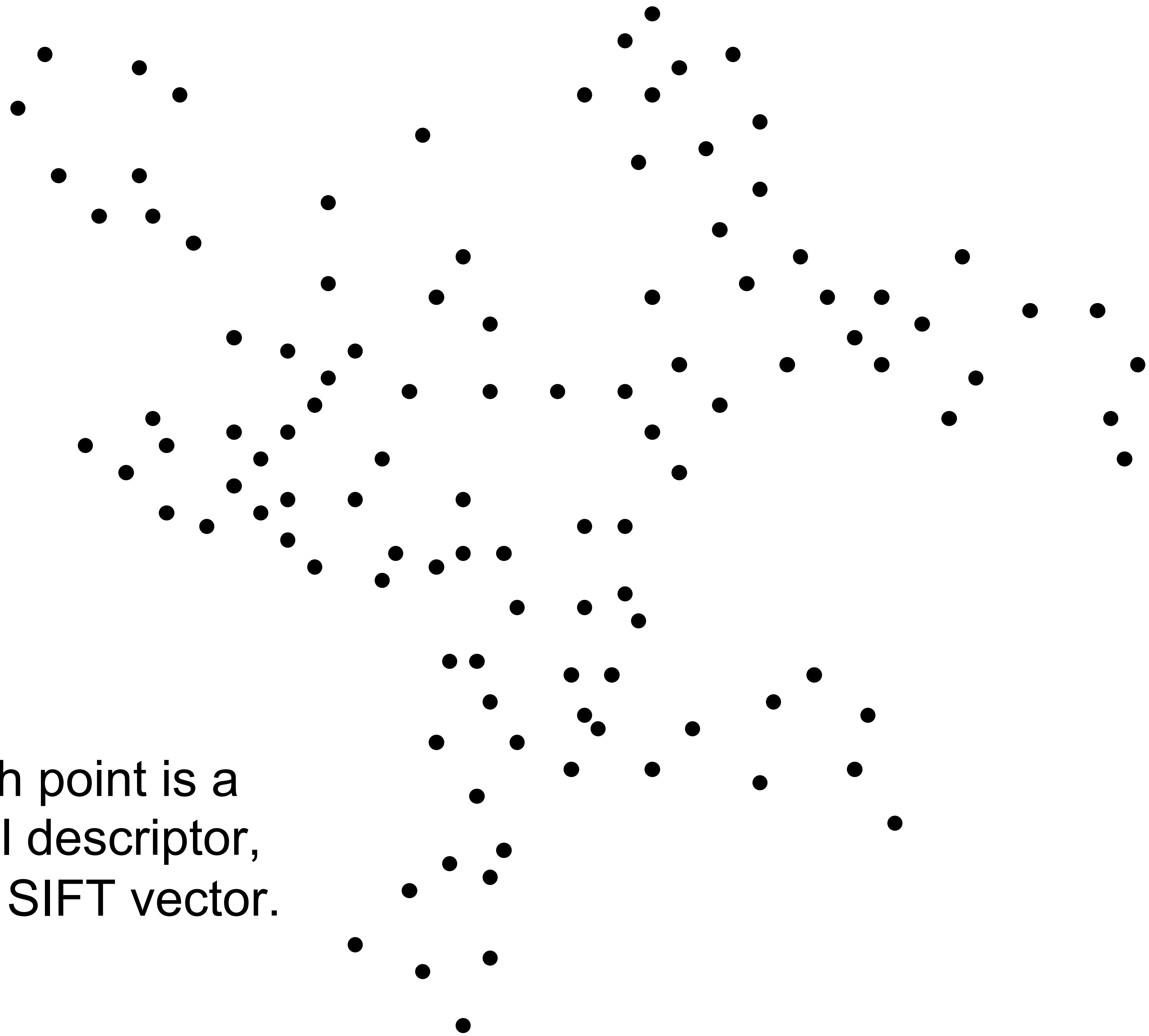
# Feature matching



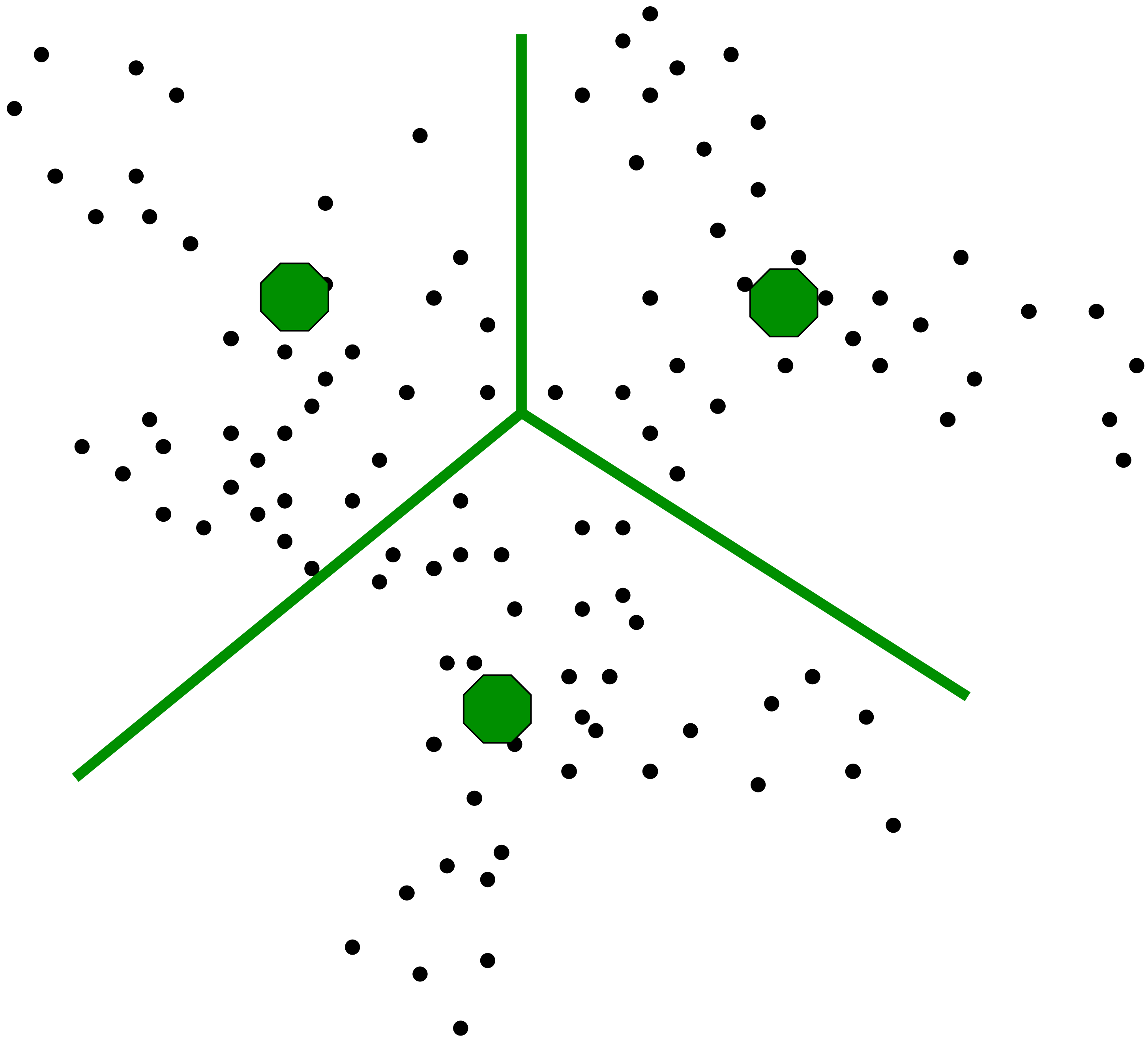

- Eliminate **bad matches**: throw out features with distance > threshold
- The distance threshold affects performance
  - True positives = # of detected matches that are correct
    - Suppose we want to maximize these—how to choose threshold?
  - False positives = # of detected matches that are incorrect
    - Suppose we want to minimize these—how to choose threshold?

