Computer Vision – Lecture 13

Indexing and Visual Vocabularies

12.12.2016

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de

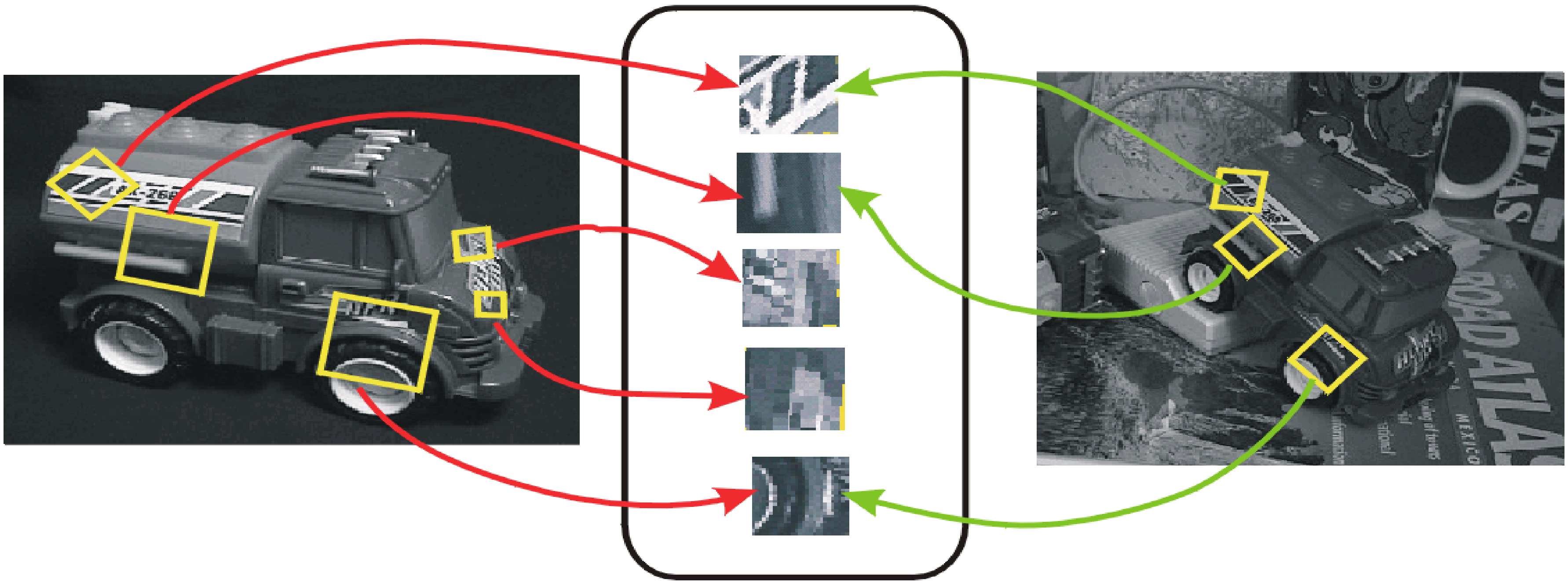
leibe@vision.rwth-aachen.de

Course Outline

- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Object Categorization I
  - Sliding Window based Object Detection
- Local Features & Matching
  - Local Features - Detection and Description
  - Recognition with Local Features
  - Indexing & Visual Vocabularies
- Object Categorization II
  - Bag-of-Words Approaches & Part-based Approaches
- 3D Reconstruction

Recap: Recognition with Local Features

- Image content is transformed into local features that are invariant to translation, rotation, and scale
- Goal: Verify if they belong to a consistent configuration

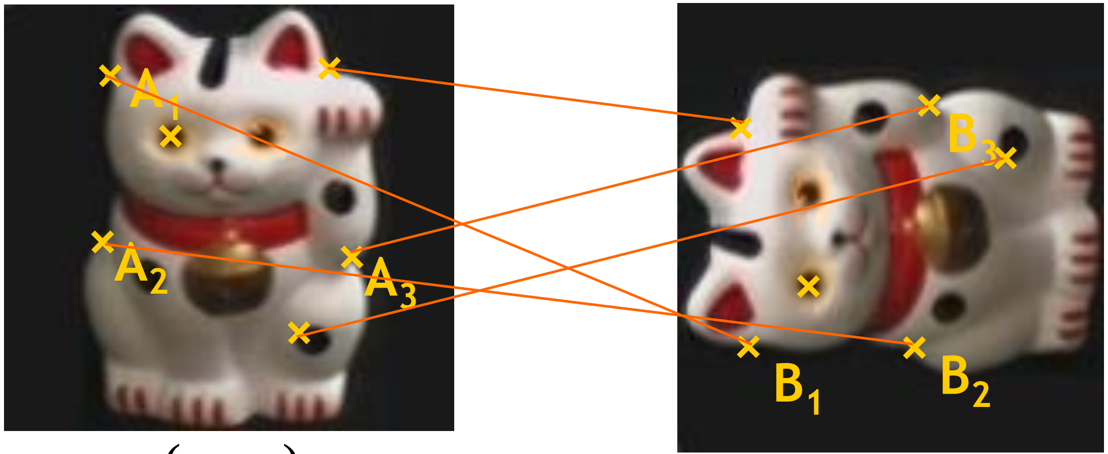


Local Features,
e.g. SIFT

Slide credit: David Lowe

Recap: Fitting an Affine Transformation

- Assuming we know the correspondences, how do we get the transformation?



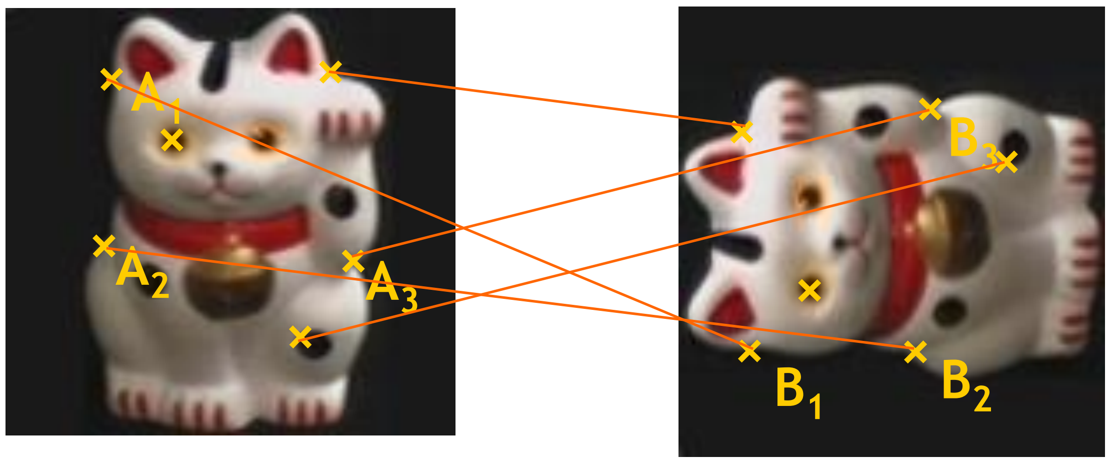
(x_i, y_i) (x_i', y_i')

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

Recap: Fitting a Homography

- Estimating the transformation



$$\mathbf{x}_{A_1} \leftrightarrow \mathbf{x}_{B_1}$$
$$\mathbf{x}_{A_2} \leftrightarrow \mathbf{x}_{B_2}$$
$$\mathbf{x}_{A_3} \leftrightarrow \mathbf{x}_{B_3}$$
$$\vdots$$

Homogenous coordinates

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Image coordinates

$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \frac{1}{z'} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

Matrix notation

$$x' = Hx$$

$$x'' = \frac{1}{z'} x'$$

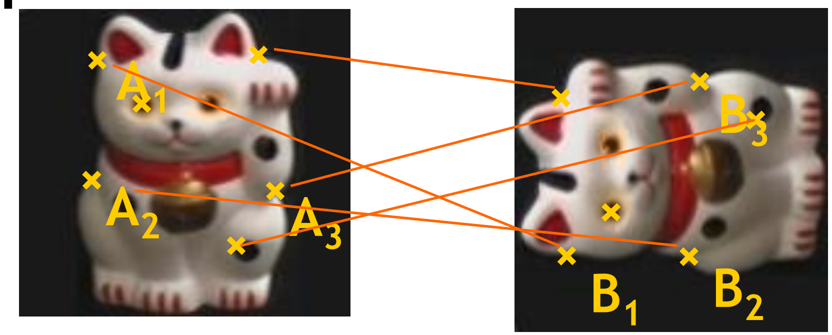
$$x_{A_1} = \frac{h_{11}\, x_{B_1} + h_{12} y_{B_1} + h_{13}}{h_{31}\, x_{B_1} + h_{32} y_{B_1} + 1} \qquad y_{A_1} = \frac{h_{21}\, x_{B_1} + h_{22} y_{B_1} + h_{23}}{h_{31}\, x_{B_1} + h_{32} y_{B_1} + 1}$$

Slide credit: Krystian Mikolajczyk

Recap: Fitting a Homography

- **Estimating the transformation**

$$\boxed{\begin{aligned} h_{11}\, x_{B_1} + h_{12} y_{B_1} + h_{13} - x_{A_1} h_{31}\, x_{B_1} - x_{A_1} h_{32} y_{B_1} - x_{A_1} &= 0 \\ h_{21}\, x_{B_1} + h_{22} y_{B_1} + h_{23} - y_{A_1} h_{31}\, x_{B_1} - y_{A_1} h_{32} y_{B_1} - y_{A_1} &= 0 \end{aligned}}$$



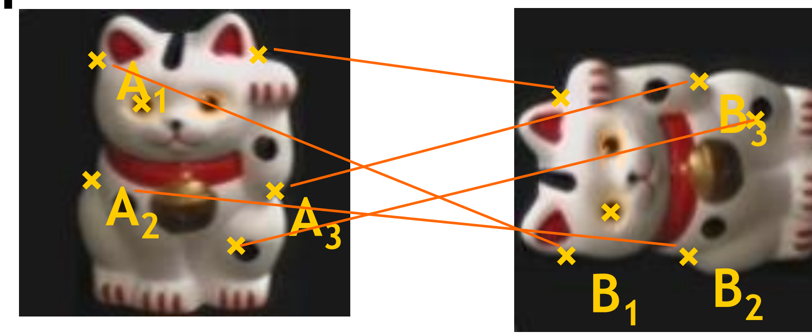
$$\mathbf{x}_{A_1} \leftrightarrow \mathbf{x}_{B_1}$$
$$\mathbf{x}_{A_2} \leftrightarrow \mathbf{x}_{B_2}$$
$$\mathbf{x}_{A_3} \leftrightarrow \mathbf{x}_{B_3}$$
$$\vdots$$

$$\begin{bmatrix} x_{B_1} & y_{B_1} & 1 & 0 & 0 & 0 & -x_{A_1} x_{B_1} & -x_{A_1} y_{B_1} & -x_{A_1} \\ 0 & 0 & 0 & x_{B_1} & y_{B_1} & 1 & -y_{A_1} x_{B_1} & -y_{A_1} y_{B_1} & -y_{A_1} \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \end{bmatrix} \cdot \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ . \end{bmatrix}$$

$$Ah = 0$$

Recap: Fitting a Homography

- **Estimating the transformation**
- **Solution:**
  - Null-space vector of A
  - Corresponds to smallest eigenvector



$$Ah = 0$$

$$\mathbf{x}_{A_1} \leftrightarrow \mathbf{x}_{B_1}$$
$$\mathbf{x}_{A_2} \leftrightarrow \mathbf{x}_{B_2}$$
$$\mathbf{x}_{A_3} \leftrightarrow \mathbf{x}_{B_3}$$
$$\vdots$$

SVD

$$\mathbf{A} = \mathbf{UDV}^T = \mathbf{U}\begin{bmatrix} d_{11} & \cdots & d_{19} \\ \vdots & \ddots & \vdots \\ d_{91} & \cdots & d_{99} \end{bmatrix}\begin{bmatrix} v_{11} & \cdots & \boxed{v_{19}} \\ \vdots & \ddots & \boxed{\vdots} \\ v_{91} & \cdots & \boxed{v_{99}} \end{bmatrix}^T$$

$$\boxed{\mathbf{h} = \frac{[v_{19}, \cdots, v_{99}]}{v_{99}}}$$

Minimizes least square error

Recap: Object Recognition by Alignment

- Assumption
  - Known object, rigid transformation compared to model image

  ⇒ *If we can find evidence for such a transformation, we have recognized the object.*

- You learned methods for
  - Fitting an *affine transformation* from ≥ 3 correspondences
  - Fitting a *homography* from ≥ 4 correspondences

Affine: solve a system

$$At = b$$

Homography: solve a system

$$Ah = 0$$

- Correspondences may be noisy and may contain outliers

  ⇒ Need to use robust methods that can filter out outliers

Recap: Robust Estimation with RANSAC

RANSAC loop:

1. Randomly select a *seed group* of points on which to base transformation estimate (e.g., a group of matches)
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers

- Keep the transformation with the largest number of inliers

Slide credit: Kristen Grauman

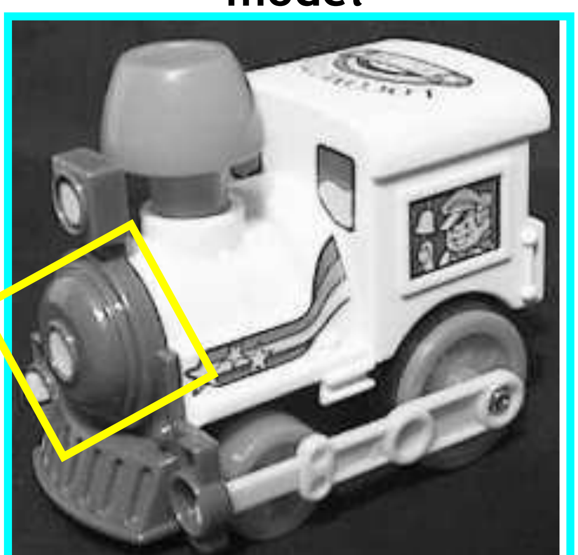
Problem with RANSAC

- In many practical situations, the percentage of outliers (incorrect putative matches) is often very high (90% or above).
- Alternative strategy: Generalized Hough Transform

Strategy 2: Generalized Hough Transform

- Suppose our features are scale- and rotation-invariant
  - Then a single feature match provides an alignment hypothesis (translation, scale, orientation).

model

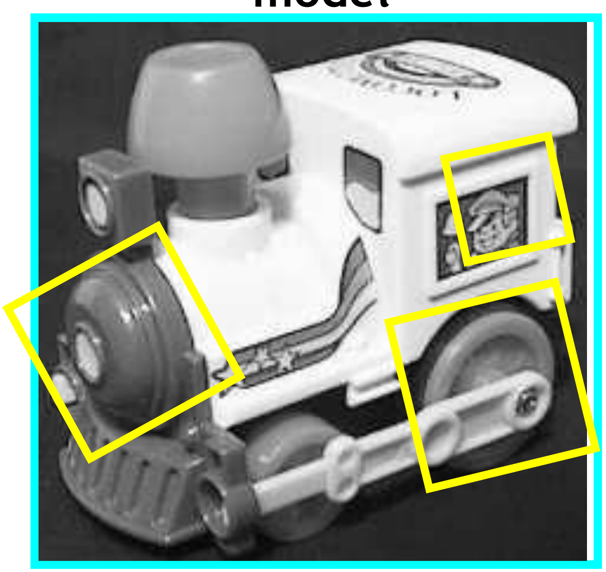


Slide credit: Svetlana Lazebnik

Strategy 2: Generalized Hough Transform

- Suppose our features are scale- and rotation-invariant
  - Then a single feature match provides an alignment hypothesis (translation, scale, orientation).
  - Of course, a hypothesis from a single match is unreliable.
  - Solution: let each match vote for its hypothesis in a Hough space with very coarse bins.

model



Slide credit: Svetlana Lazebnik

Pose Clustering and Verification with SIFT

- To detect instances of objects from a model base:

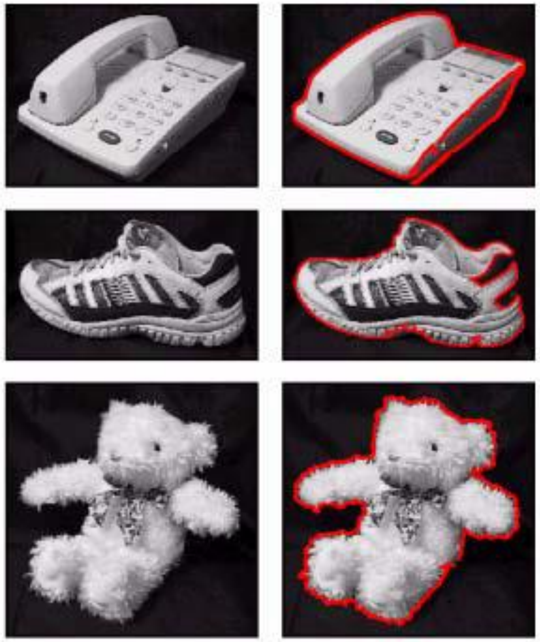
1. Index descriptors
   - Distinctive features narrow down possible matches

