From Shallow to Deep representation for multimedia data classification

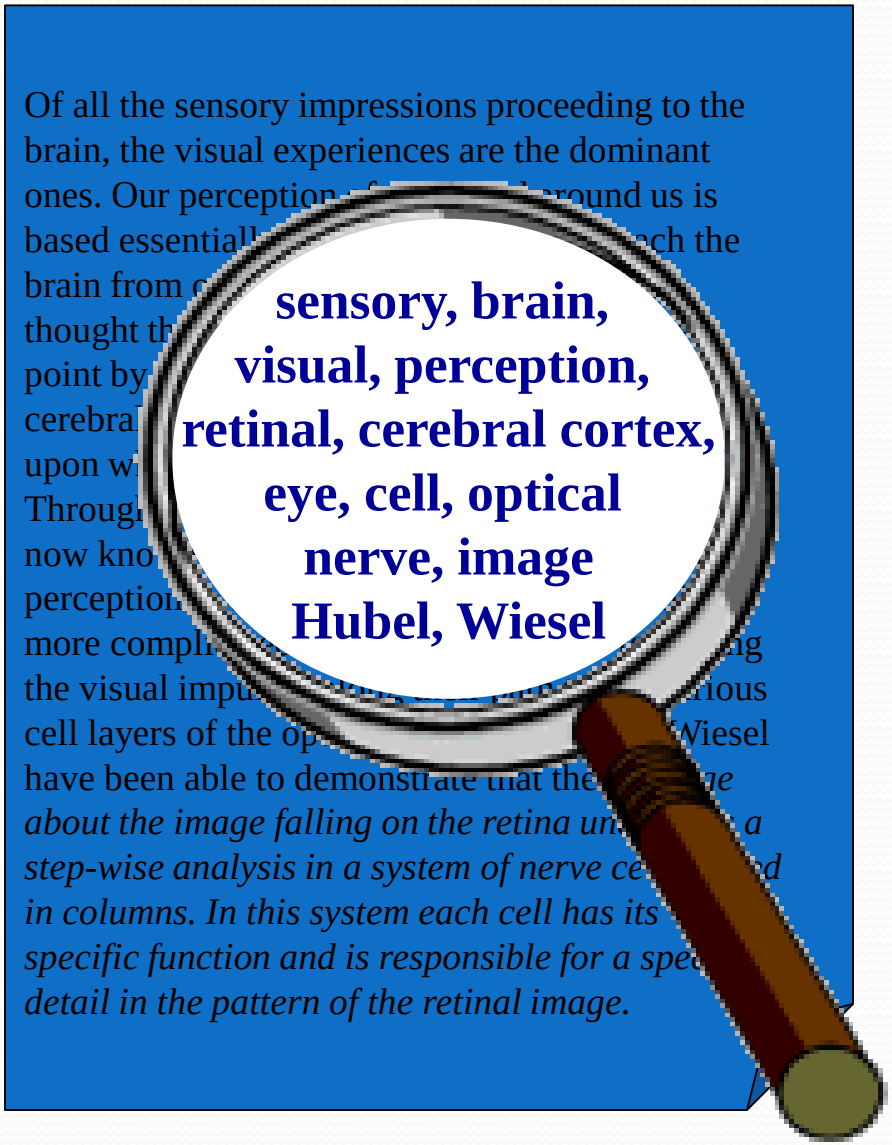
Lecture # 3

BoW and extensions: coding, pooling & spatial pyramid

2017-2018

Frederic Precioso

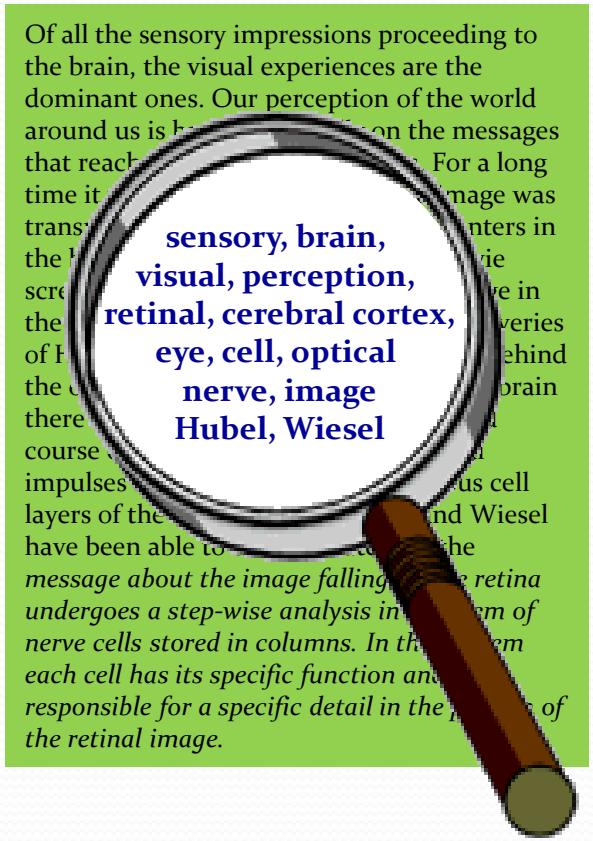
Analogy to documents



Bag of Word: introduction

- Requires to find the same object with deformations
- Winning combinaison: sparse PoI detector + discriminant and invariant descriptors like SIFT
- Local => intrinsically robust to occlusions and noise (i.e. the background)
- Challenge: fast search of nearest neighbors in the very large scale datasets
- Geometrical checking in order to constraint the matches

Bag of Word model (BoW) from text retrieval



We compare two documents by comparing their histograms of occurrence of words

Slide credit L. Fei-Fei

Documents (Google)

Create search engines understanding natural language
-> evolution of *Google* beyond keywords
-> « text-mining » or « web-mining »

Collection: ensemble of documents

Vector Space Model (VSM): *N*-dimension space with *N*, the number of useful terms from the language (“the”, “of”...are useless and are thus called **stop-words**). The space is so split into *N* clusters…

Documents (cont'd)

- Step 1: For each of the *N* terms of the language, create an **Inverse Index** which store, for each term of the language the documents containing this term