# Category recognition

- Feature matching of local descriptors allows us to compare two images and find **instances** of objects
- What if we want to classify objects in a given image based on their **category** (e.g. person, car, plane, etc.)
  - Classifiers such as SVMs can be used for this task
  - The model is trained using the features extracted from previously labeled examples
  - But, how can we incorporate in a single feature vector, an unknown number of interest points and their description?

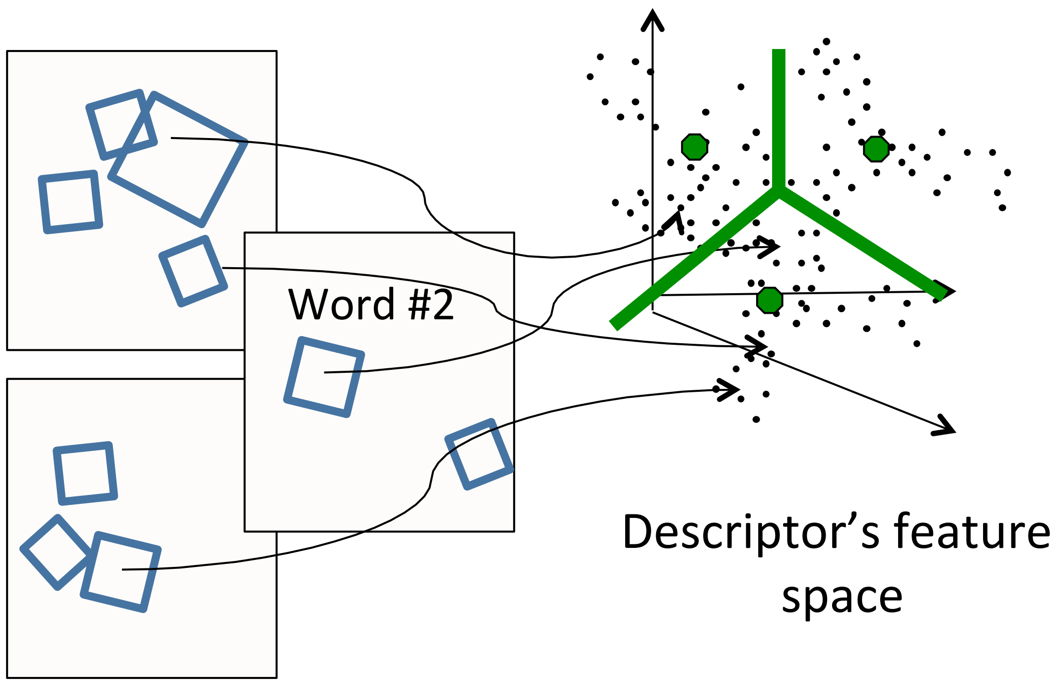


Each point is a local descriptor, e.g. SIFT vector.



# Visual words

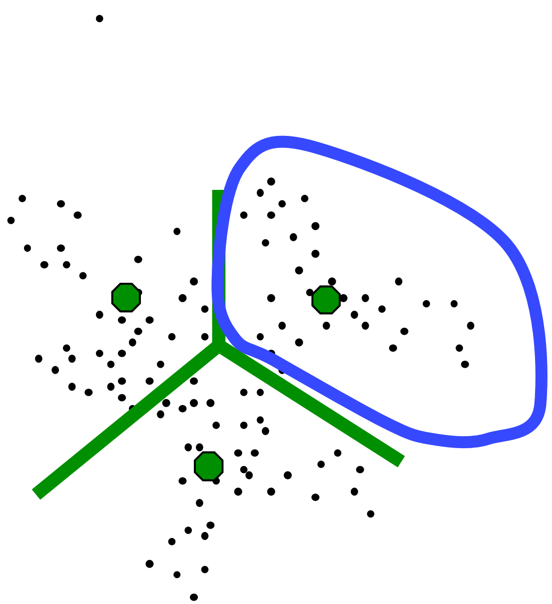
- Map high-dimensional descriptors to words by quantizing the feature space




- Quantize via clustering, let cluster centers be the prototype "words"
- Determine which word to assign to each new image region by finding the closest cluster center.