Pose Clustering and Verification with SIFT

- To detect instances of objects from a model base:

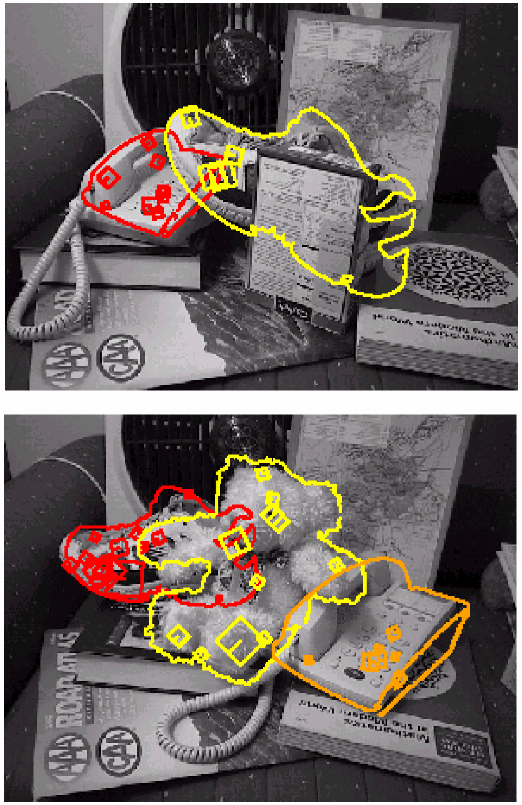
1. Index descriptors
   - Distinctive features narrow down possible matches
2. Generalized Hough transform to vote for poses
   - Keypoints have record of parameters relative to model coordinate system
3. Affine fit to check for agreement between model and image features
   - Fit and verify using features from Hough bins with 3+ votes

Object Recognition Results



Background subtract for model boundaries

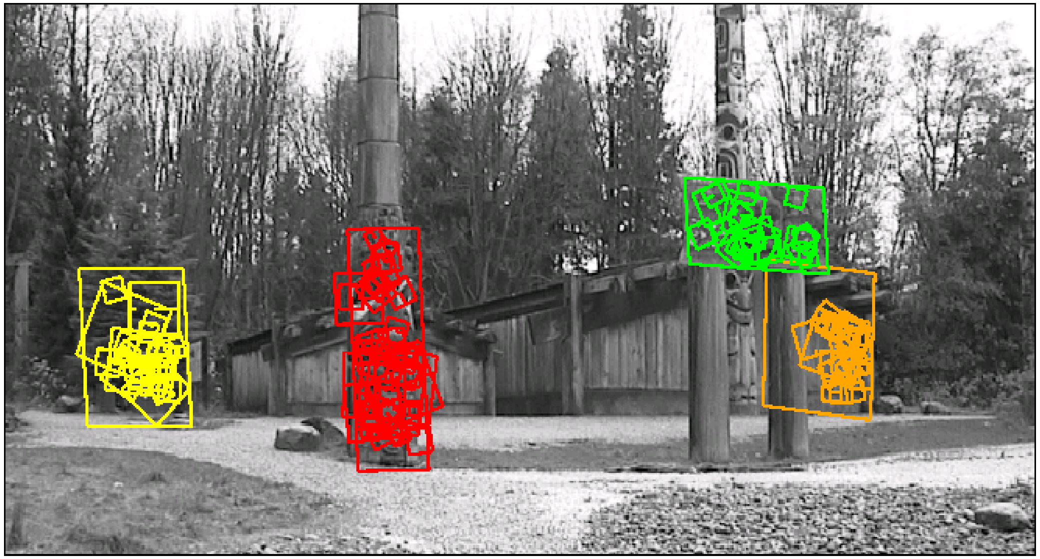
Objects recognized



Recognition in spite of occlusion

Location Recognition

Training



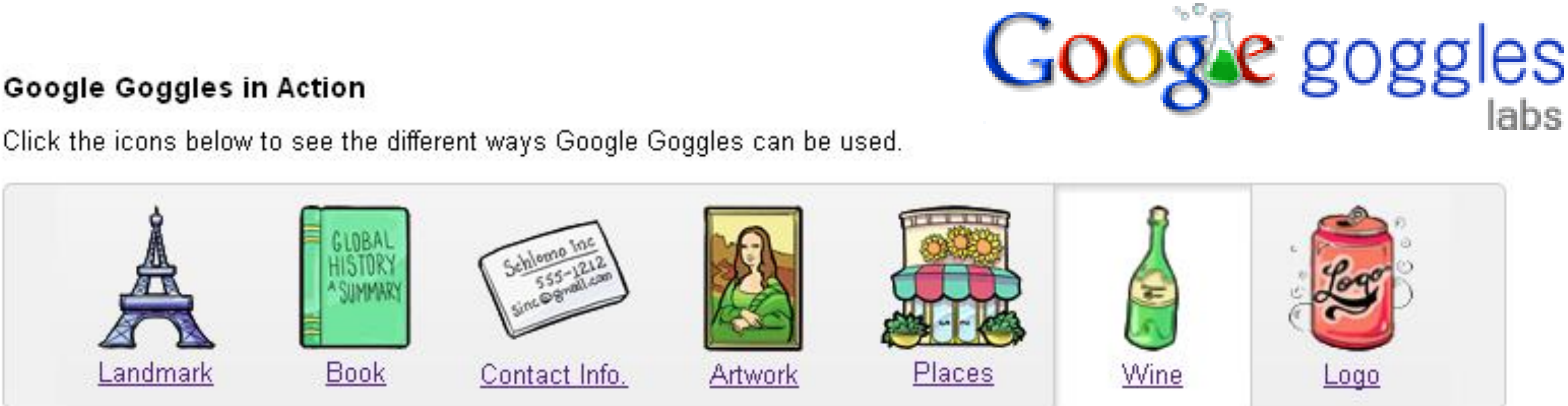
[Lowe, IJCV'04]

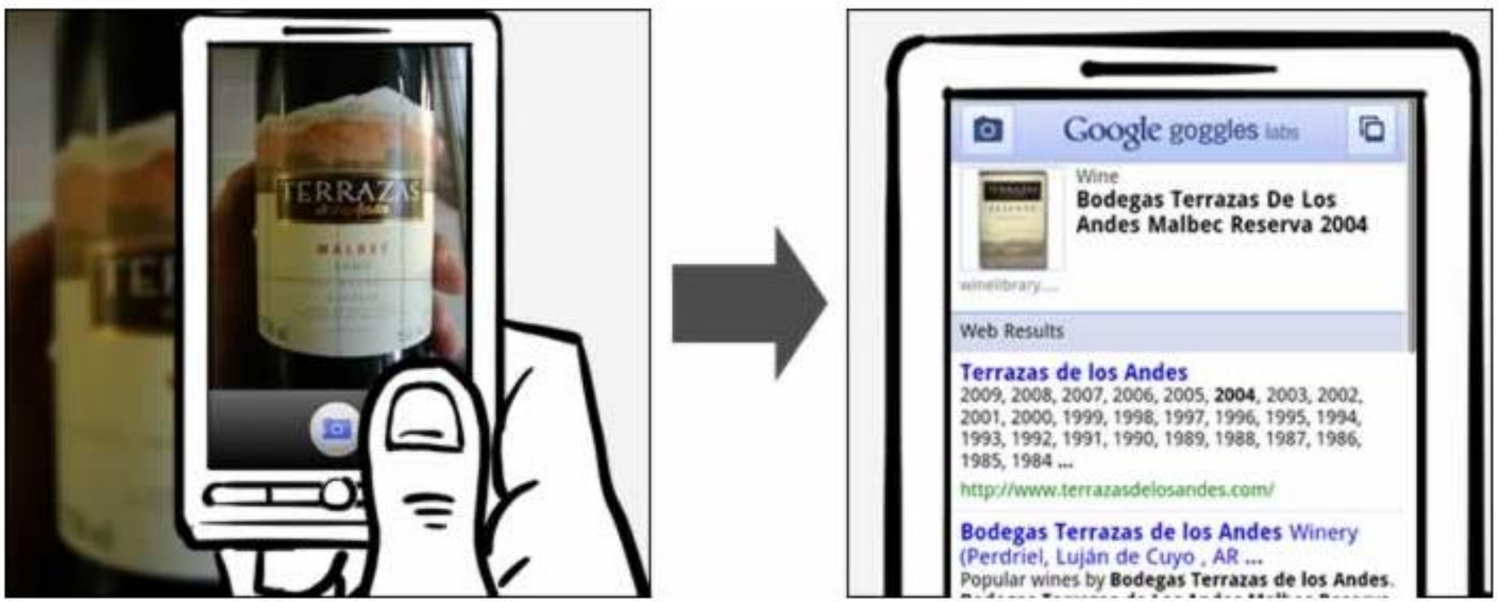
Slide credit: David Lowe

Topics of This Lecture

- **Indexing with Local Features**
  - Inverted file index
  - Visual Words
  - Visual Vocabulary construction
  - tf-idf weighting
- **Bag-of-Words Model**
  - Use for image classification

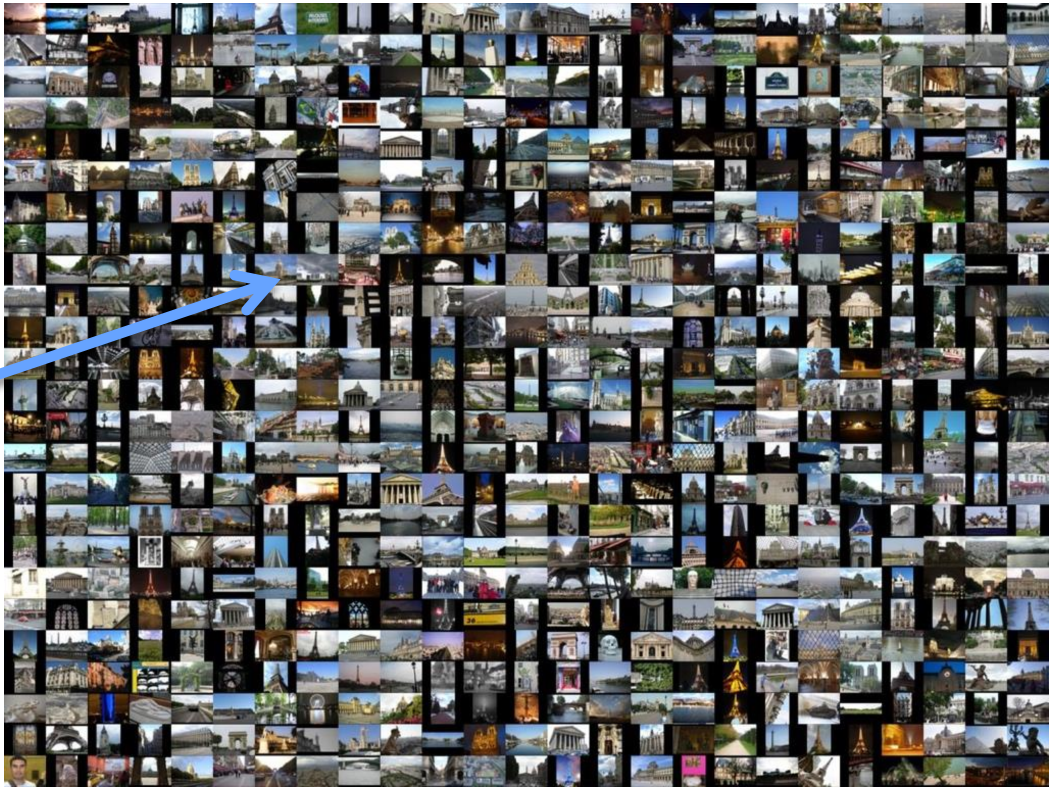
Application: Mobile Visual Search





- Take photos of objects as queries for visual search

Large-Scale Image Matching Problem



Database with thousands (millions) of images

- How can we perform this matching step efficiently?

Indexing Local Features

- Each patch / region has a descriptor, which is a point in some high-dimensional feature space (e.g., SIFT)

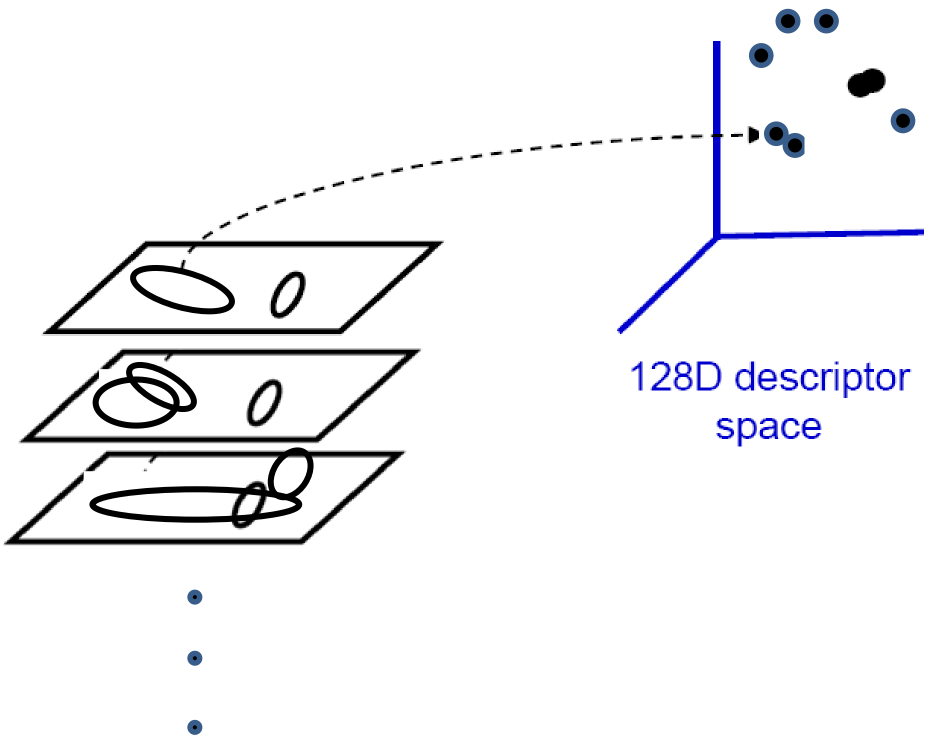
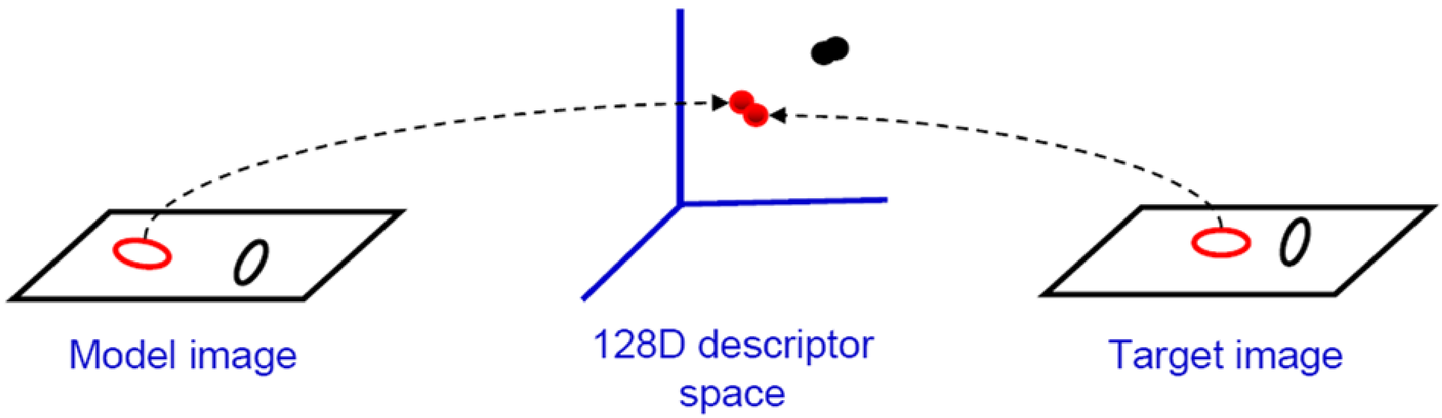


Figure credit: A. Zisserman

Indexing Local Features

- When we see close points in feature space, we have similar descriptors, which indicates similar local content.



- This is of interest for many applications
  - E.g. Image matching,
  - E.g. Retrieving images of similar objects,
  - E.g. Object recognition, categorization, 3d Reconstruction,…

Figure credit: A. Zisserman

Indexing Local Features

- With potentially thousands of features per image, and hundreds to millions of images to search, how to efficiently find those that are relevant to a new image?

- Low-dimensional descriptors (e.g. through PCA):
  - Can use standard efficient data structures for nearest neighbor search

- High-dimensional descriptors
  - Approximate nearest neighbor search methods more practical

- Inverted file indexing schemes

Indexing Local Features: Inverted File Index

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".