  - Computation of **Document Frequency** = nb doc using this term / total document = **df** (greater the df, the less the term has importance from an informative point of view)
  - soit **idf = log (1/df)**
- Step 2: for each document,
  - for each term, computation of the **Term Frequency** (greater tf is in a document, more this term must be important to the subject of the document)
  - Computation of a feature vector VSM :
    - For *i* from *0* to *N*, *VSM[i]* = **tf**(term *i* in this document)*__idf__(term *i*)
    - We normalize this vector to get ||*VSM*(document)||=*1*

Documents (cont'd)

- If you want to be more statistician = measures of proximity between 2 vectors-distributions
- For the documents (a feature = a word)…

Table 2: Tableau de contingence

		Mots					
		m_1	…	m_j	…	m_L	somme
	D_1	$f_{1,1}$	…	f_{1j}	…	f_{1L}	…
	…	…	…	…	…	…	…
Textes	D_i	f_{i1}	…	f_{ij}	…	f_{iL}	$fi. = \sum_{j=1}^{L} f_{ij}$
	…	…	…	…	…	…	…
	D_D	f_{D1}	…	f_{Dj}	…	f_{DL}	…
	somme	…	…	$f.j = \sum_{i=1}^{D} f_{ij}$	…	…	f

Documents (cont'd)

- In the context of text mining, the **distance du χ2** (*chi2* or *khi2*) is the Euclidean distance between two vectors-documents normalized by their lenght (number of words),

$$d^2_{eNid}\left(VSM_1, VSM_2\right) = \sum_{j=1}^{L}\left(\frac{f_{1j}}{f_{1.}} - \frac{f_{2j}}{f_{2.}}\right)^2$$