# Visual words

**Example**: each group of patches belongs to the same visual word



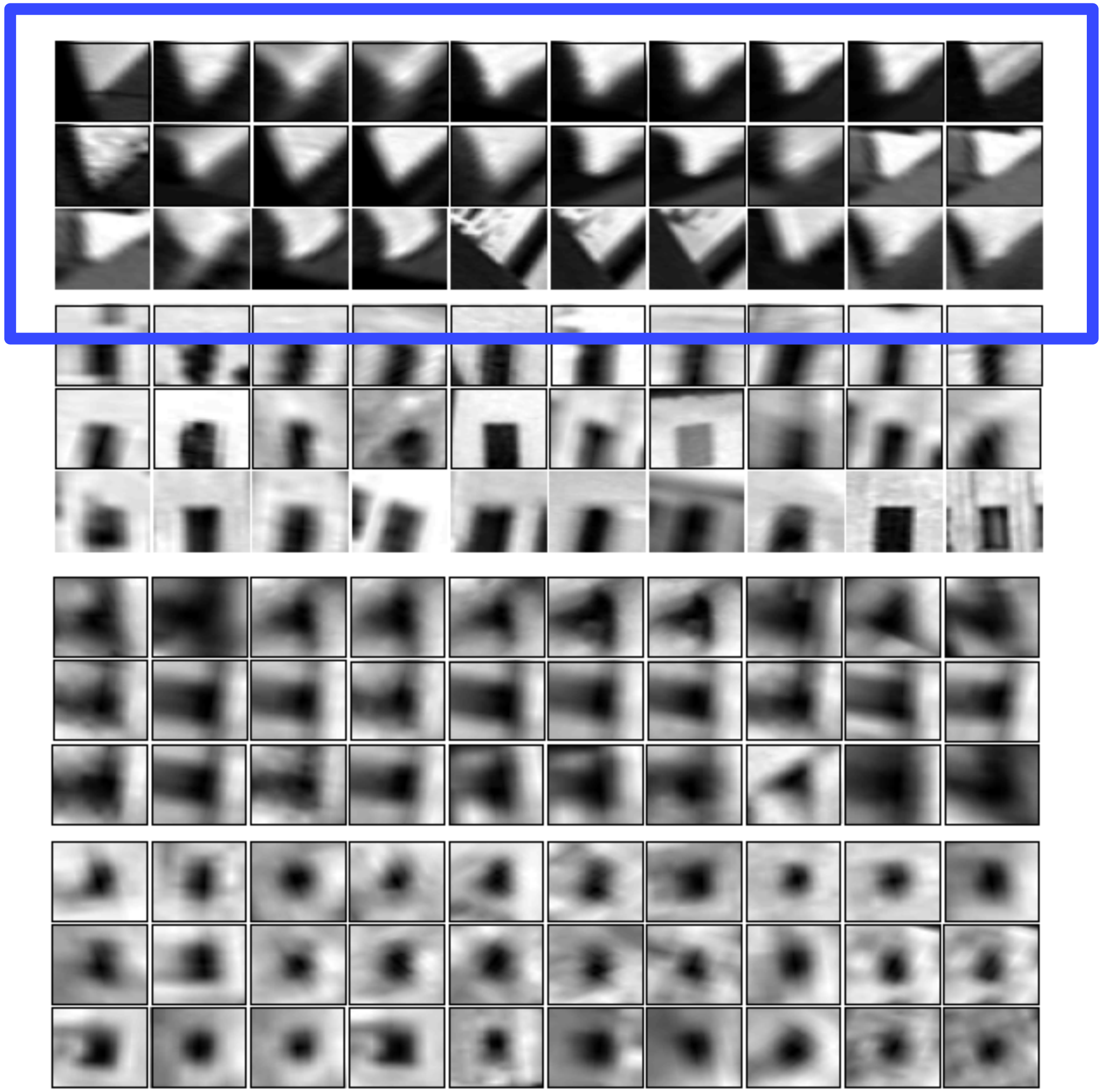


Figure from Sivic & Zisserman, ICCV 2003

# Indexing local features

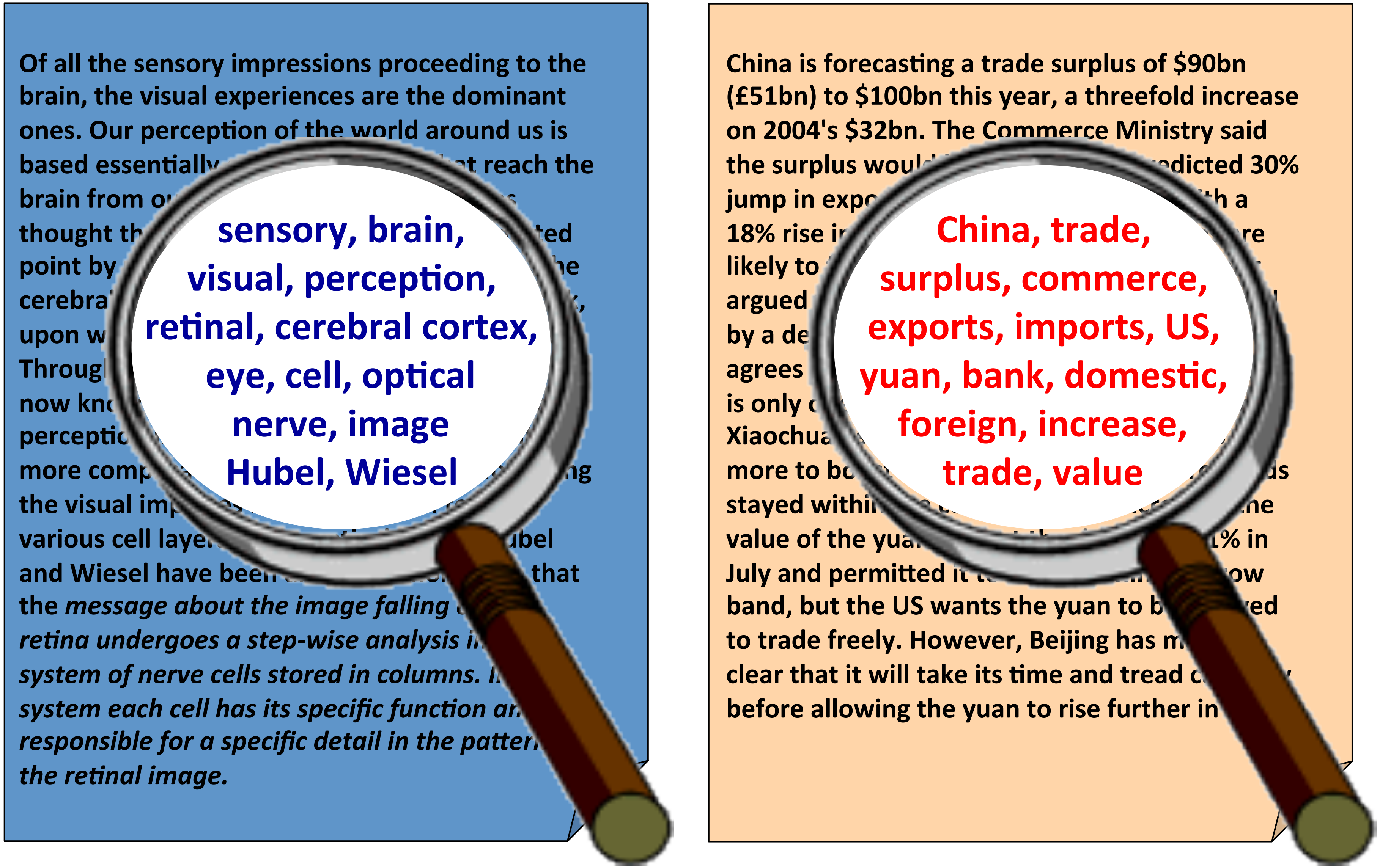
Index

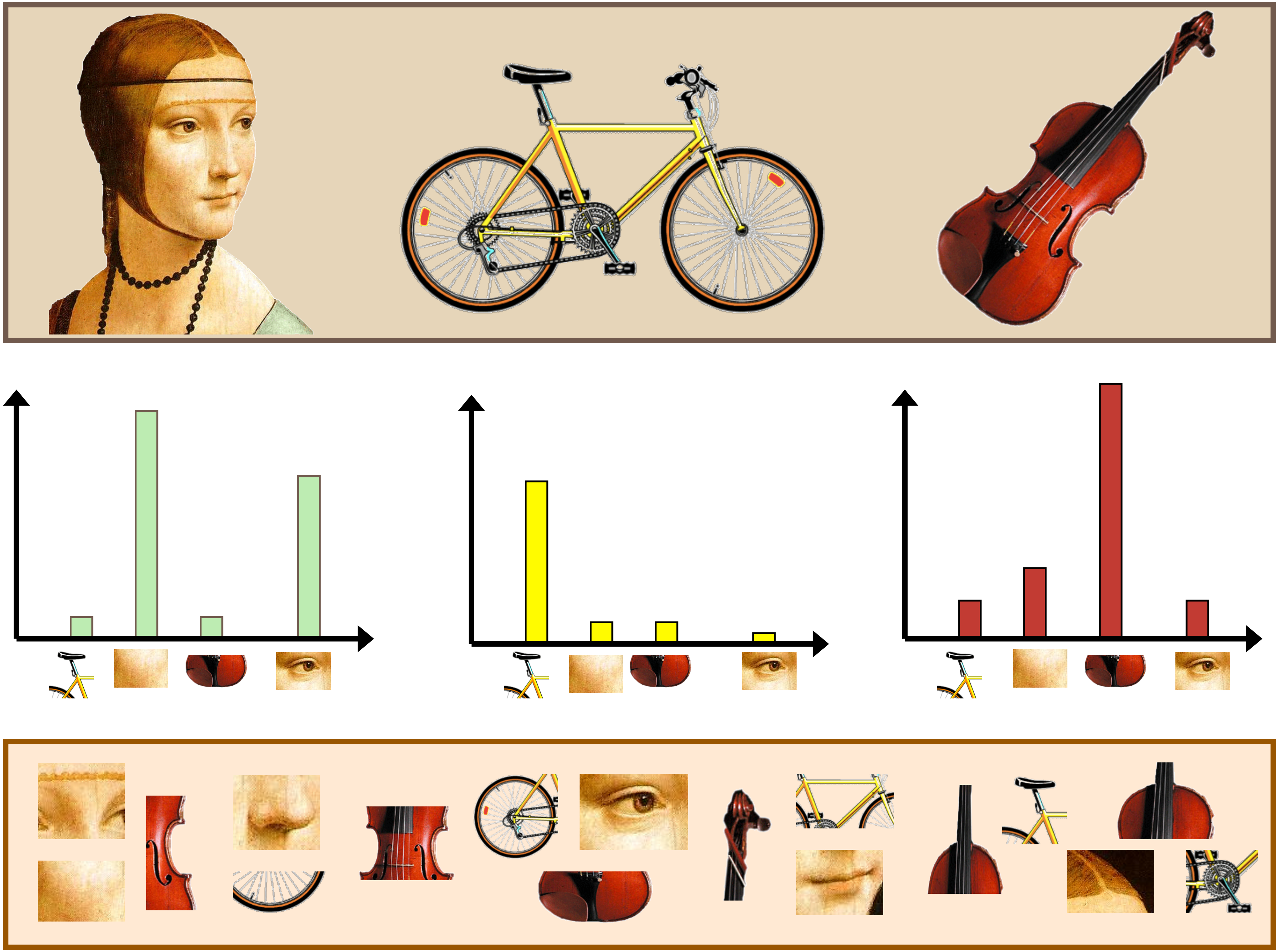


- **Inverted file index**
- For text documents, an efficient way to find all *pages* on which a *word* occurs is to use an index…
- We want to find all *images* in which a *feature* occurs.
- To use this idea, we'll need to map our features to "visual words".