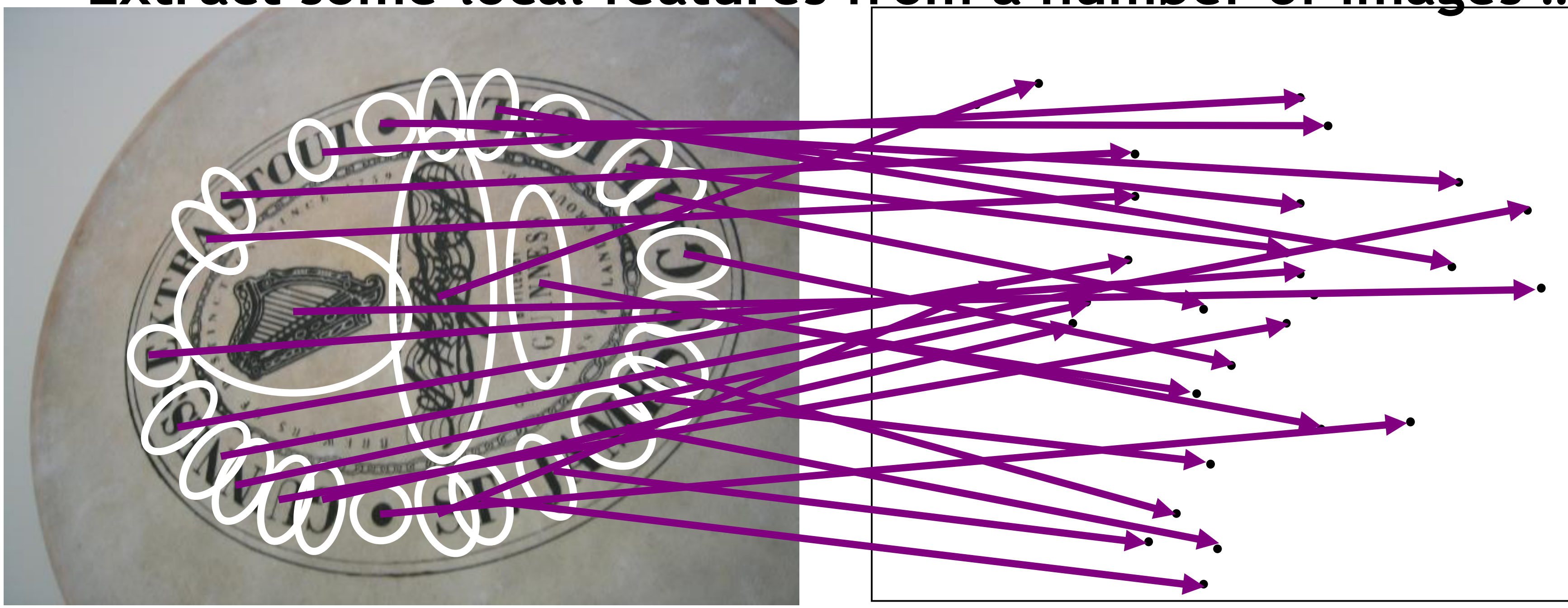
Text Retrieval vs. Image Search

- **What makes the problems similar, different?**

Slide credit: Kristen Grauman

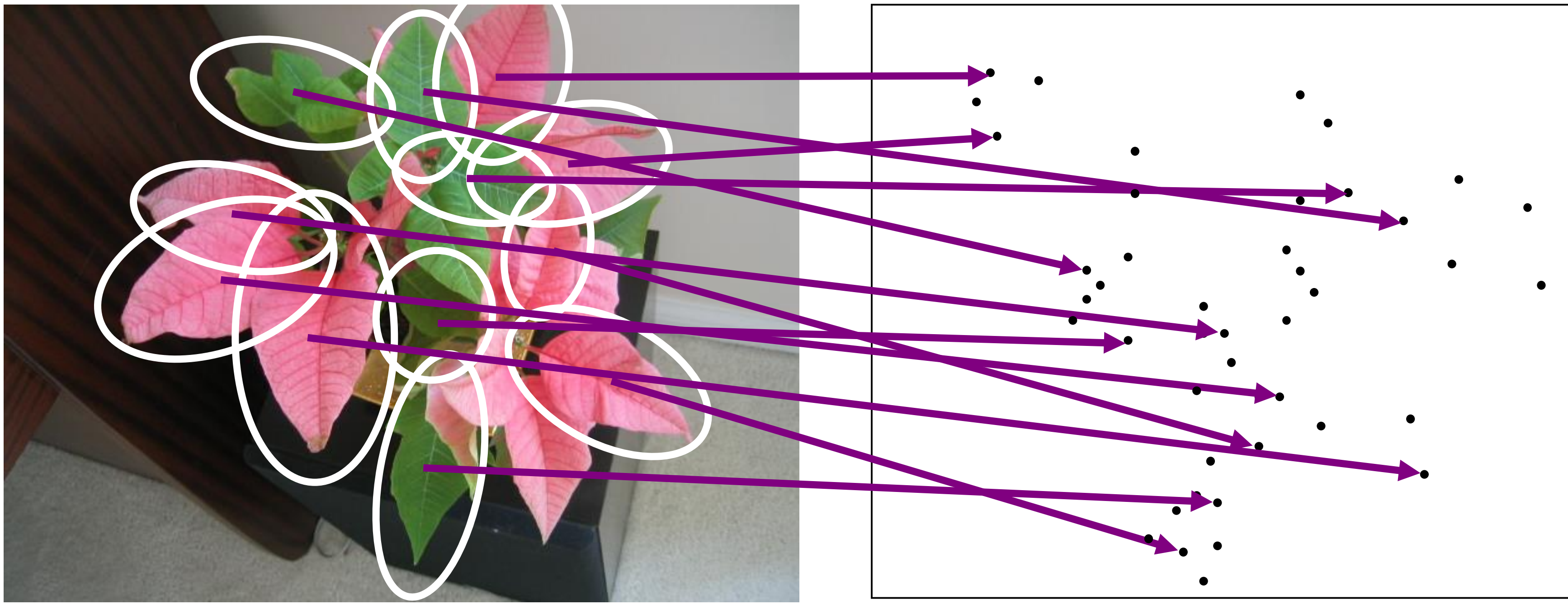
Visual Words: Main Idea

- Extract some local features from a number of images …



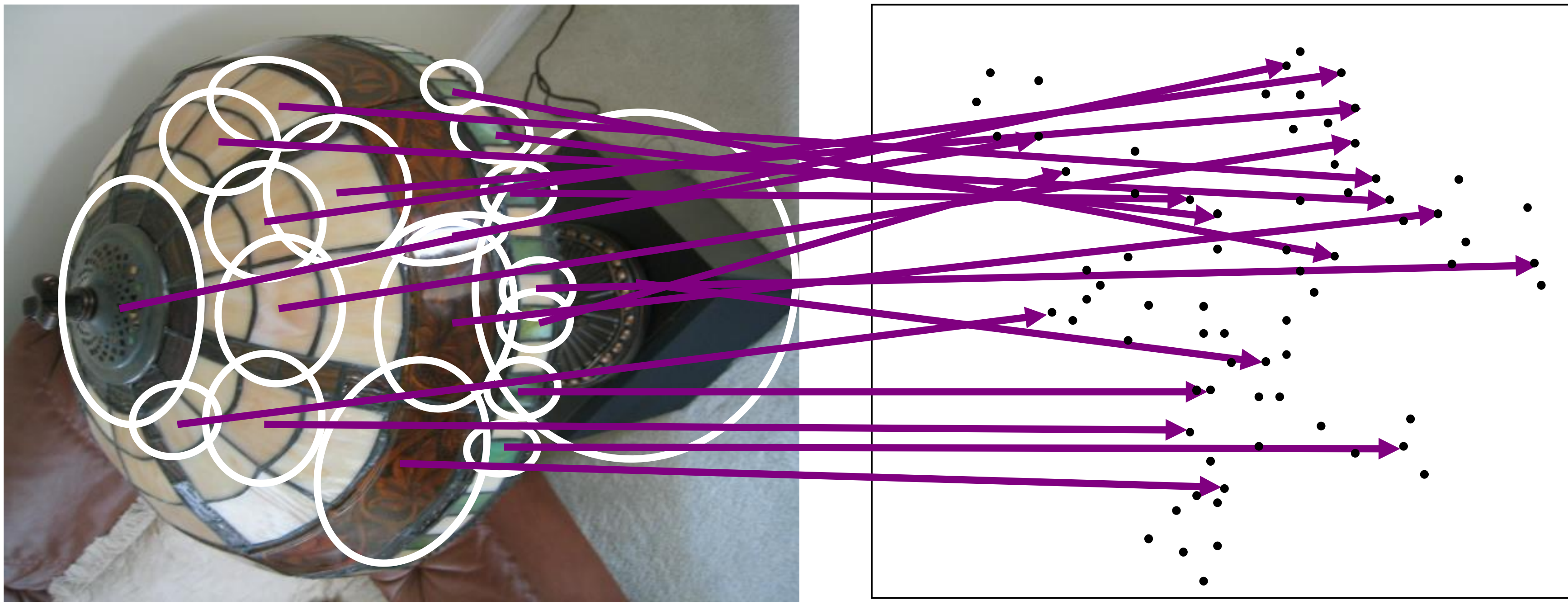
Slide credit: David Nister

Visual Words: Main Idea



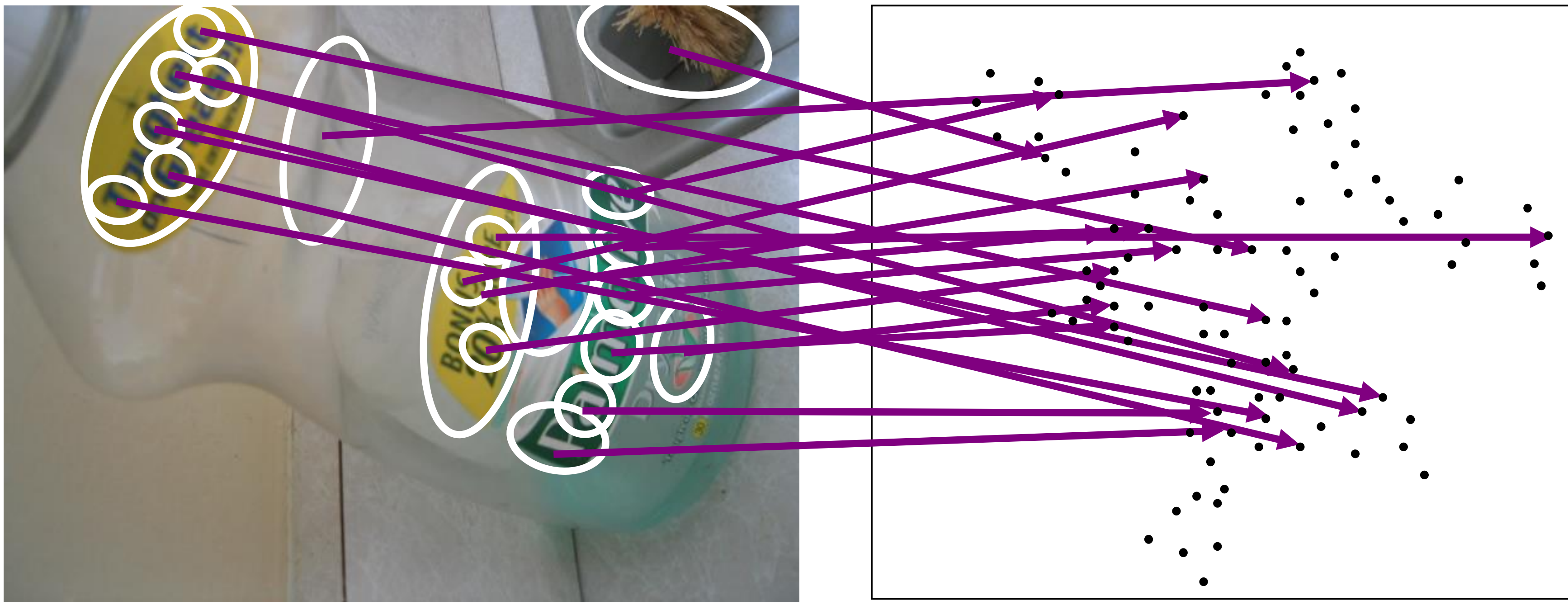
Slide credit: David Nister

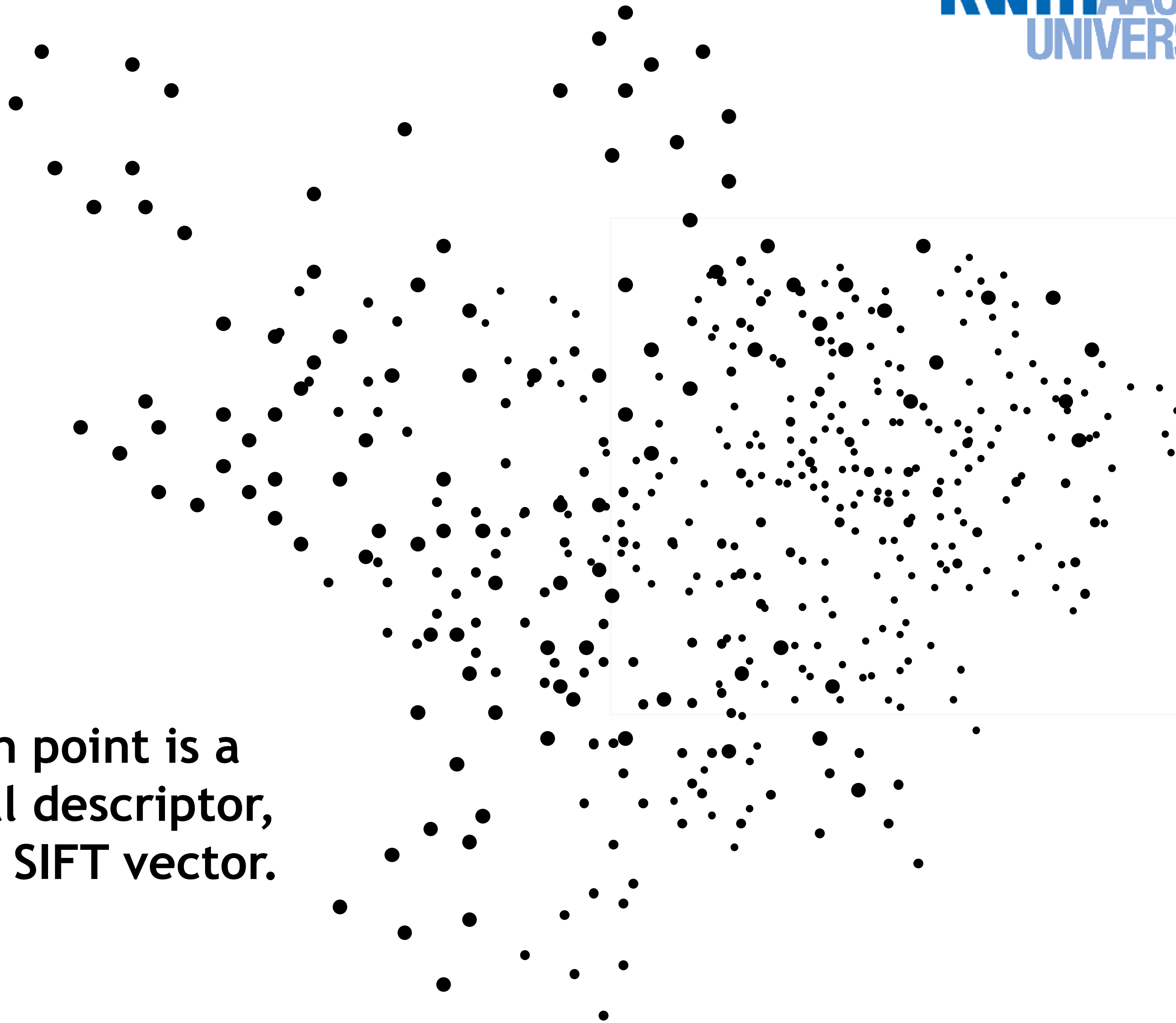
Visual Words: Main Idea



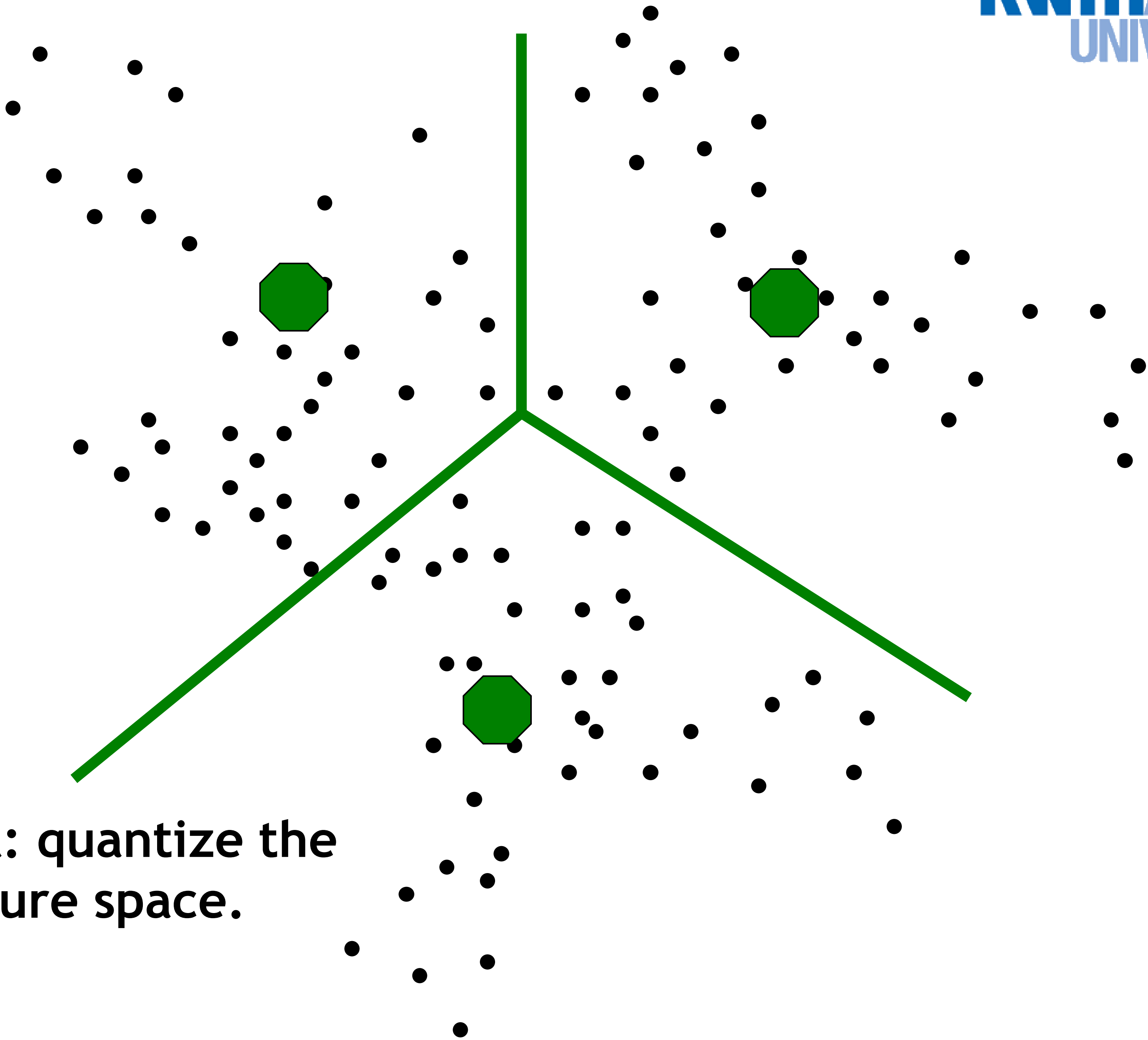
RWTH AACHEN UNIVERSITY

Visual Words: Main Idea





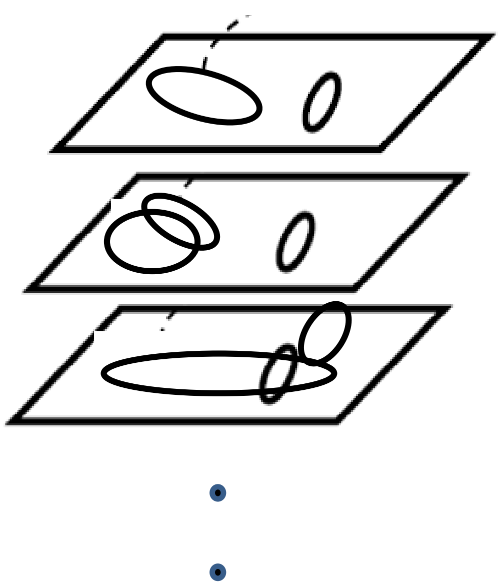
Each point is a local descriptor, e.g. SIFT vector.

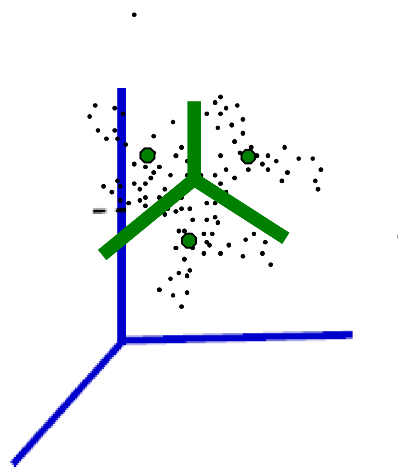


Idea: quantize the feature space.

Indexing with Visual Words

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



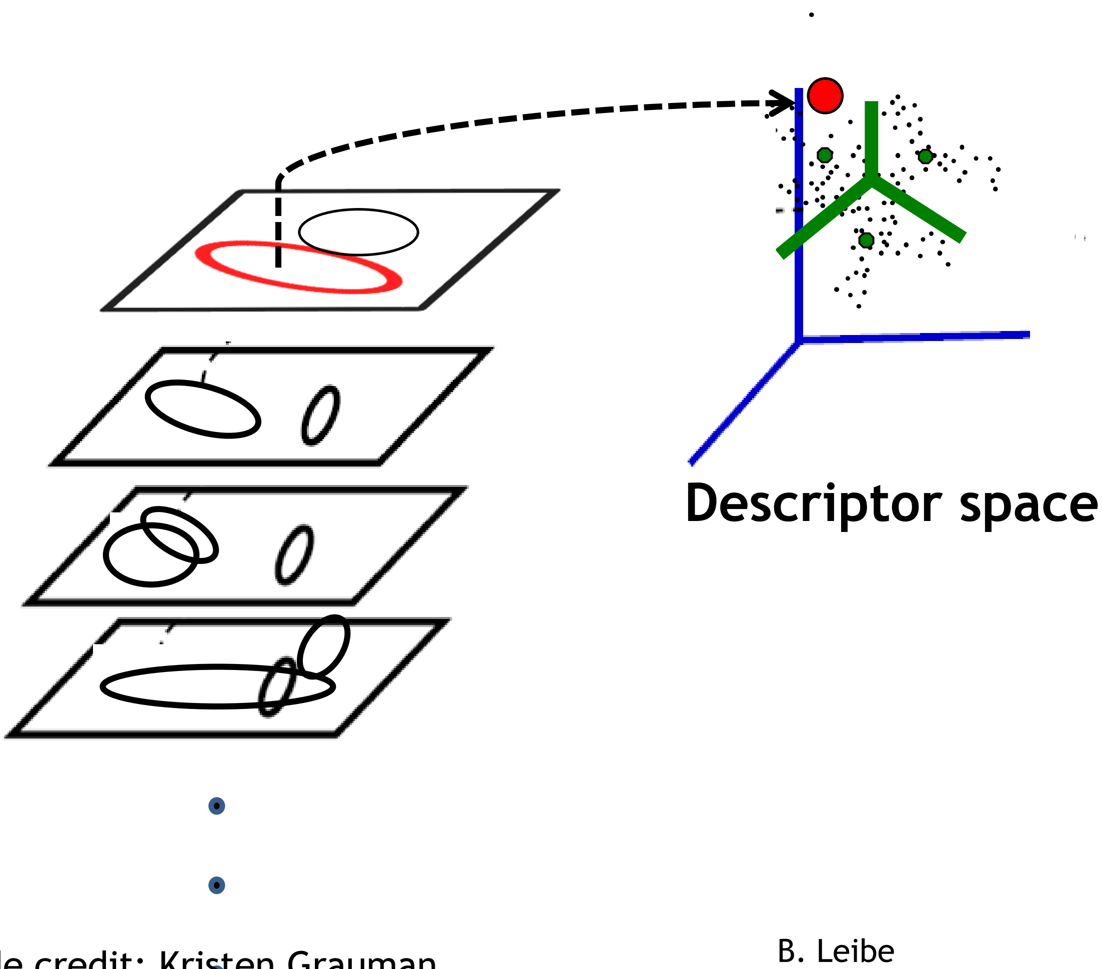