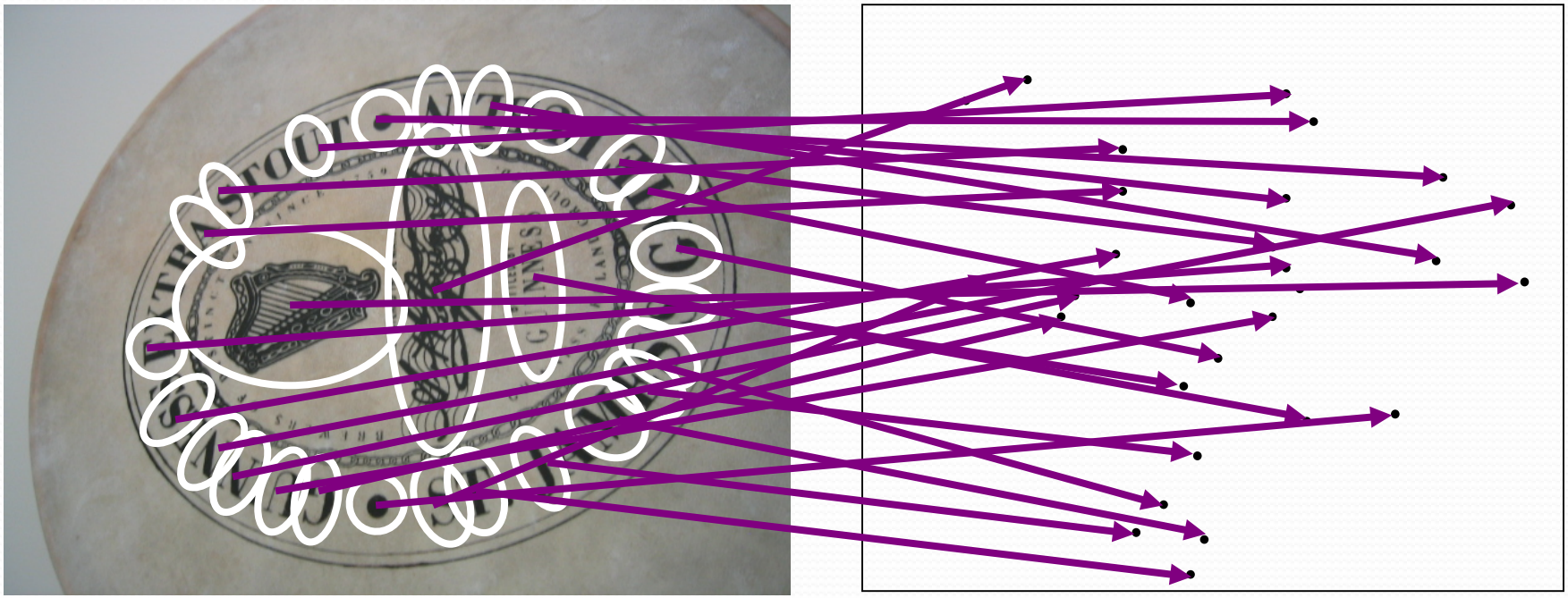
weighted by the mass of each of the words compared to all of the texts (total number of appearance of a word in all of the texts)

$$\chi^2\left(VSM_1, VSM_2\right) = \sum_{j=1}^{L}\frac{f}{f_{.j}}\left(\frac{f_{1j}}{f_{1.}} - \frac{f_{2j}}{f_{2.}}\right)^2$$

  - f corresponds to the total number of words in the set of documents
  - f_{ij} corresponds to the frequency of the word *j* in document *i*
  - $f_{i.}$ corresponds to the total number of words in document *i*
  - $f_{.j}$ corresponds to the frequency of the word *j* in the set of documents
- Very useful whenever you want to compare histograms, probability distributions...