Kristen Grauman

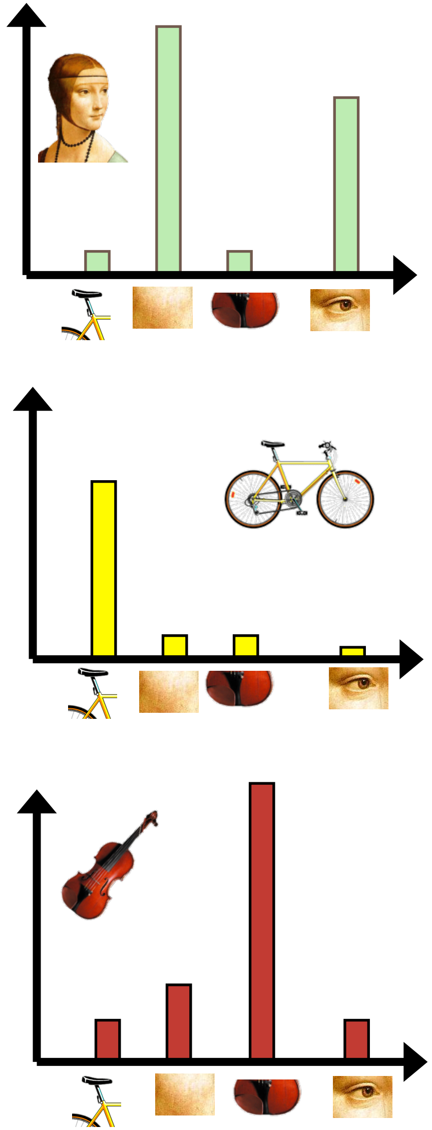
# Analogy to documents





# Bags of visual words

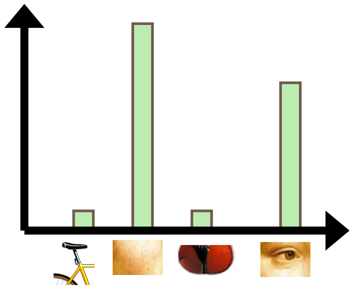
- Summarize entire image based on its distribution (histogram) of word occurrences.
- Analogous to bag of words representation commonly used for documents.



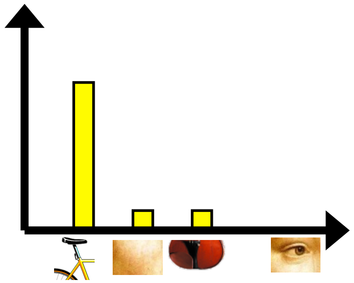
# Bags of visual words

- Rank frames by normalized scalar product between their (possibly weighted) occurrence counts--*nearest neighbor* search for similar images.

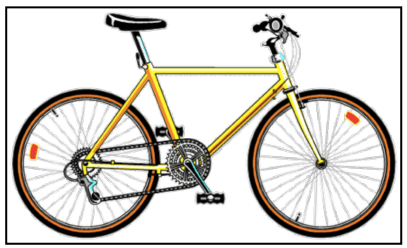
**[1 8 1 4]**



**[5 1 1 0]**



$\vec{d}_j$



$\vec{q}$

$$sim(d_j, q) = \frac{\langle d_j, q \rangle}{\|d_j\| \|q\|}$$

$$= \frac{\sum_{i=1}^{V} d_j(i) * q(i)}{\sqrt{\sum_{i=1}^{V} d_j(i)^2} * \sqrt{\sum_{i=1}^{V} q(i)^2}}$$

**for vocabulary of *V* words**

# Inverted file index and Bag of words


New query image

| Word # | Image # |
|---|---|
| 1 | 3 |
| 2 | |
| ... | |
| 7 | 1, 2 |
| 8 | 3 |
| 9 | |
| 10 | |
| ... | |
| 91 | 2 |
| ⋮ | ⋮ |

...


Image #1


Image #2

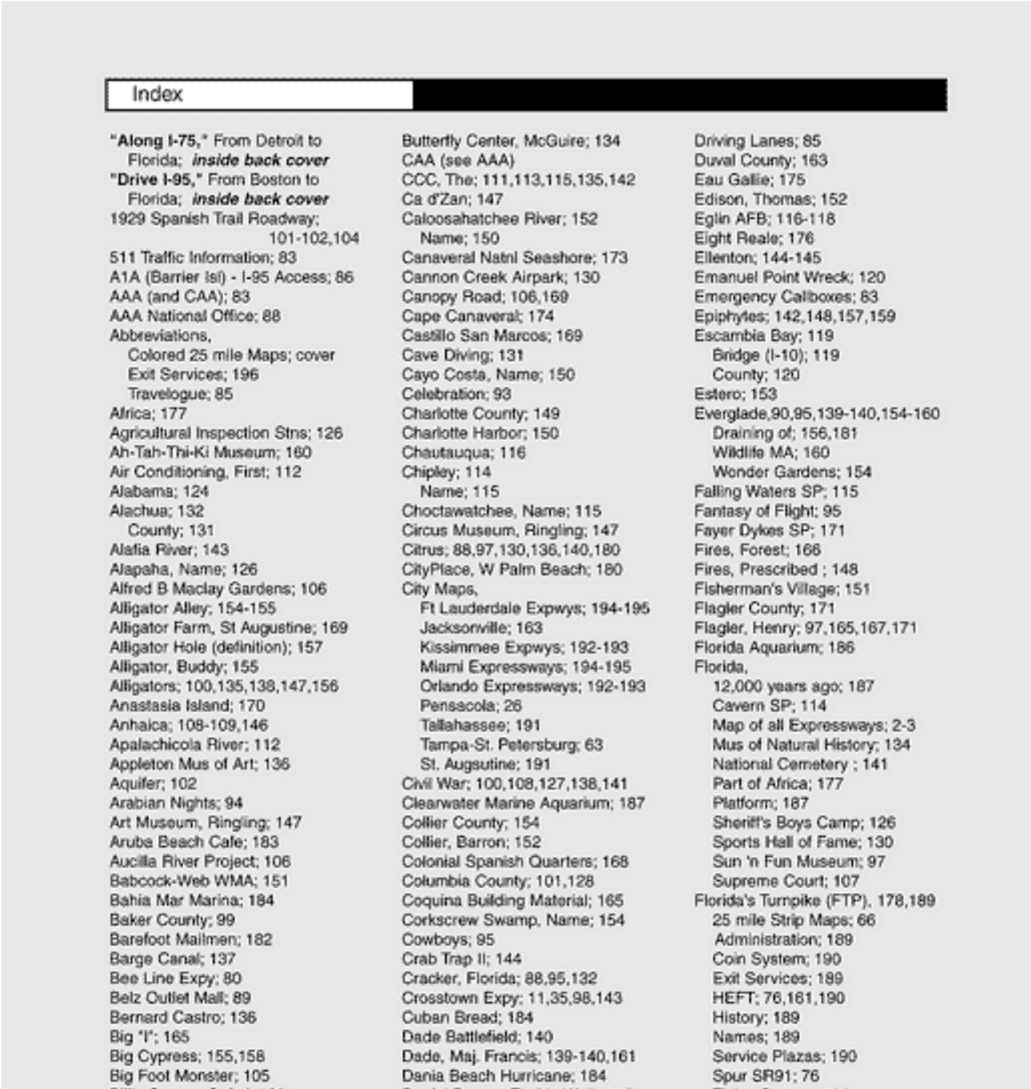
1. Extract words in query
2. Inverted file index to find relevant frames
3. Compare word counts

Kristen Grauman

# Large-scale image search

- Build the database:
  - Extract features from the database images
  - Learn a vocabulary using k-means (typical k: 100,000)
  - Compute *weights* for each word
  - Create an inverted file mapping words → images

# What else can we borrow from text retrieval?



Index



# Weighting the words

- Just as with text, some visual words are more discriminative than others

***the, and, or*** **vs.** ***cow, AT&T, Cher***

- the bigger fraction of the documents a word appears in, the less useful it is for matching
  - e.g., a word that appears in *all* documents is not helping us