Descriptor space

- Quantize via clustering, let cluster centers be the prototype "words"

Indexing with Visual Words

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

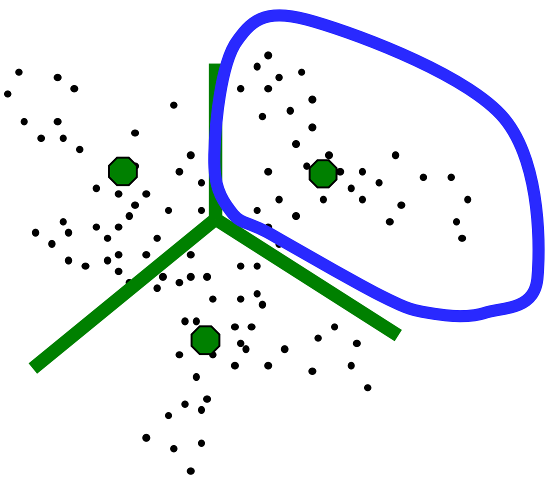


- Determine which word to assign to each new image region by finding the closest cluster center.

Slide credit: Kristen Grauman

Visual Words

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



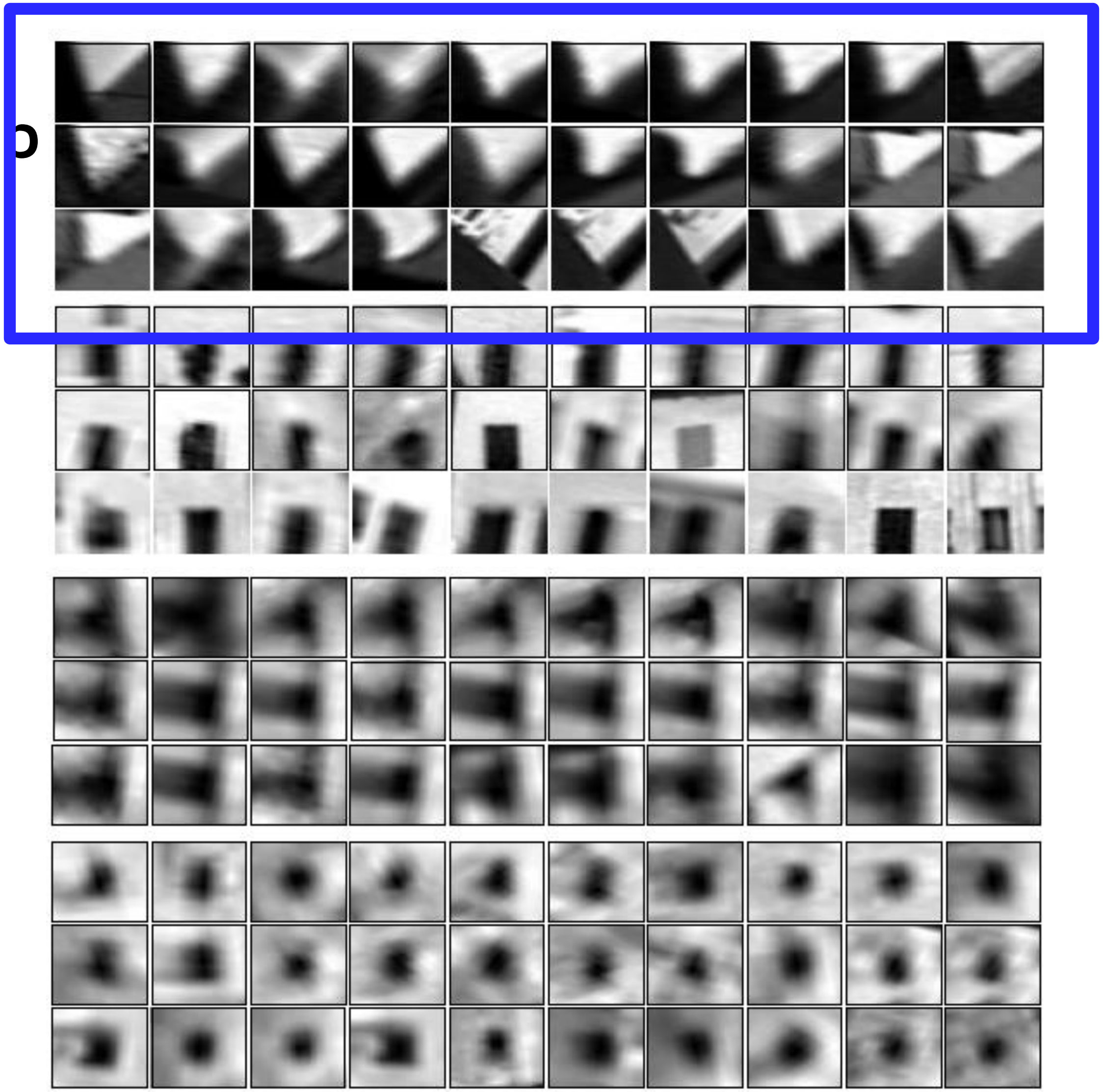
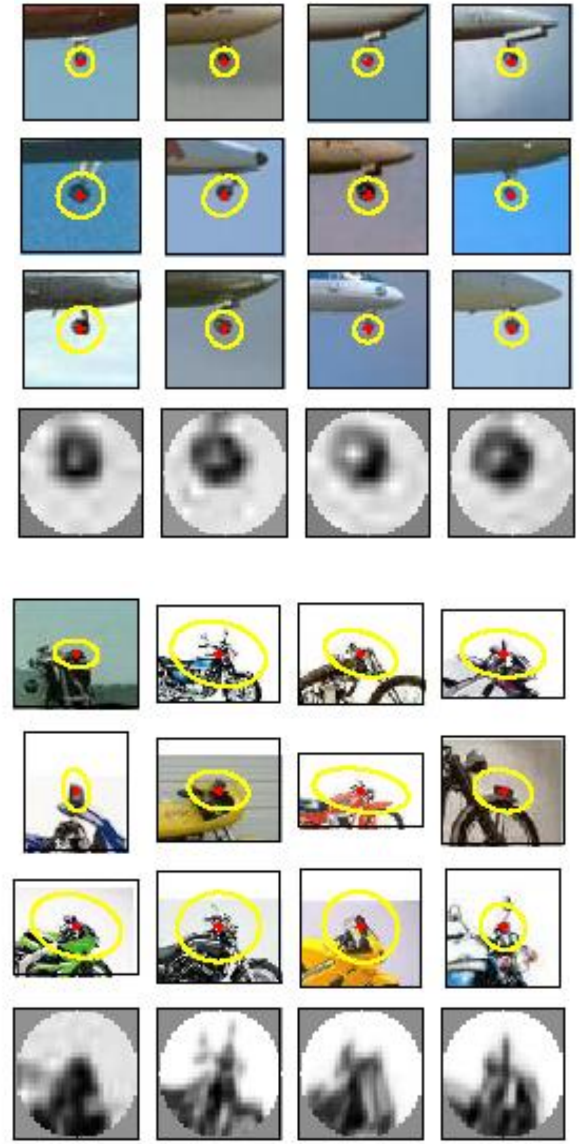


Figure from Sivic & Zisserman, ICCV 2003

Visual Words

- Often used for describing scenes and objects for the sake of indexing or classification.



Sivic & Zisserman 2003; Csurka, Bray, Dance, & Fan 2004; many others.

Inverted File for Images of Visual Words

frame #5

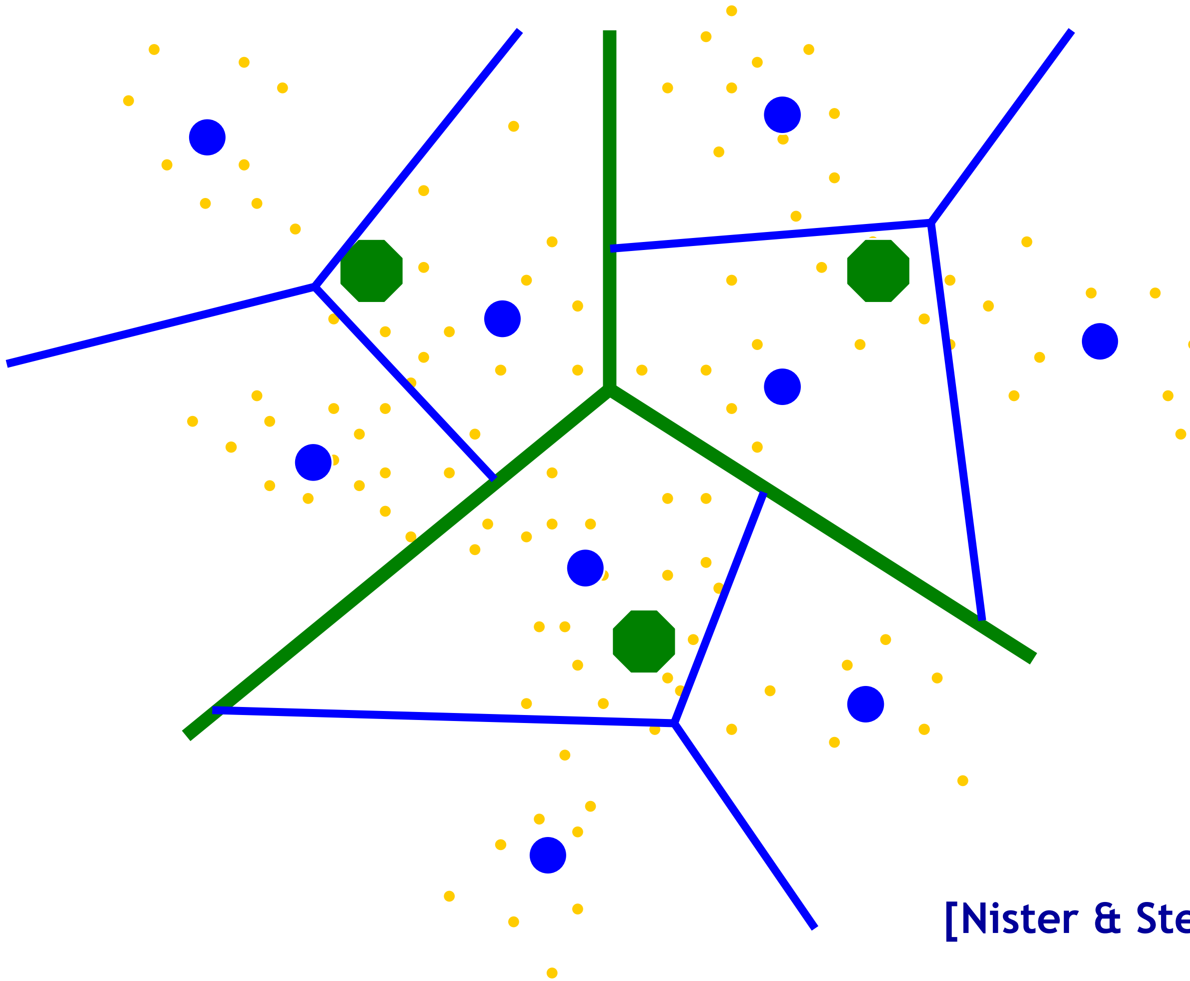
frame #10

Word number		List of image numbers
1	→	5,10, ...
2	→	10,...
...		...