Visual words: main idea

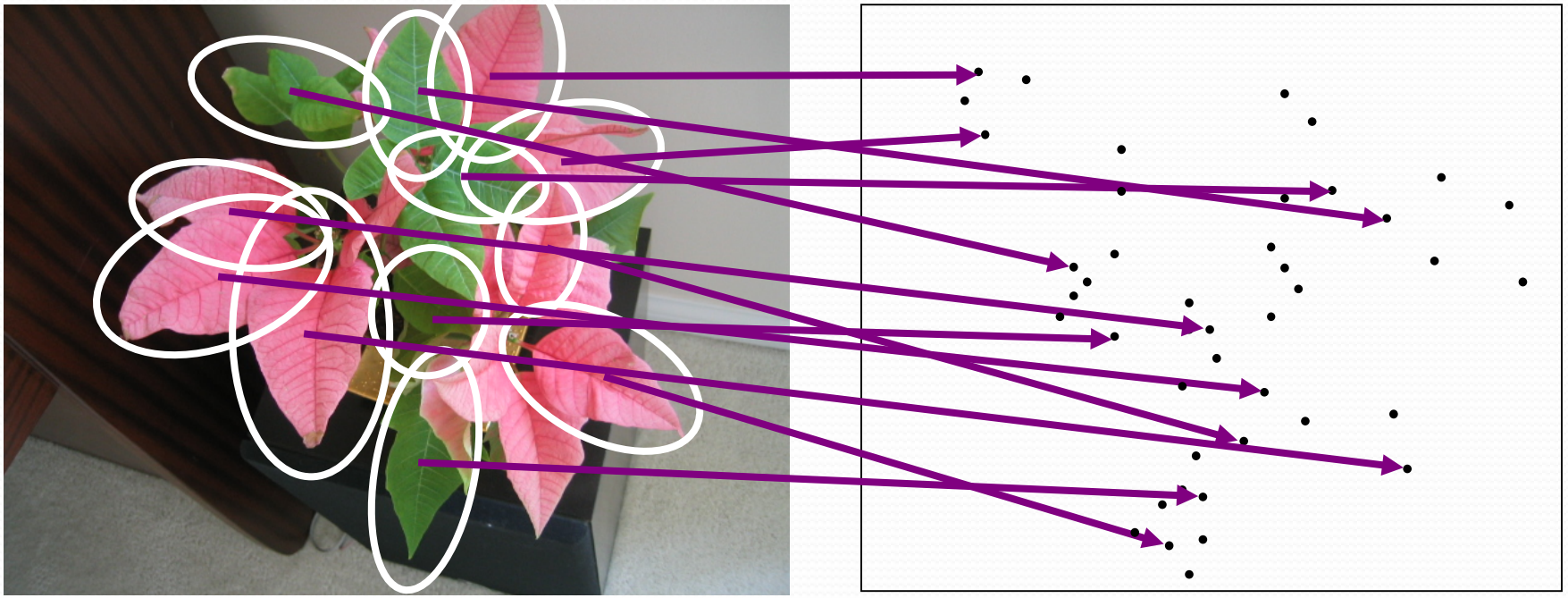
- Extract some local features from a number of images …