# *tf-idf* weighting

- **T**erm **f**requency – **i**nverse **d**ocument **f**requency
- Describe frame by frequency of each word within it, downweight words that appear often in the database
- (Standard weighting for text retrieval)

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i}$$

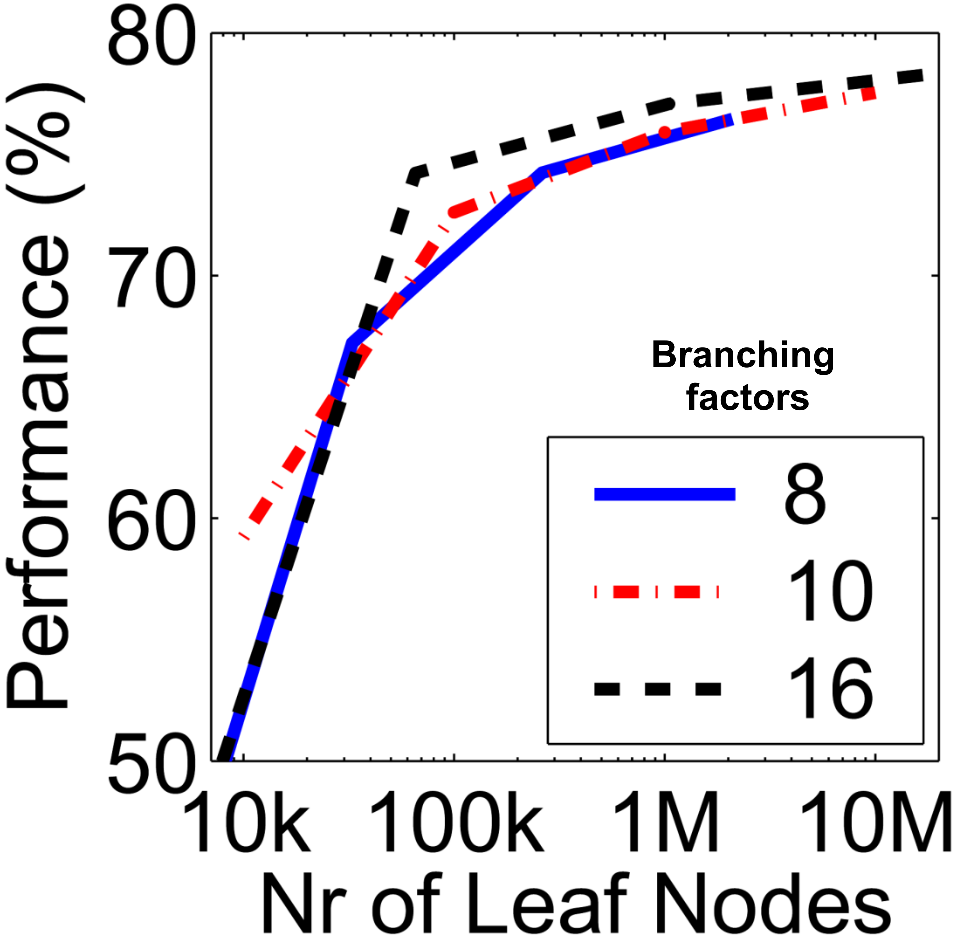
Number of occurrences of word i in document d

Number of words in document d

Total number of documents in database

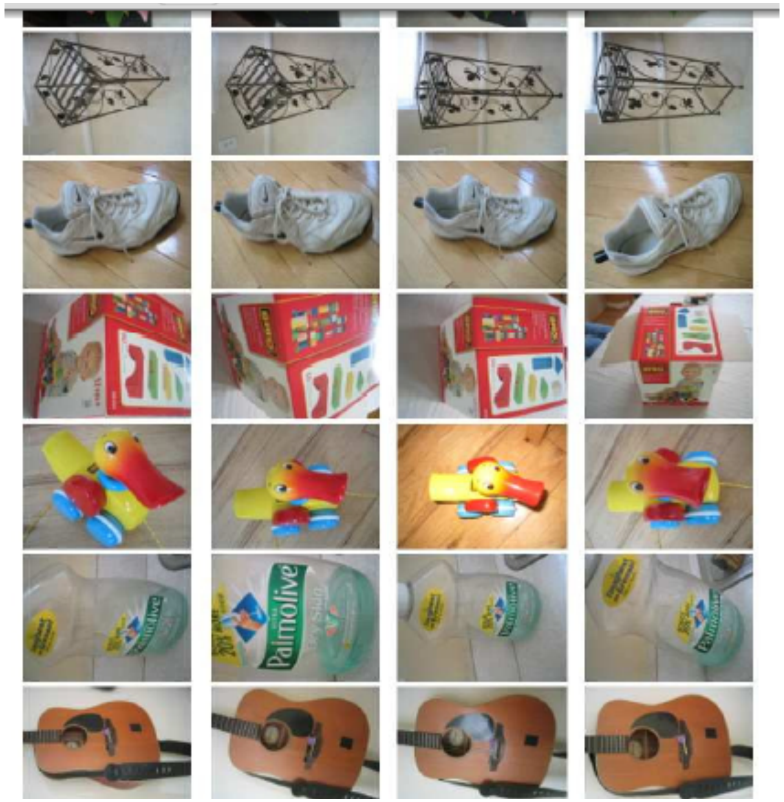
Number of documents word i occurs in, in whole database