When will this give us a significant gain in efficiency?

Example: Recognition with Vocabulary Tree

- **Tree construction:**

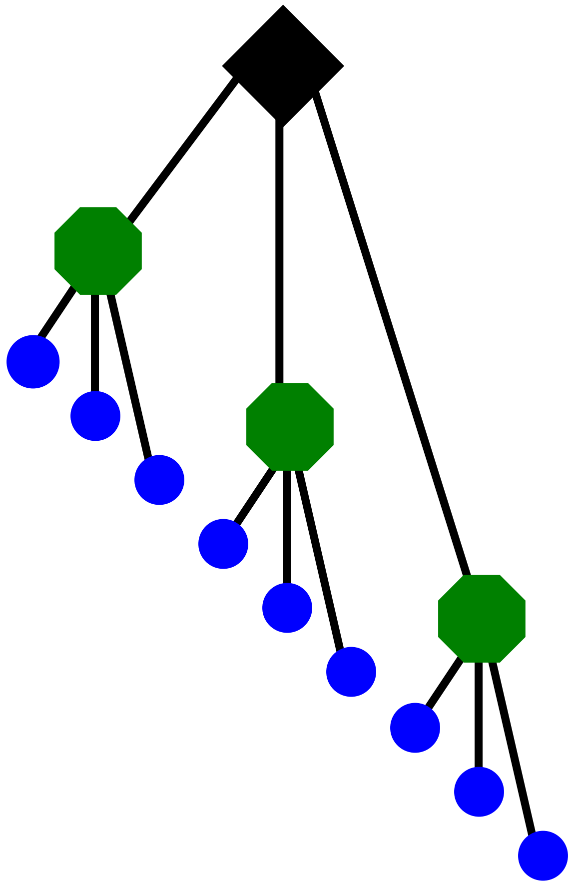


[Nister & Stewenius, CVPR'06]

Slide credit: David Nister

Vocabulary Tree

- **Training: Filling the tree**

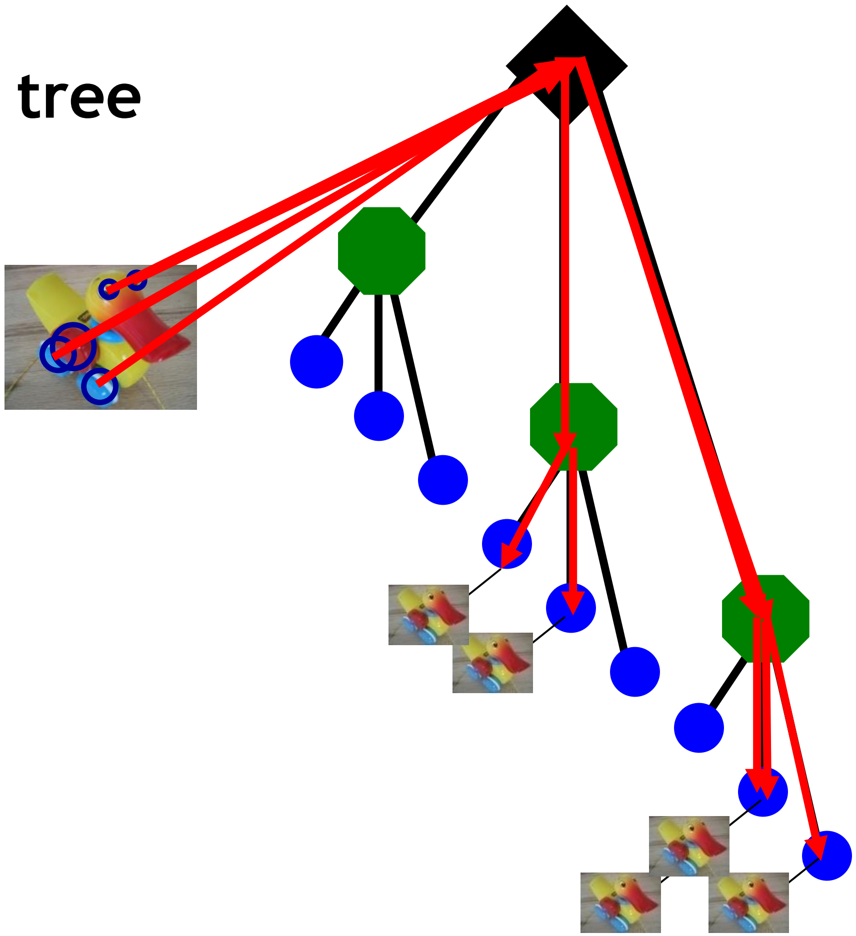


[Nister & Stewenius, CVPR'06]

Slide credit: David Nister

Vocabulary Tree

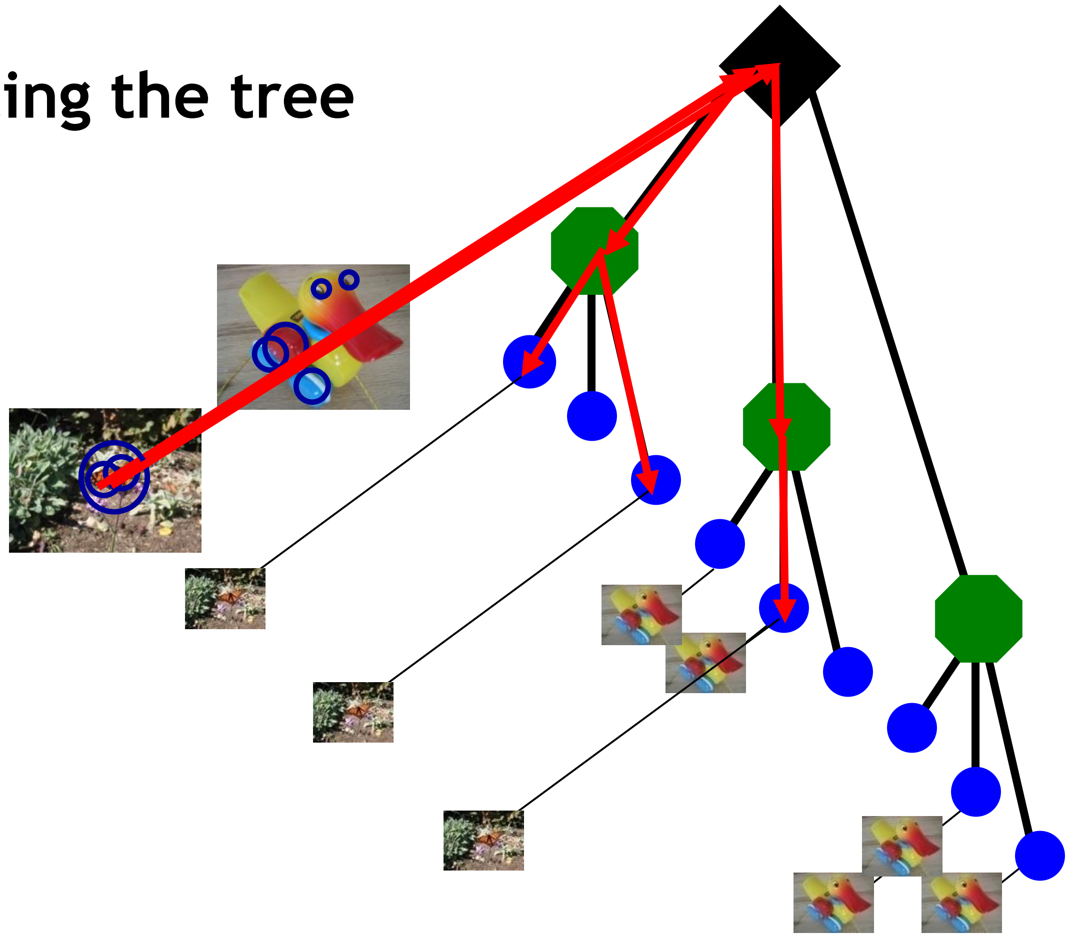
- **Training: Filling the tree**



[Nister & Stewenius, CVPR'06]

Vocabulary Tree

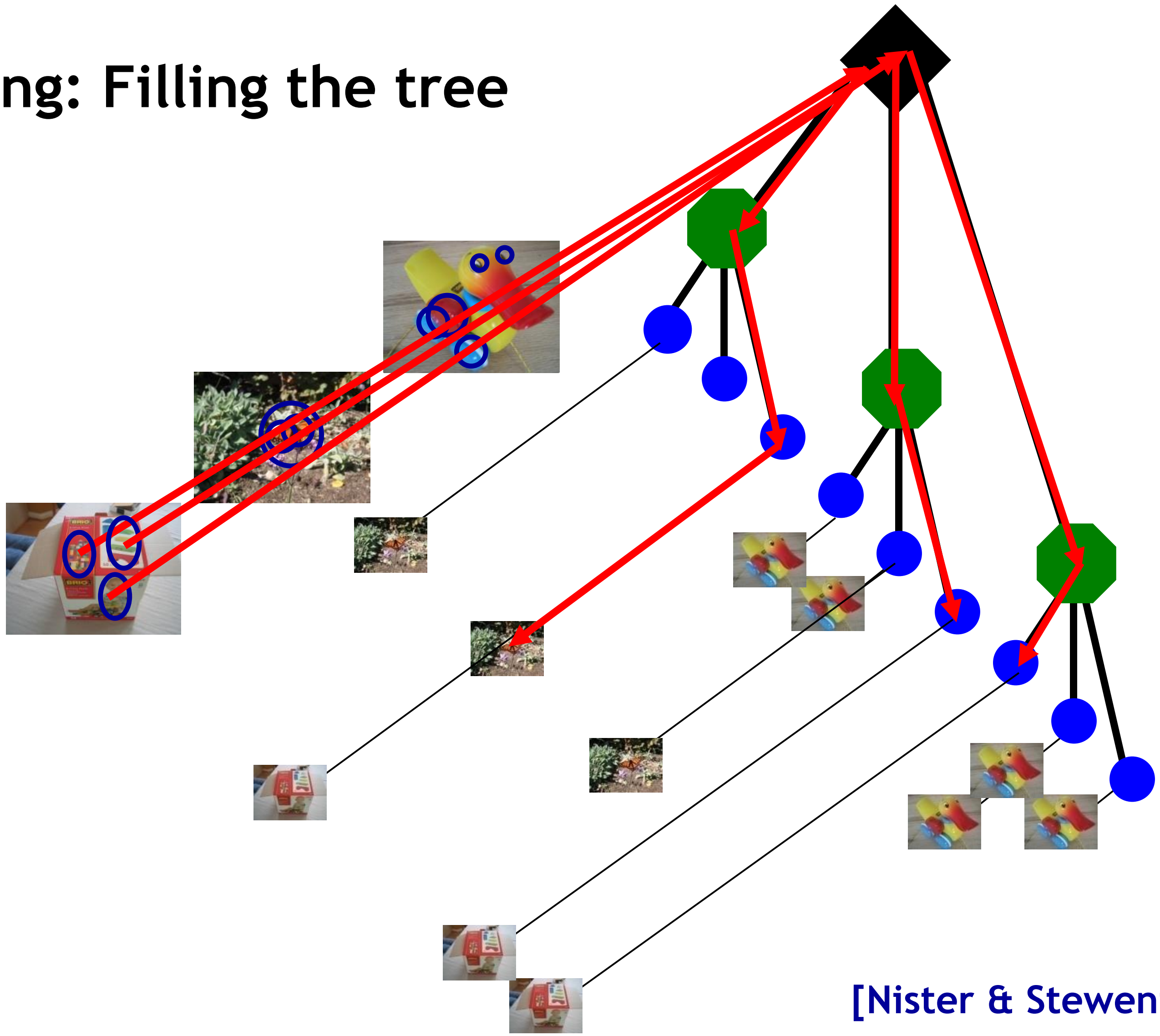
- **Training: Filling the tree**



[Nister & Stewenius, CVPR'06]

Vocabulary Tree

- **Training: Filling the tree**



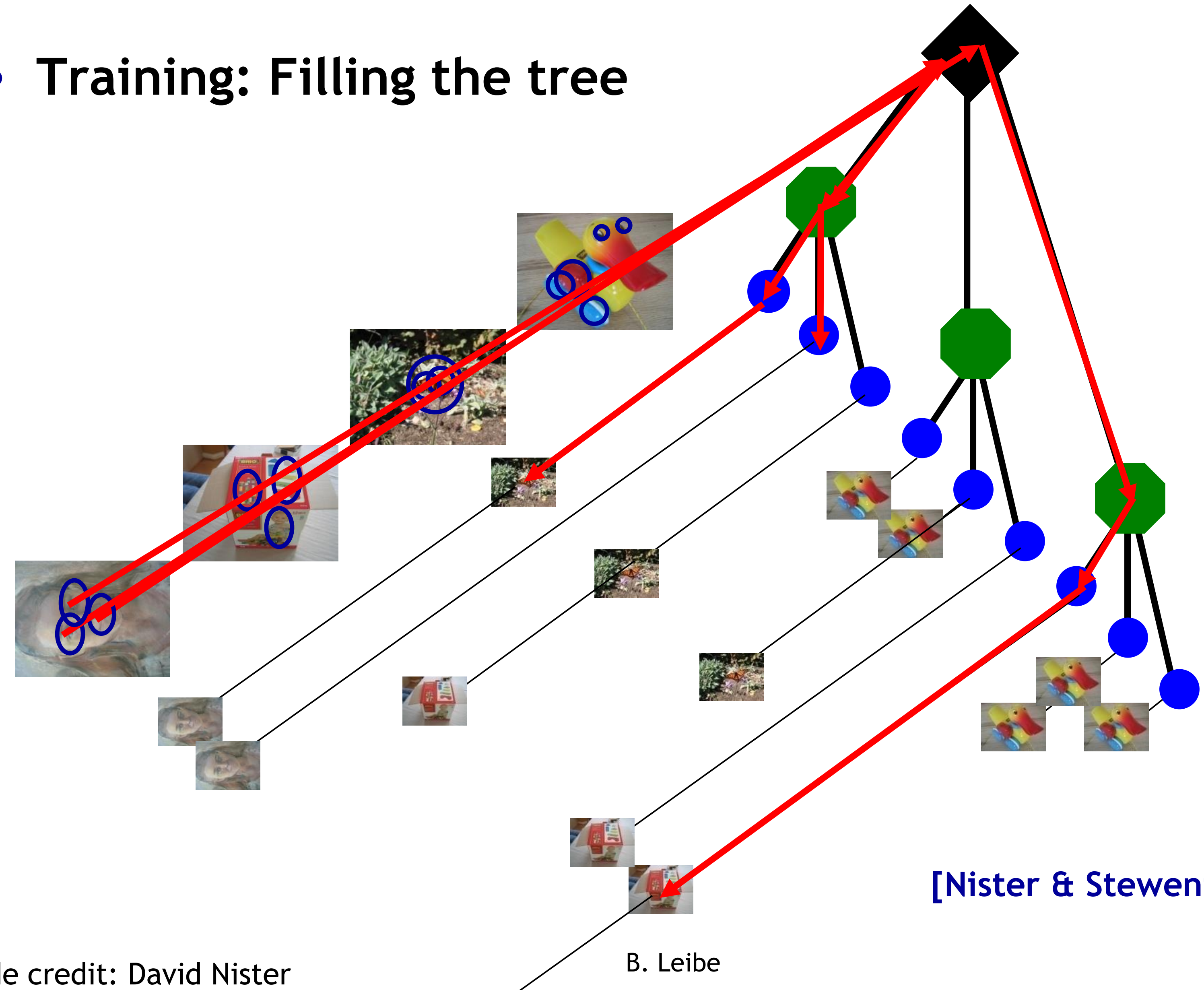
[Nister & Stewenius, CVPR'06]