e.g., SIFT descriptor space: each point is 128-dimensional

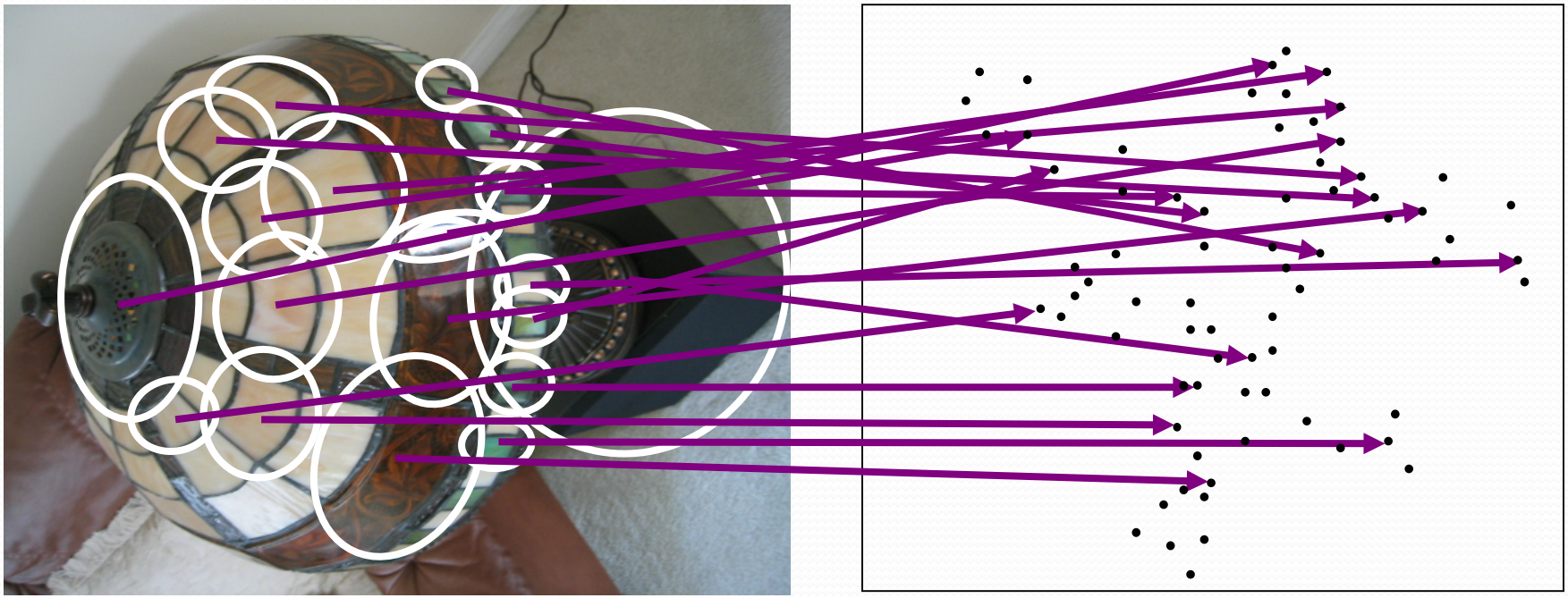
Slide credit: D. Nister

Visual words: main idea



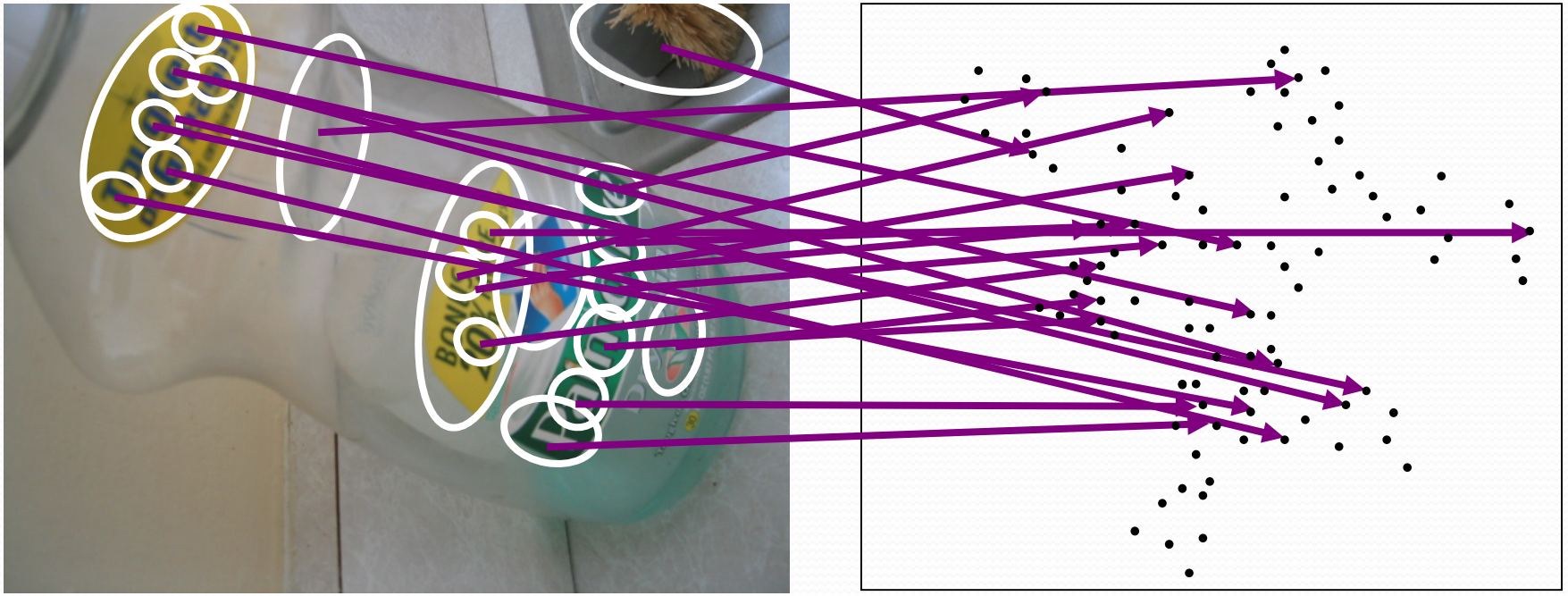