# Vocabulary size




*Influence on performance, sparsity*

Results for recognition task with 6347 images

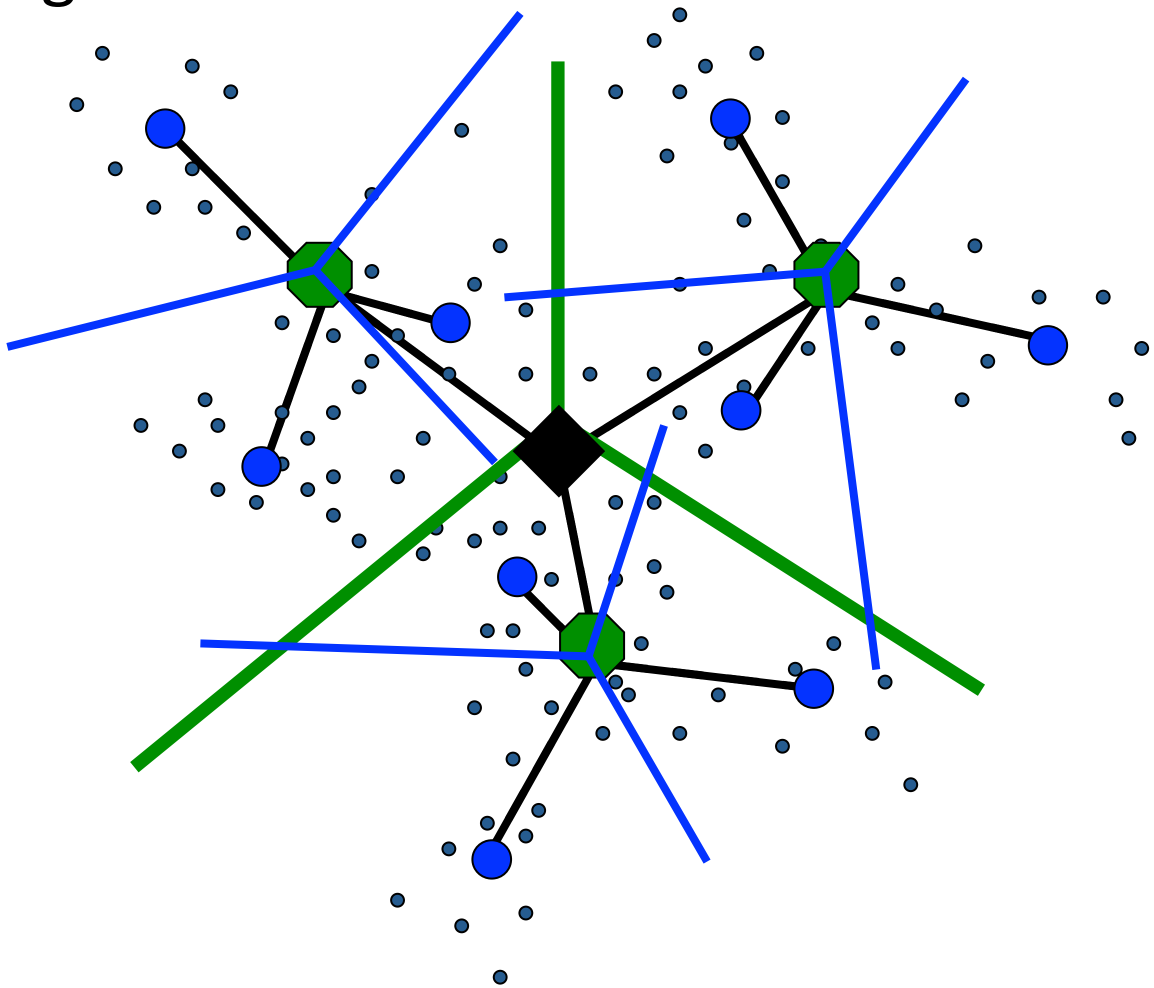



Nister & Stewenius, CVPR 2006

Kristen Grauman

# Vocabulary Trees

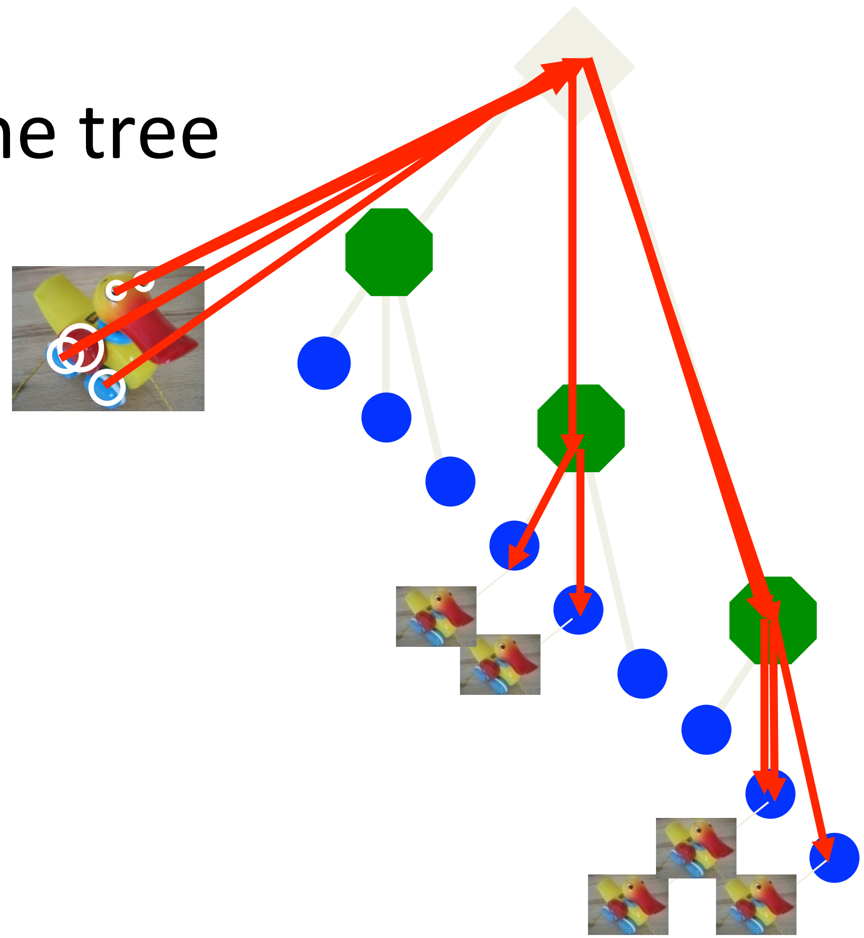
- Large vocabularies can be improved with hierarchical clustering



[Nister & Stewenius, CVPR'06]