Slide credit: David Nister

RWTH AACHEN UNIVERSITY

Vocabulary Tree

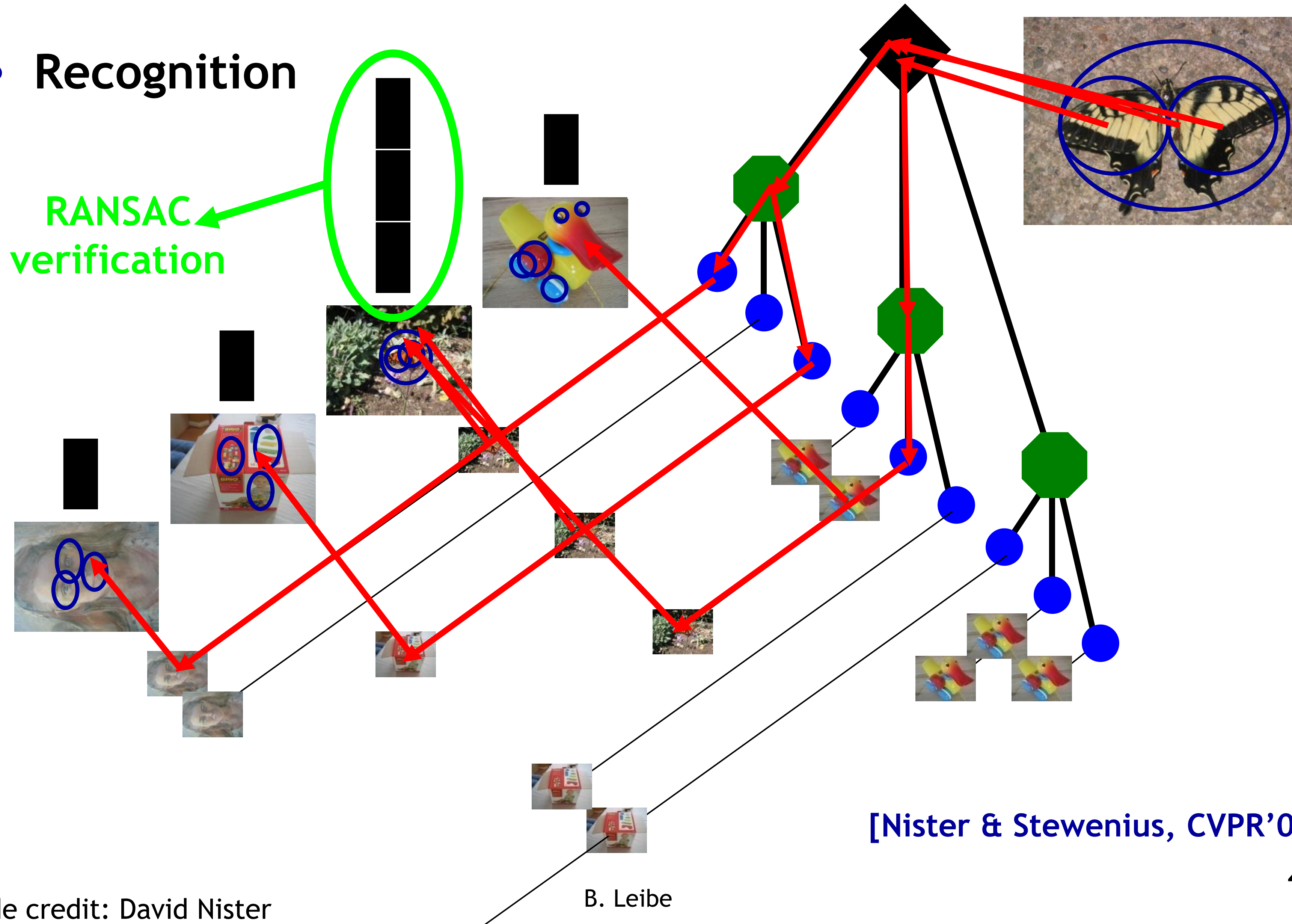
- **Training: Filling the tree**



[Nister & Stewenius, CVPR'06]

Vocabulary Tree

- **Recognition**



[Nister & Stewenius, CVPR'06]

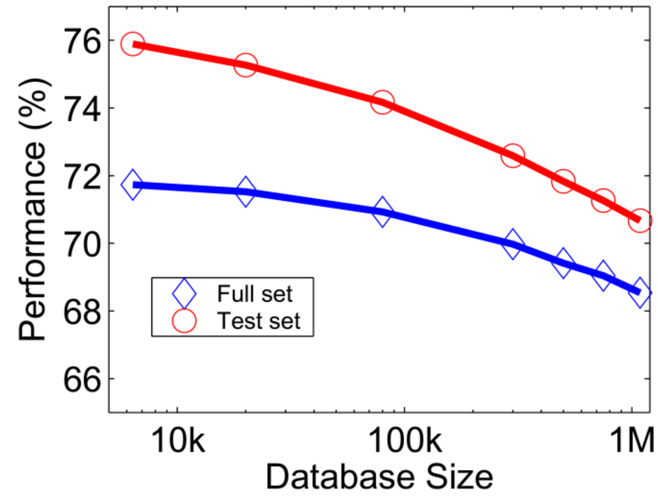
Quiz Questions

- What is the computational advantage of the hierarchical representation vs. a flat vocabulary?

- What dangers does such a representation carry?

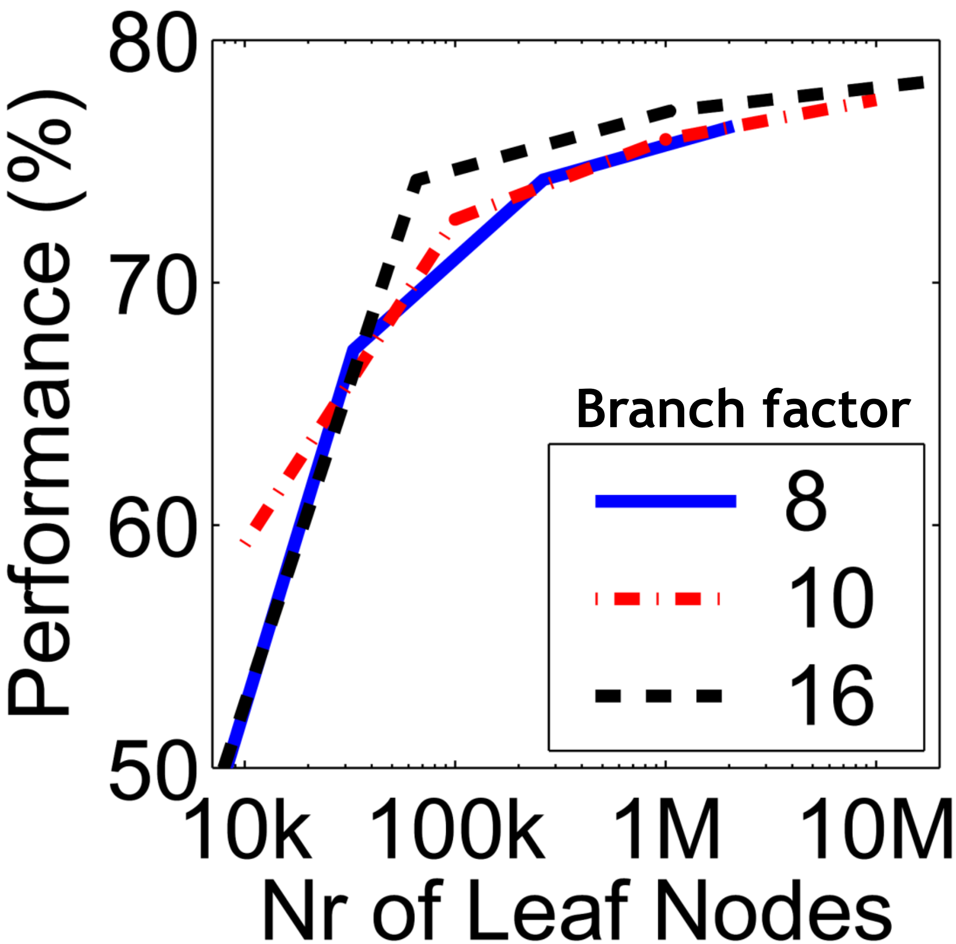
Vocabulary Tree: Performance

- Evaluated on large databases
  - Indexing with up to 1M images
- Online recognition for database of 50,000 CD covers
  - Retrieval in ~1s (in 2006)
- Experimental finding that large vocabularies can be beneficial for recognition



[Nister & Stewenius, CVPR'06]

Vocabulary Size



- Larger vocabularies can be advantageous…
- But what happens when the vocabulary gets too large?
  - Efficiency?
  - Robustness?

Figure from [Nister & Stewenius, CVPR'06]

tf-idf Weighting

- Term frequency - inverse document frequency
- Describe frame by frequency of each word within it, downweight words that appear often in the database
- (Standard weighting for text retrieval)

Number of occurrences of word i in document d

Number of words in document d

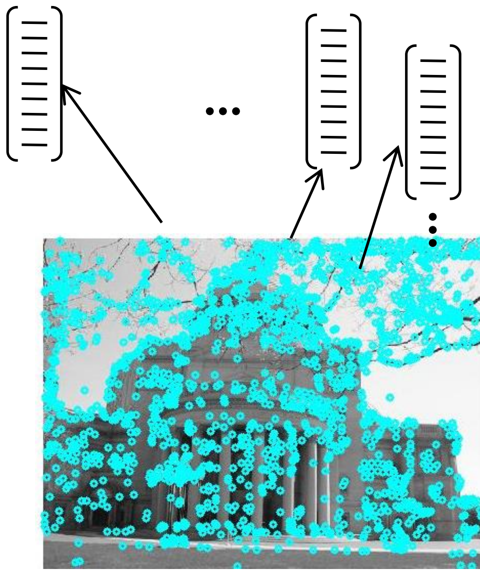
$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i}$$

Total number of documents in database

Number of occurrences of word i in whole database

Summary: Indexing features

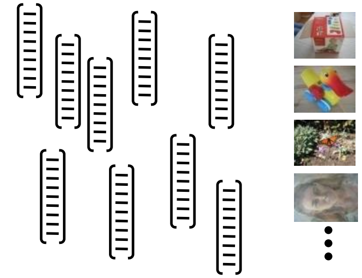
Detect or sample features



Describe features

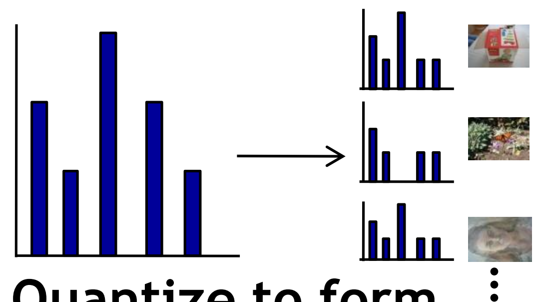
List of positions, scales, orientations

Associated list of d-dimensional descriptors



Index each one into pool of descriptors from previously seen images

or



Quantize to form "bag of words" vector for the image

Application for Content Based Img Retrieval

- **What if query of interest is a portion of a frame?**

Visually defined query

"Groundhog Day" [Rammis, 1993]

"Find this clock"

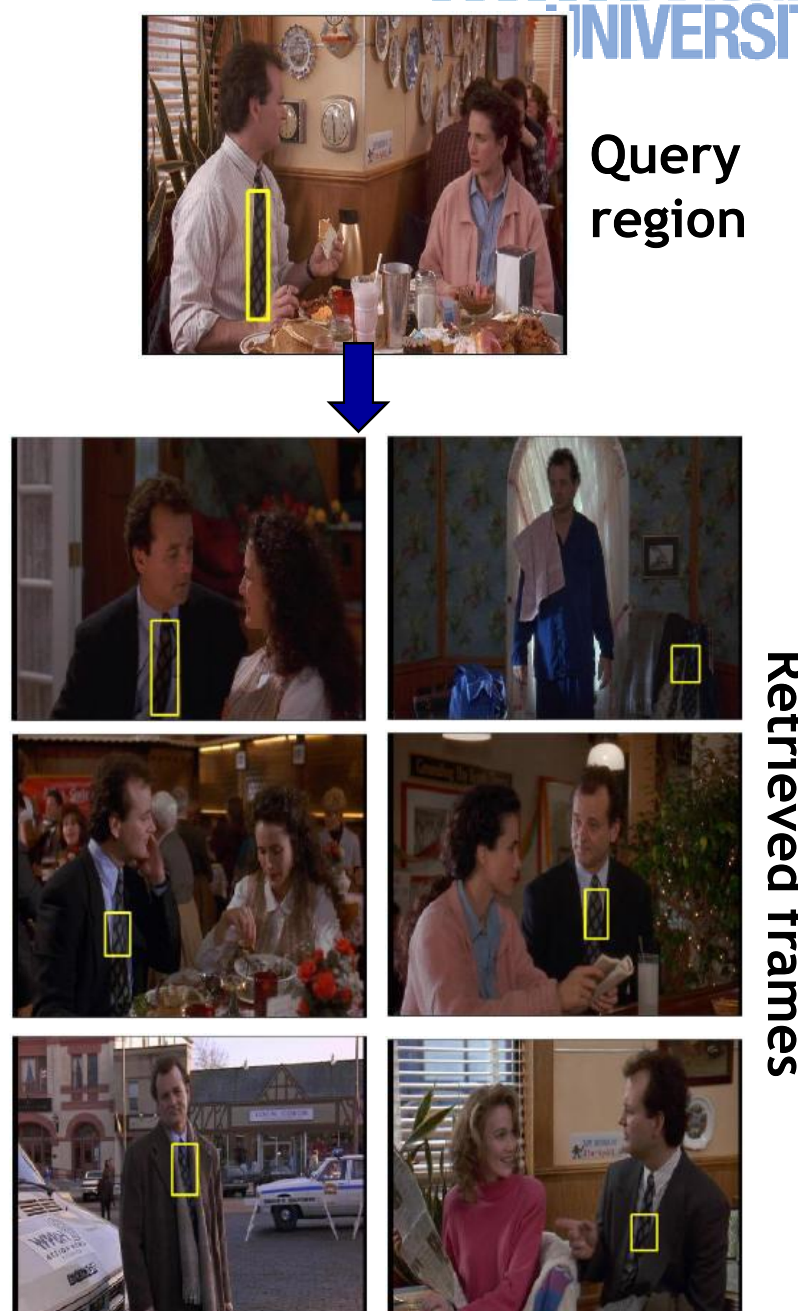
"Find this place"

[Sivic & Zisserman, ICCV'03]

Video Google System

1. Collect all words within query region
2. Inverted file index to find relevant frames
3. Compare word counts
4. Spatial verification

Sivic & Zisserman, ICCV 2003



Collecting Words Within a Query Region

- Example: Friends

Query region:
pull out only the SIFT descriptors whose positions are within the polygon