Slide credit: D. Nister

Visual words: main idea

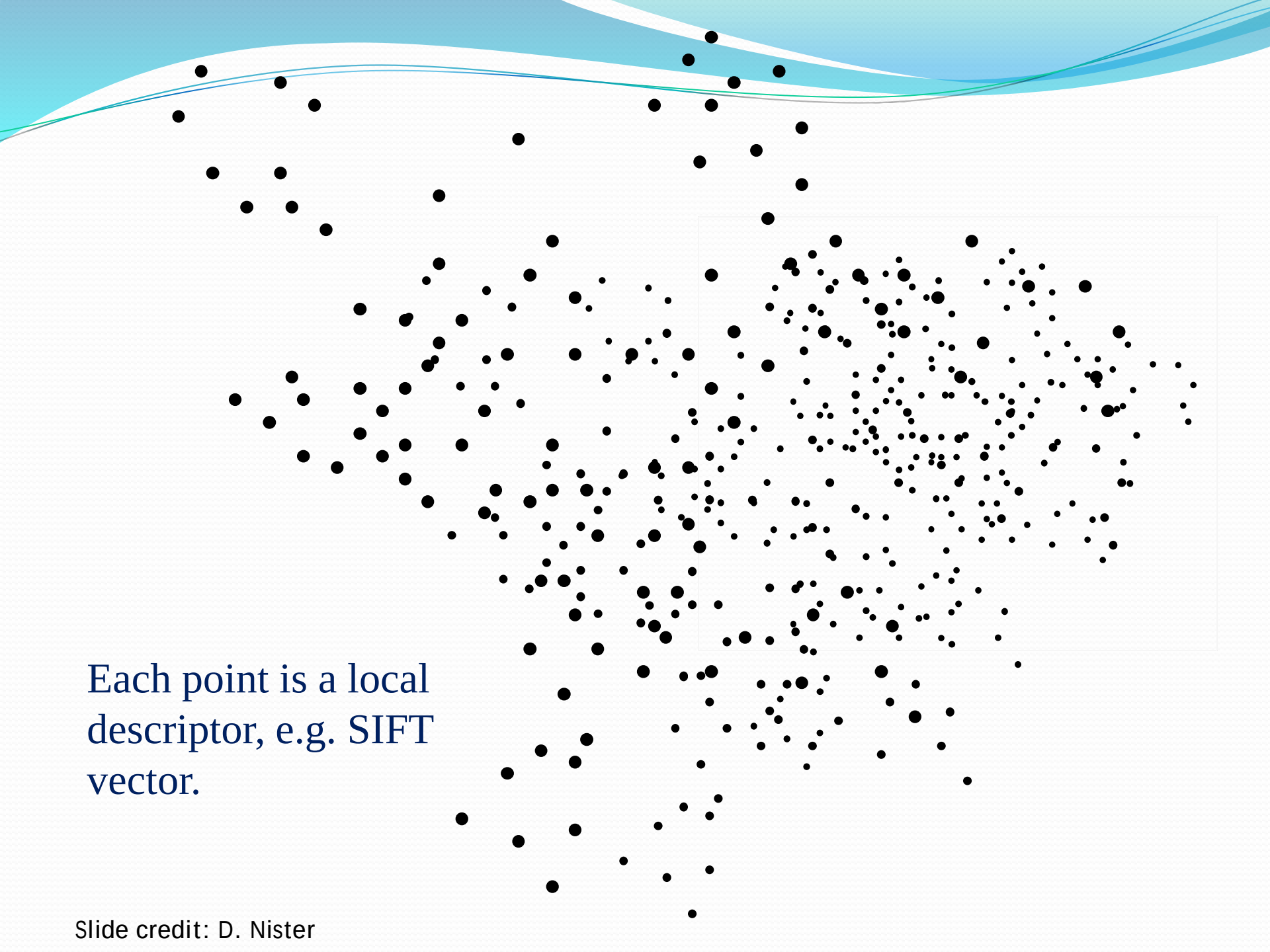


Slide credit: D. Nister

Visual words: main idea