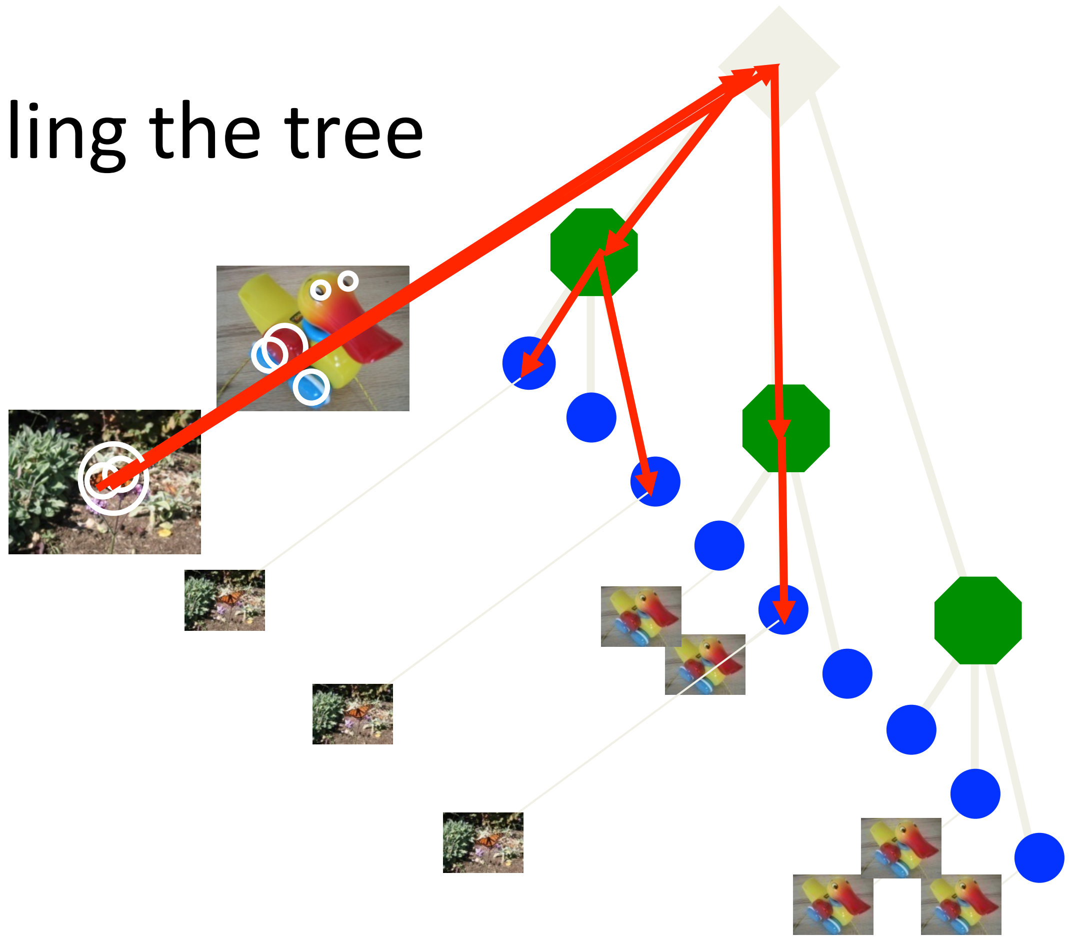
# Vocabulary Tree

- Training: Filling the tree



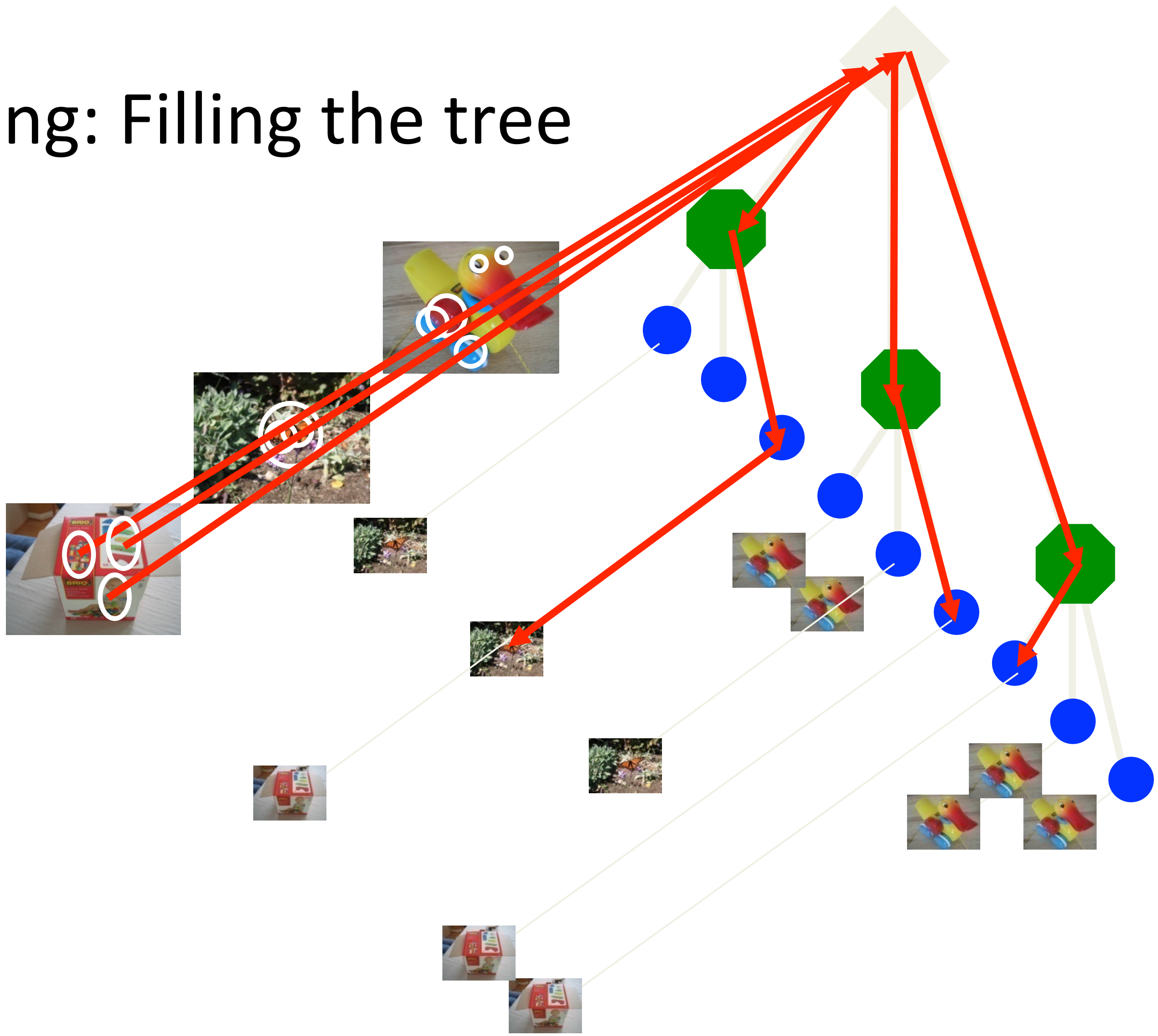
# Vocabulary Tree

- Training: Filling the tree



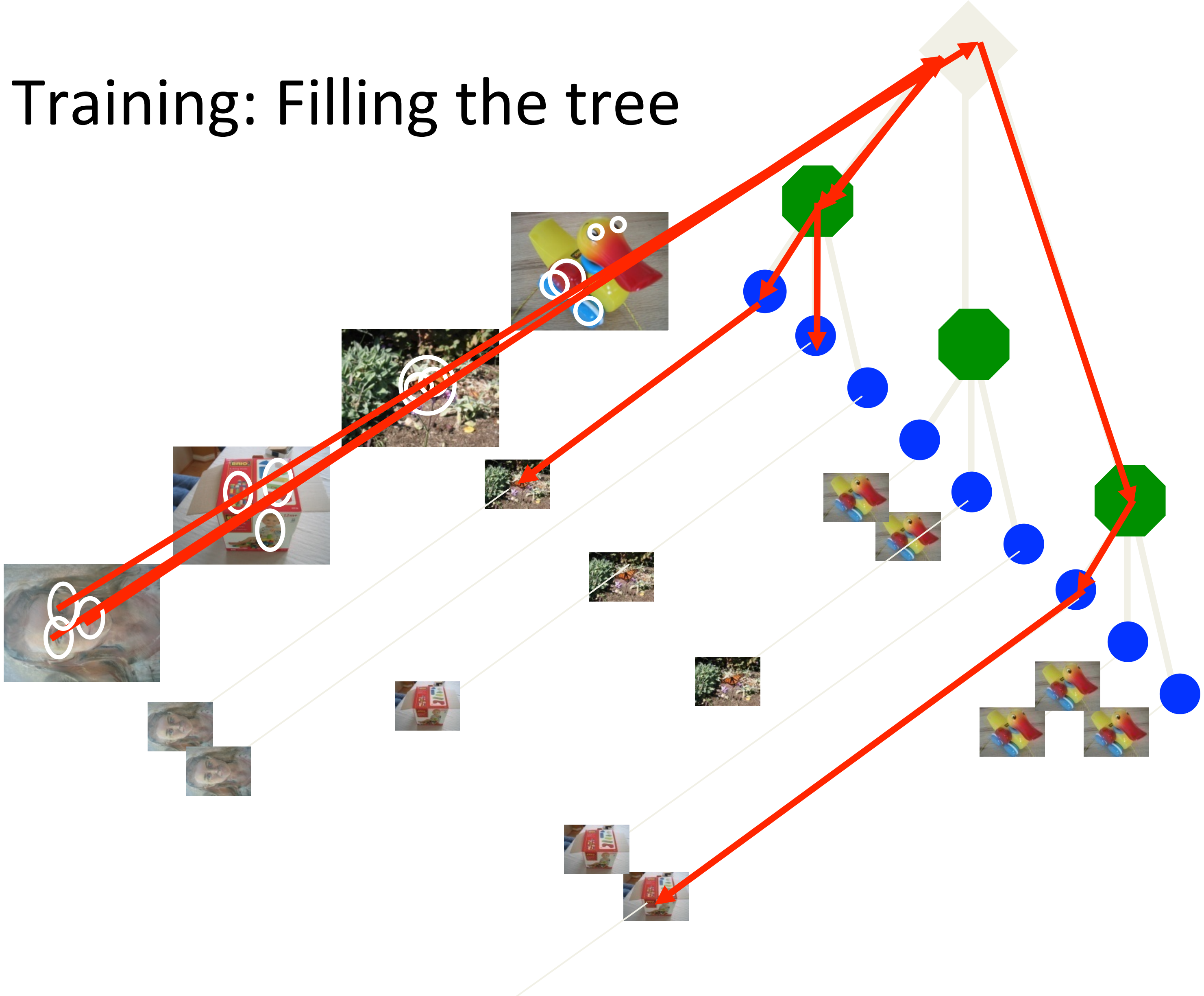
# Vocabulary Tree

- Training: Filling the tree



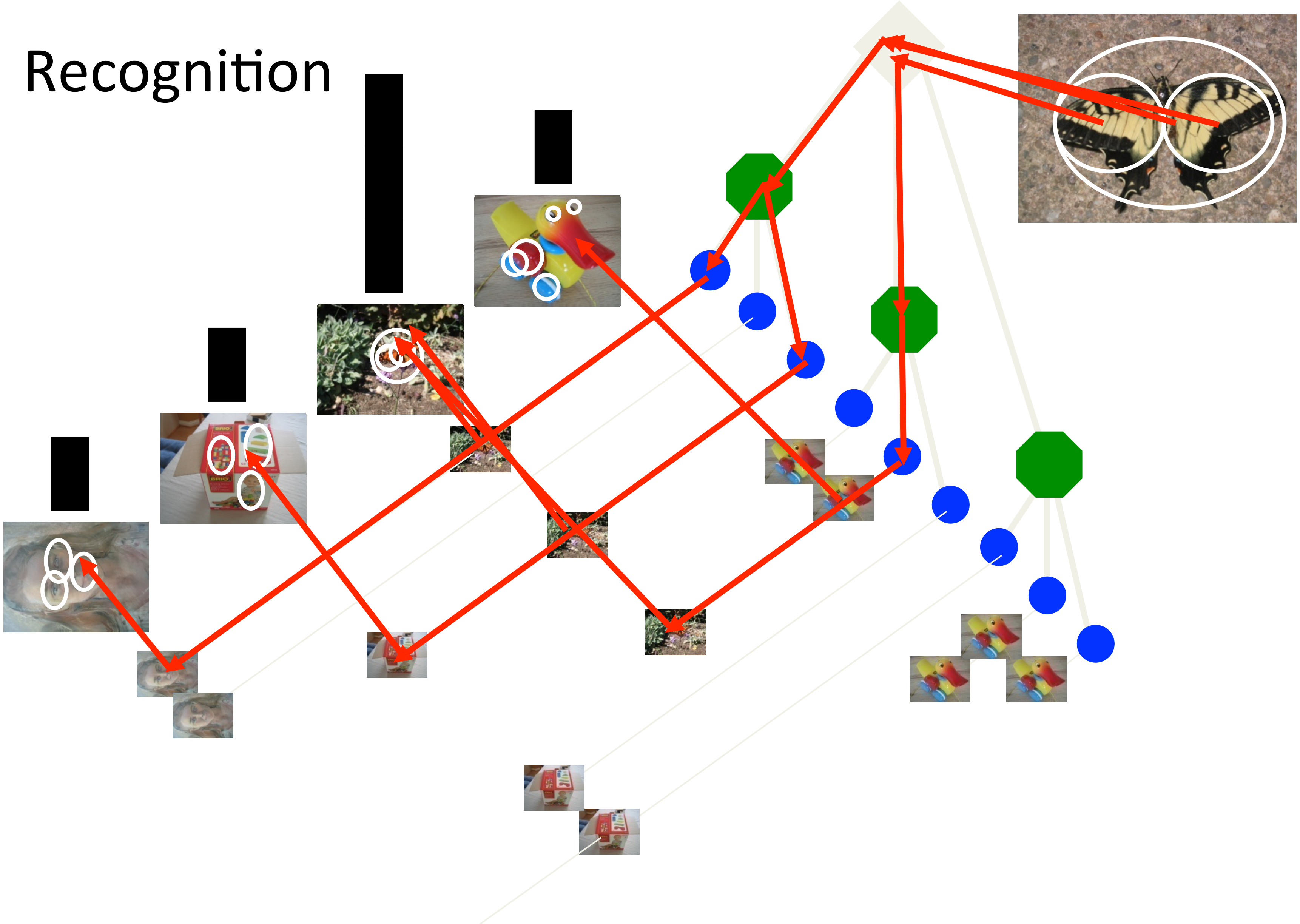
# Vocabulary Tree

- Training: Filling the tree



# Vocabulary Tree

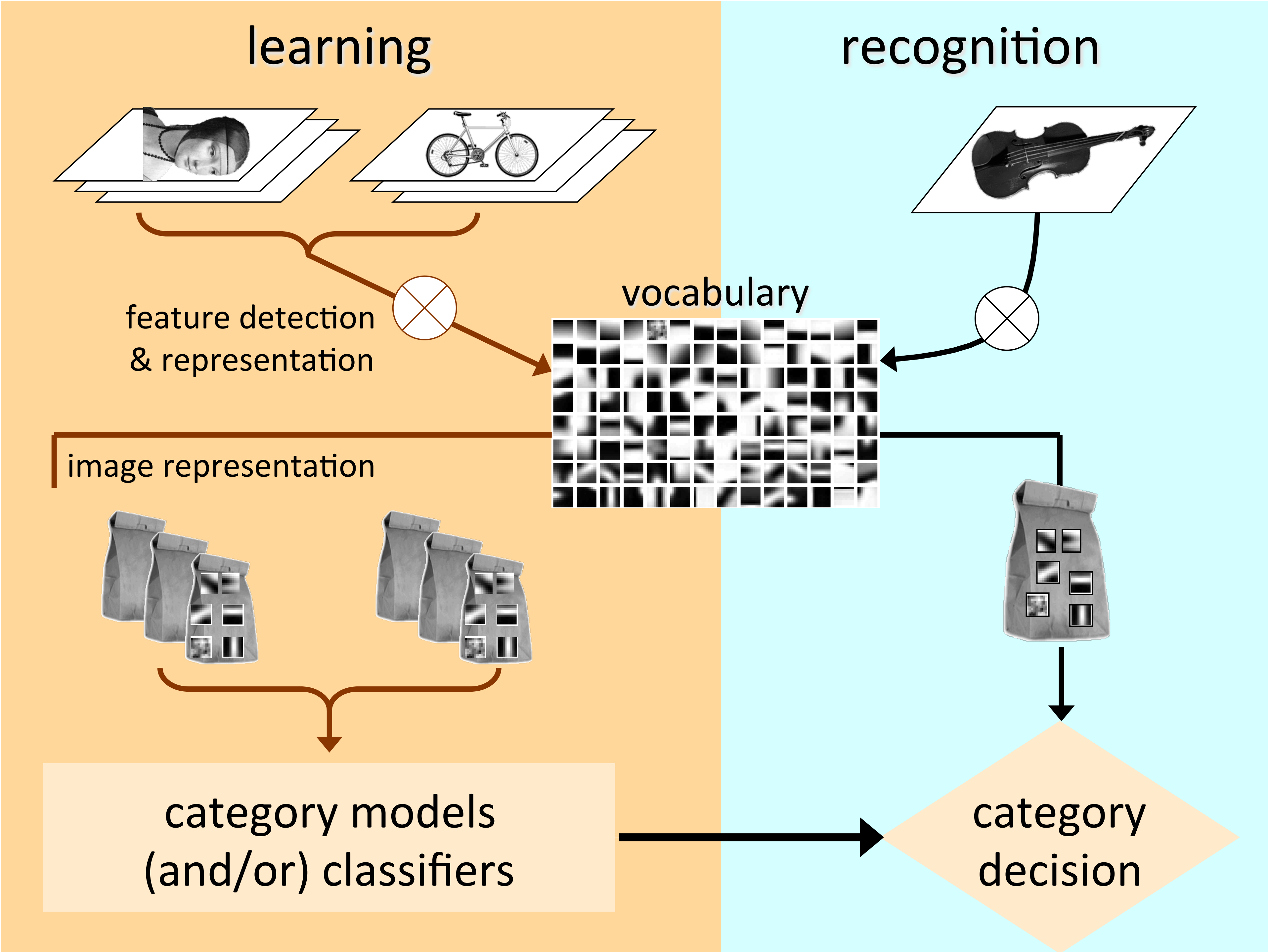
- Recognition



# Vocabulary Tree

**Complexity** defined by:

- Number of words given by the tree parameters:
  - branching factor and number of levels
- Word assignment cost vs. flat vocabulary


learning
recognition
feature detection
& representation
vocabulary
image representation
category models
(and/or) classifiers
category
decision

# Visual words/bags of words

- Advantages
  - flexible to geometry / deformations / viewpoint
  - compact summary of image content
  - provides vector representation for sets
  - very good results in practice

- Disadvantages
  - background and foreground mixed when bag covers whole image
  - optimal vocabulary formation remains unclear
  - basic model ignores geometry – must verify afterwards, or encode via features

# References


- Kristen Grauman, Local Invariant Features, http://www.cs.utexas.edu/~grauman/courses/spring2011/slides/lecture13_localfeats.pdf
- Pedro Quelhas, Pattern Recognition for Computer Vision, http://www.dcc.fc.up.pt/~mcoimbra/lectures/MAPI_1011/CV_1011_9_PatternRecognitionConcepts.pdf
- Alyosha Efros, What should be done at the low level?, http://www.cs.cmu.edu/~efros/courses/LBMV12/LowLevel.ppt
- Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
- David A. Forsyth and Jean Ponce, Computer Vision: A Modern Approach, Prentice Hall, 2011 (2nd edition)