Slide credit: Kristen Grauman

Example Results

Query

raw nn 1sim=0.56697

raw nn 2sim=0.56163

raw nn 5sim=0.54917

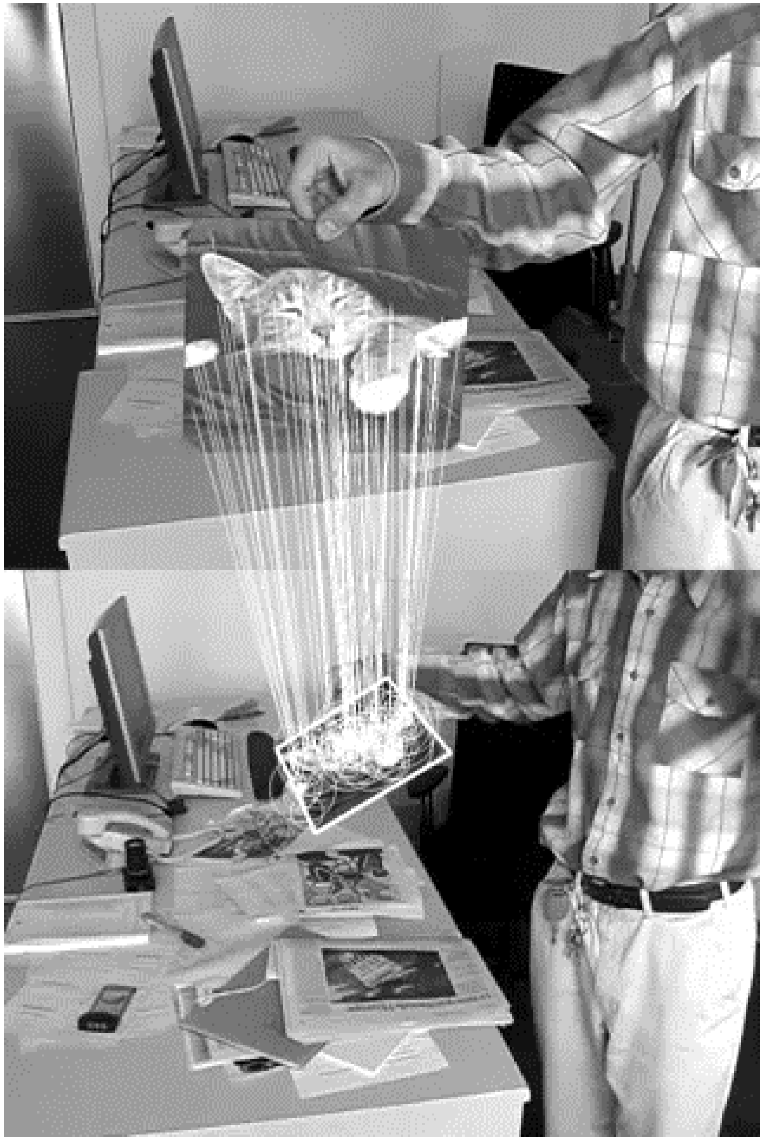
More Results

Query

Retrieved shots

Applications: Aachen Tourist Guide

Applications: Fast Image Registration



Applications: Mobile Augmented Reality

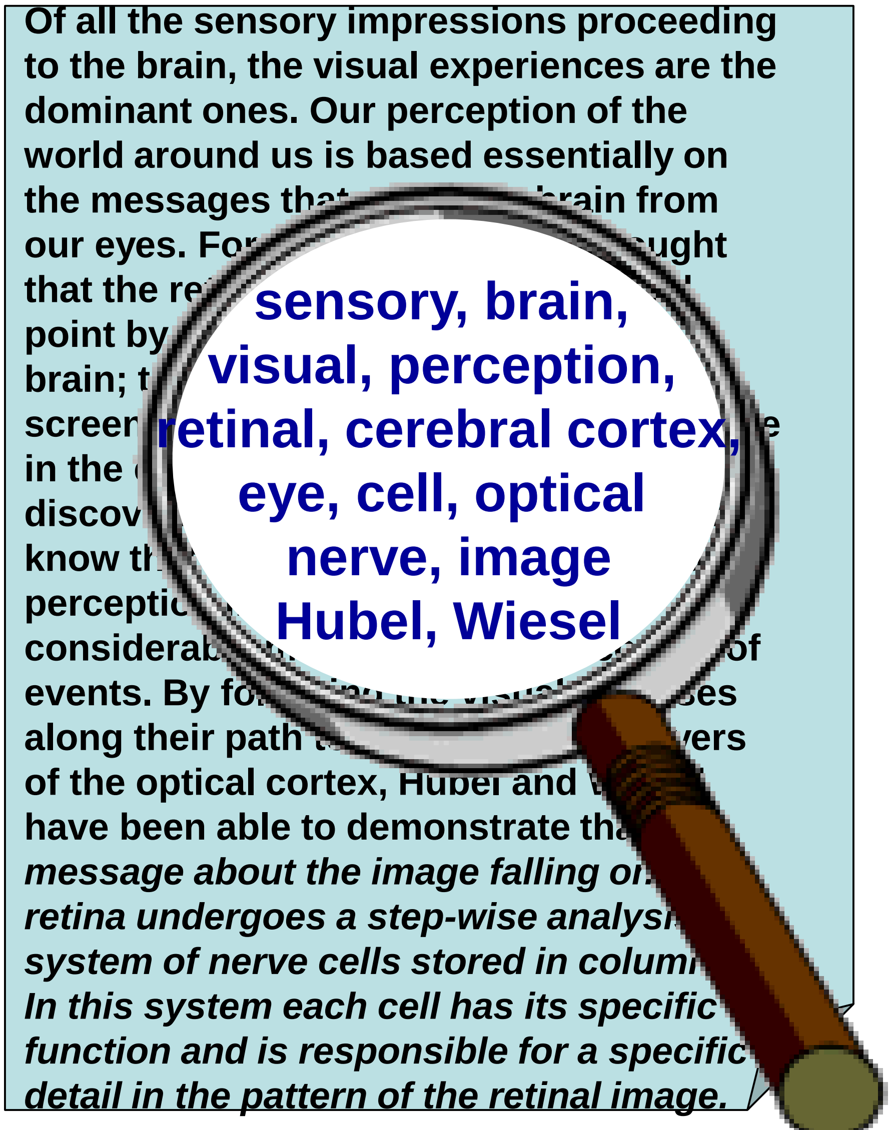
D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, D. Schmalstieg, Pose Tracking from Natural Features on Mobile Phones. In *ISMAR 2008*.

Topics of This Lecture

- Indexing with Local Features
  - Inverted file index
  - Visual Words
  - Visual Vocabulary construction
  - tf-idf weighting
- Bag-of-Words Model
  - Use for image classification

RWTH AACHEN UNIVERSITY

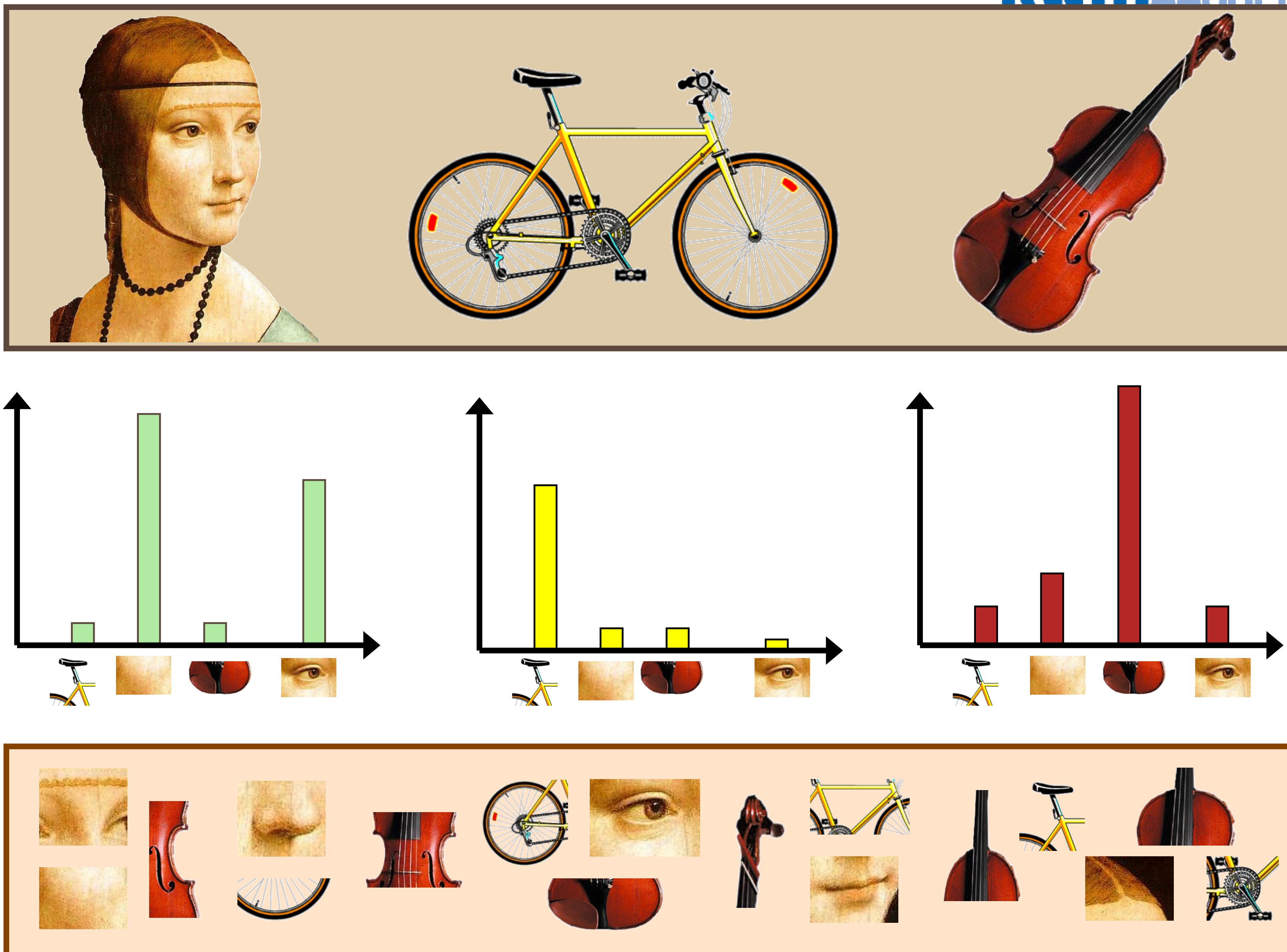
Analogy to Documents



Slide credit: Li Fei-Fei

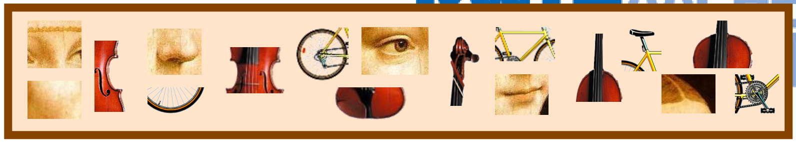
Object → Bag of ‘words’

Source: ICCV 2005 short course, Li Fei-Fei



Source: ICCV 2005 short course, Li Fei-Fei

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.

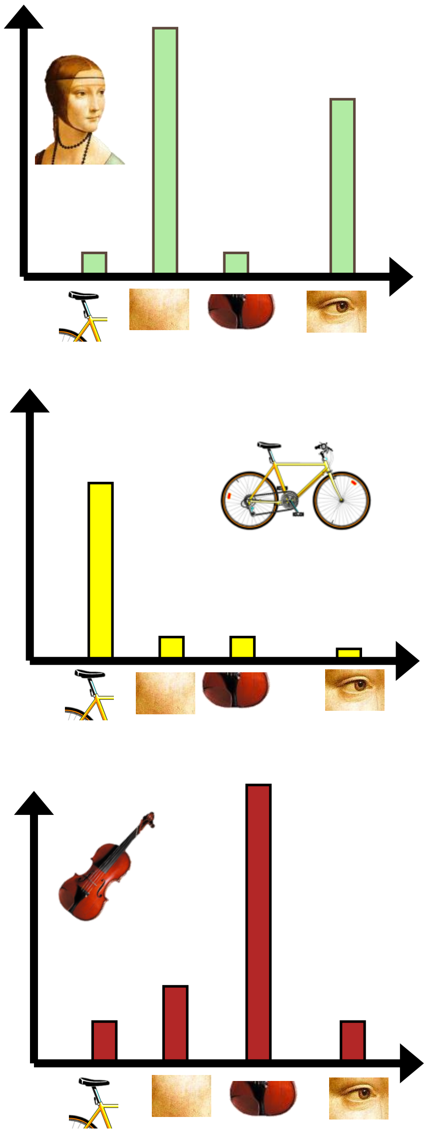
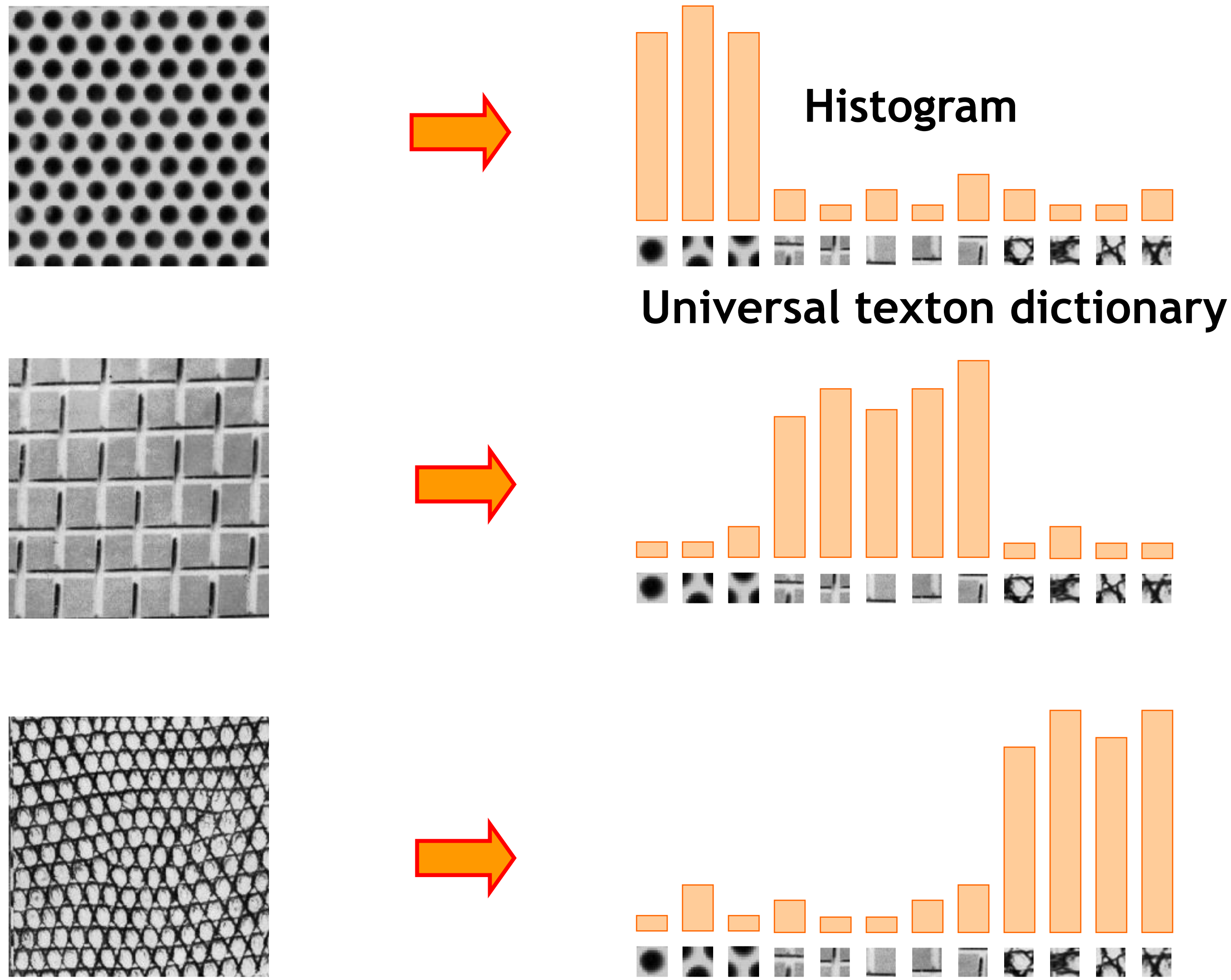


Image credit: Li Fei-Fei

Similarly, Bags-of-Textons for Texture Repr.



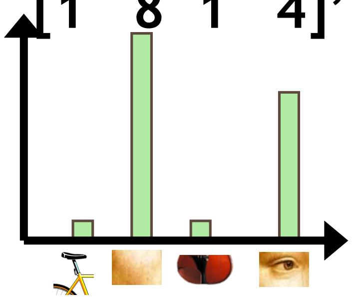
Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

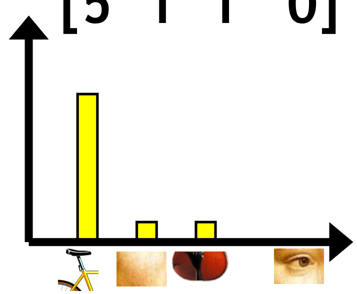
Slide credit: Svetlana Lazebnik

Comparing Bags of Words

- We build up histograms of word activations, so any histogram comparison measure can be used here.
- E.g. we can rank frames by normalized scalar product between their (possibly weighted) occurrence counts
  - *Nearest neighbor* search for similar images.

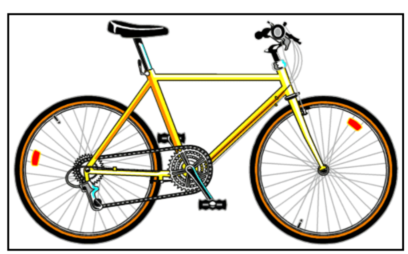
[1 8 1 4]' • [5 1 1 0]





$$sim(d_j, q) = \frac{\vec{d_j} \bullet \vec{q}}{|\vec{d_j}| \times |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{j=1}^{t} w_{i,q}^2}}$$

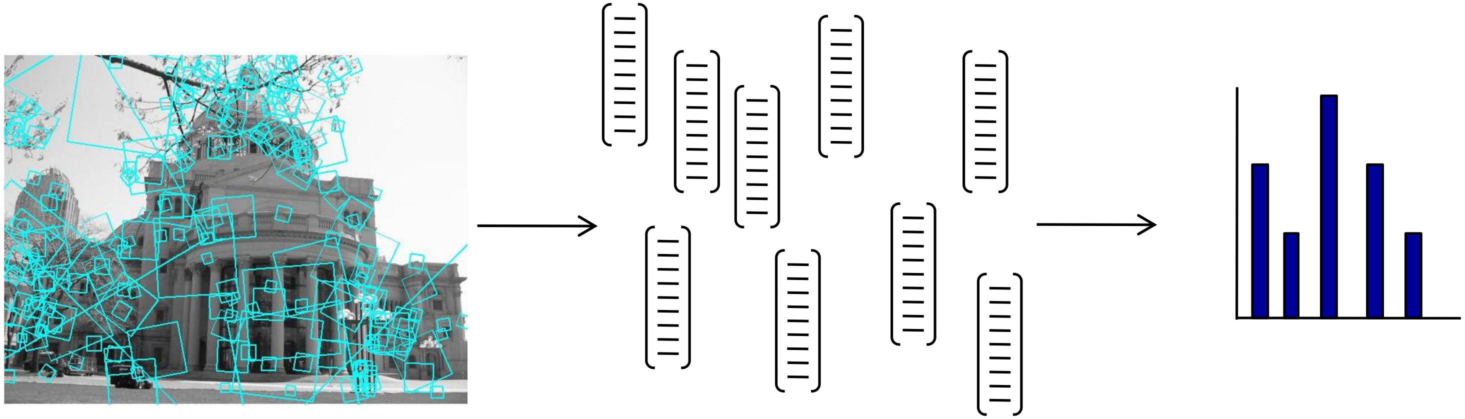
$\vec{d}_j$



$\vec{q}$

Learning/Recognition with BoW Histograms

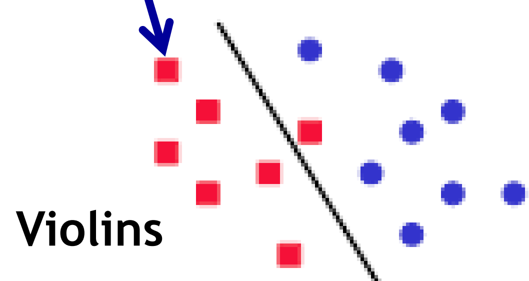
- Bag of words representation makes it possible to describe the unordered point set with a single vector (of fixed dimension across image examples)



- Provides easy way to use distribution of feature types with various learning algorithms requiring vector input.