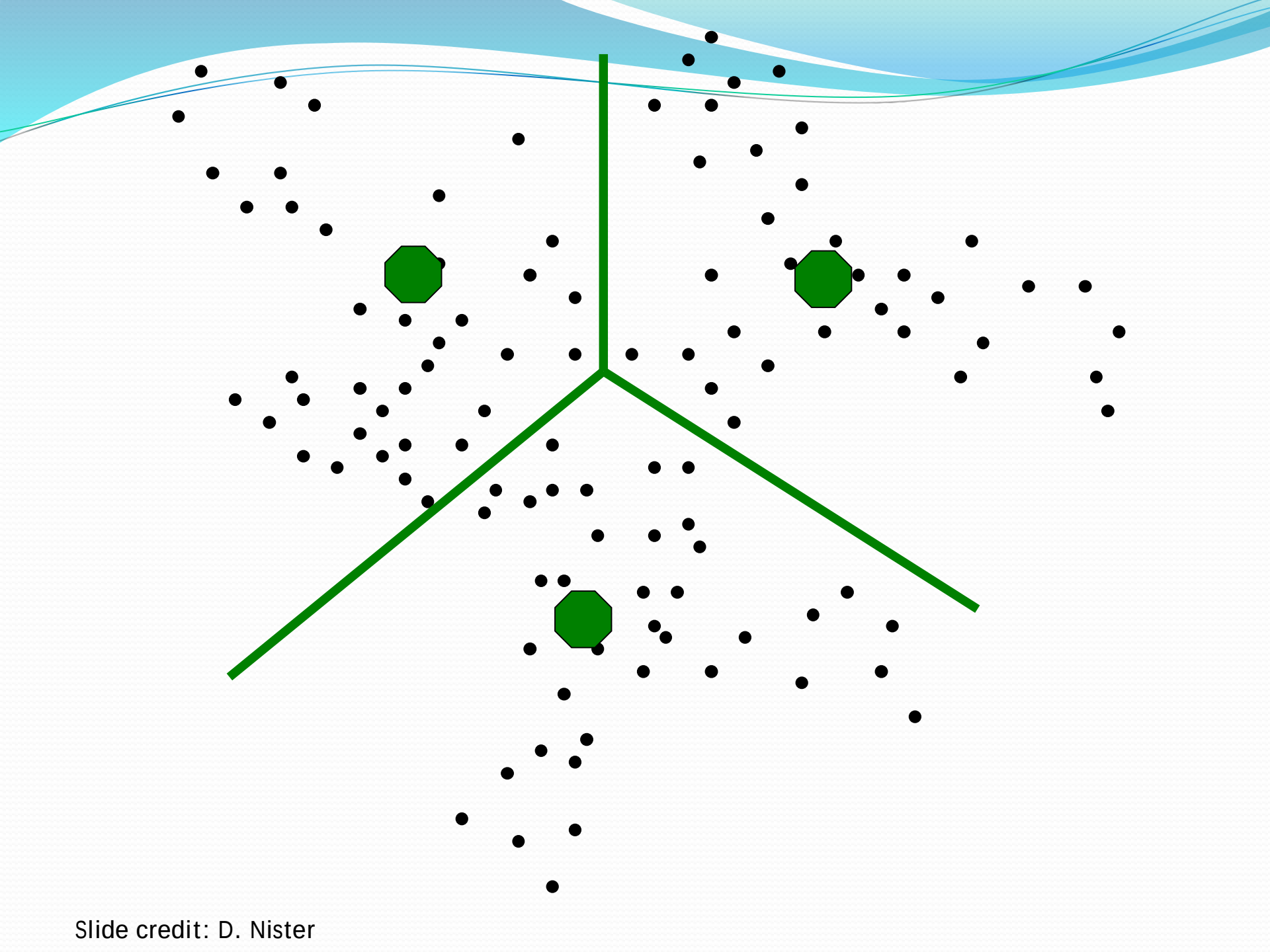


Slide credit: D. Nister



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

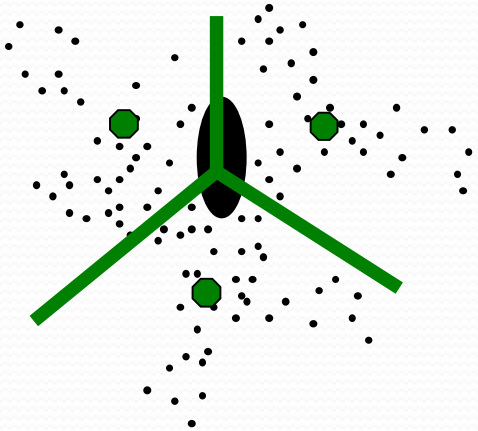
Slide credit: D. Nister