Bags-of-Words for Classification

- Compute the word activation histogram for each image.
- Let each such BoW histogram be a feature vector.
- Use images from each class to train a classifier (e.g., an SVM).



BoW for Object Categorization

{face, flowers, building}

- Works pretty well for image-level classification

Csurka et al. (2004), Willamowski et al. (2005), Grauman & Darrell (2005), Sivic et al. (2003, 2005)

BoW for Object Categorization

Caltech6 dataset

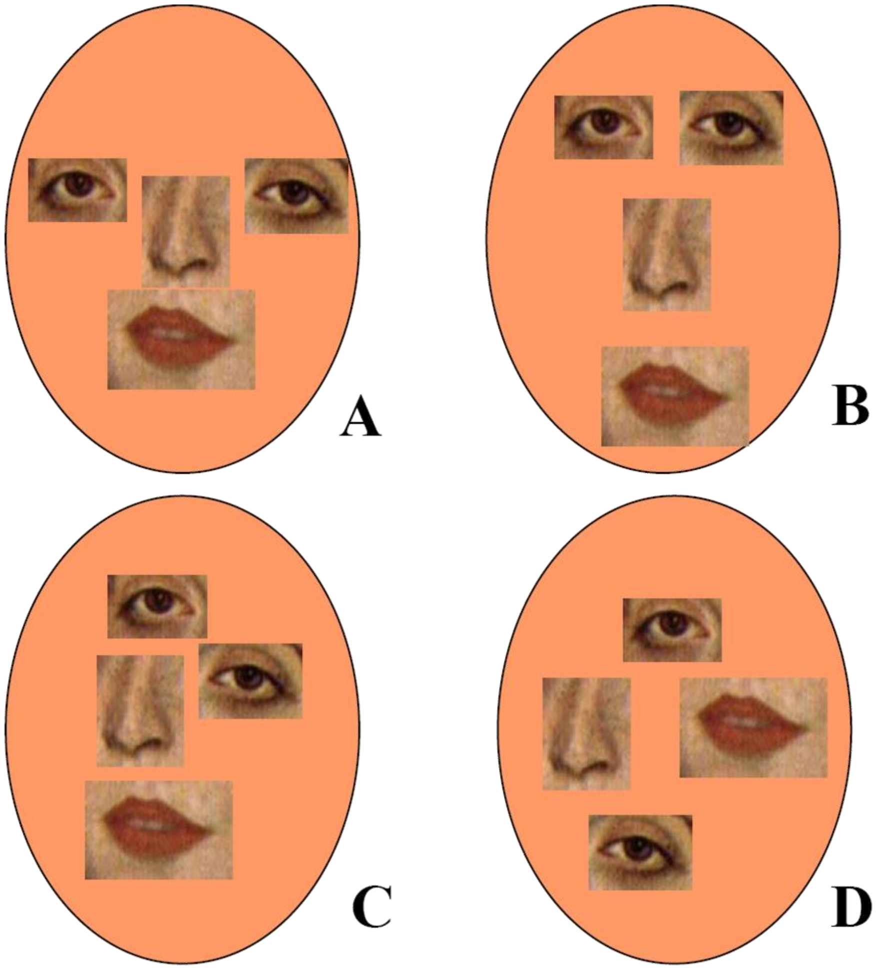
     

class	bag of features Zhang et al. (2005)	bag of features Willamowski et al. (2004)	Parts-and-shape model Fergus et al. (2003)
airplanes	**98.8**	97.1	90.2
cars (rear)	98.3	**98.6**	90.3
cars (side)	**95.0**	87.3	88.5
faces	**100**	99.3	96.4
motorbikes	**98.5**	98.0	92.5
spotted cats	**97.0**	—	90.0

- Good performance for pure classification (object present/absent)
  - Better than more elaborate part-based models with spatial constraints…
  - What could be possible reasons why?

Limitations of BoW Representations

- The bag of words removes spatial layout.
- This is both a strength and a weakness.
- *Why a strength?*
- *Why a weakness?*



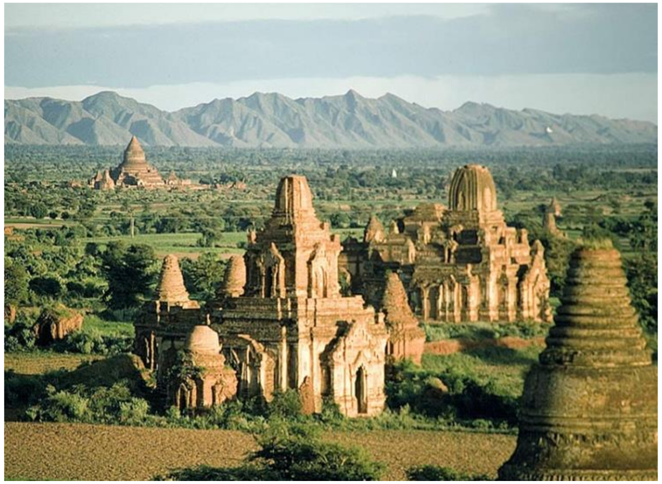
BoW Representation: Spatial Information

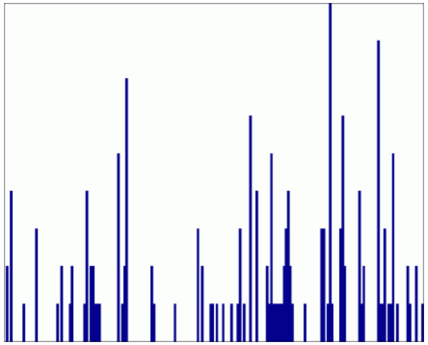
- A bag of words is an *orderless* representation: throwing out spatial relationships between features
- Middle ground:
  - Visual "phrases" : frequently co-occurring words
  - Semi-local features : describe configuration, neighborhood
  - Let position be part of each feature
  - Count bags of words only within sub-grids of an image
  - After matching, verify spatial consistency (e.g., look at neighbors - are they the same too?)

RWTH AACHEN UNIVERSITY

Spatial Pyramid Representation

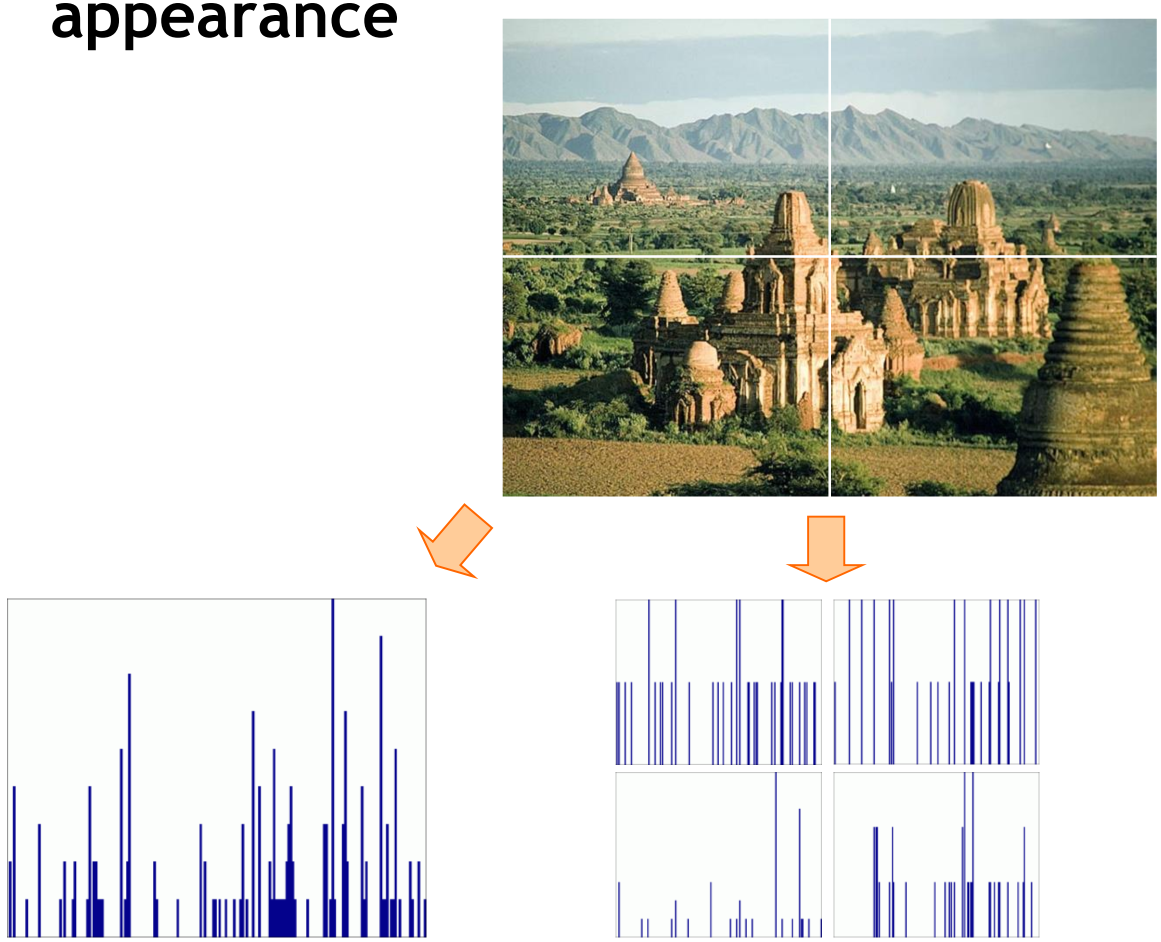
- Representation in-between orderless BoW and global appearance





Spatial Pyramid Representation

- **Representation in-between orderless BoW and global appearance**

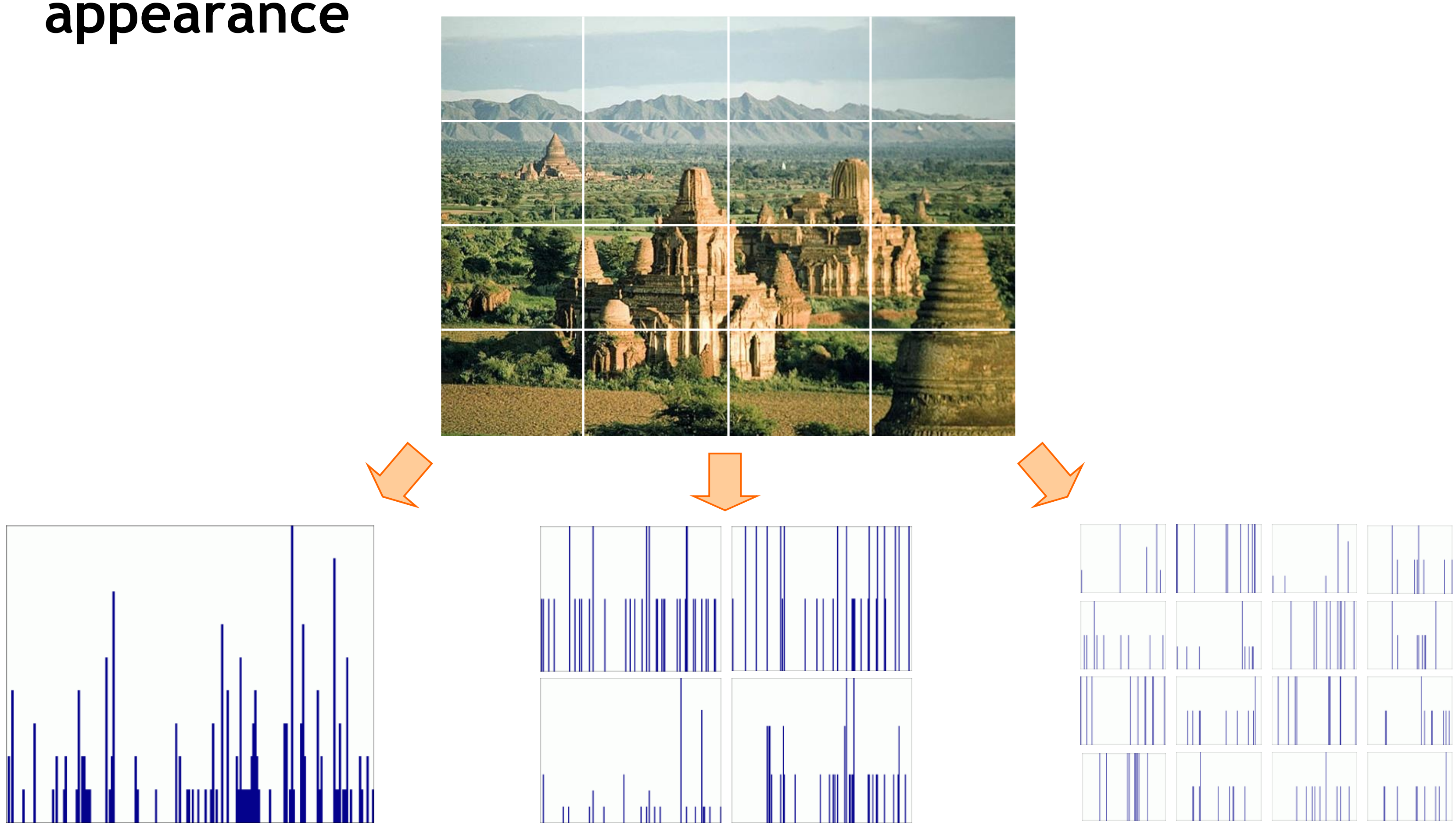


Slide credit: Svetlana Lazebnik

[Lazebnik, Schmid & Ponce, CVPR'06]

Spatial Pyramid Representation

- Representation in-between orderless BoW and global appearance



[Lazebnik, Schmid & Ponce, CVPR'06]

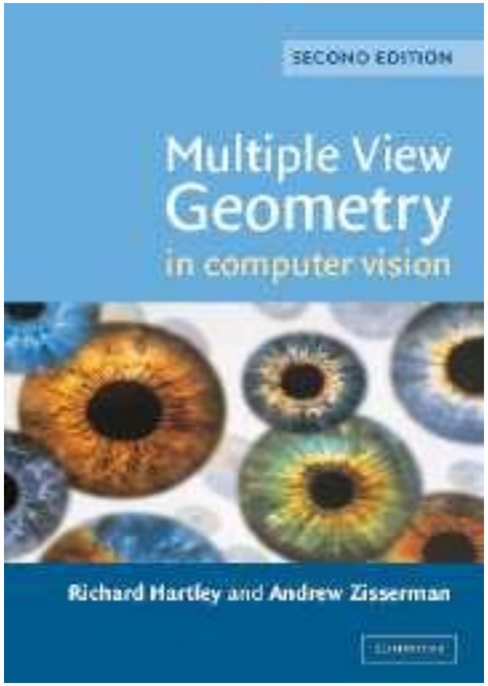
Summary: Bag-of-Words

- **Pros:**
  - Flexible to geometry / deformations / viewpoint
  - Compact summary of image content
  - Provides vector representation for sets
  - Empirically good recognition results in practice

- **Cons:**
  - Basic model ignores geometry - must verify afterwards, or encode via features.
  - Background and foreground mixed when bag covers whole image
  - Interest points or sampling: no guarantee to capture object-level parts.
  - Optimal vocabulary formation remains unclear.

References and Further Reading

- More details on RANSAC can be found in Chapter 4.7 of
  - R. Hartley, A. Zisserman
    Multiple View Geometry in Computer Vision
    2nd Ed., Cambridge Univ. Press, 2004



- Details about the Hough transform for object recognition can be found in
  - D. Lowe, Distinctive image features from scale-invariant keypoints, *IJCV* 60(2), pp. 91-110, 2004

- Details about the Video Google system can be found in
  - *J. Sivic, A. Zisserman,*
    Video Google: A Text Retrieval Approach to Object Matching in Videos, ICCV'03, 2003.