Slide credit: D. Nister

Visual words

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



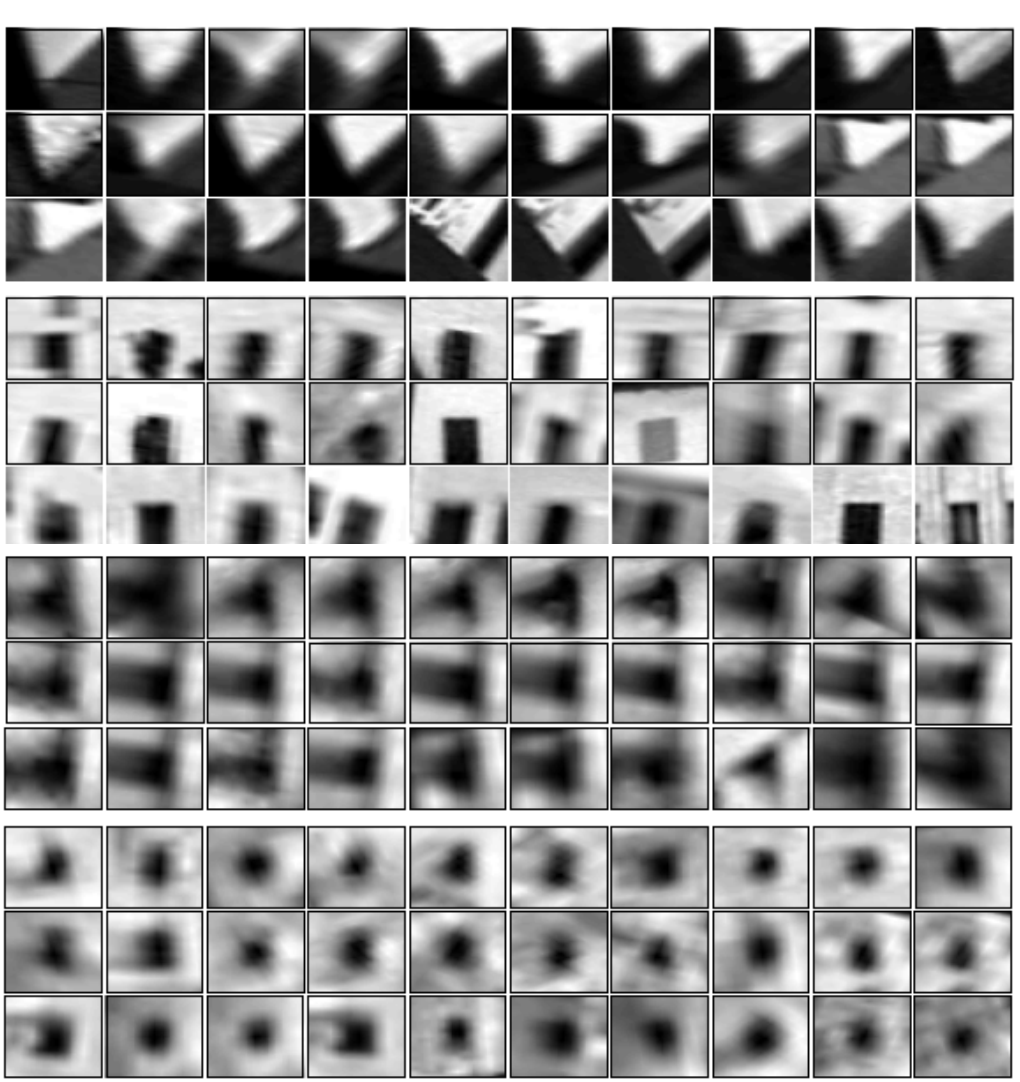
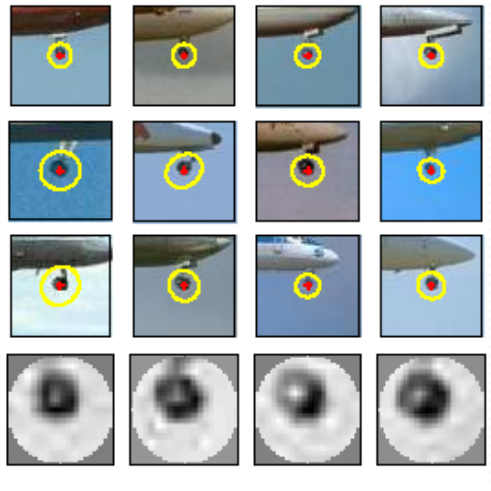
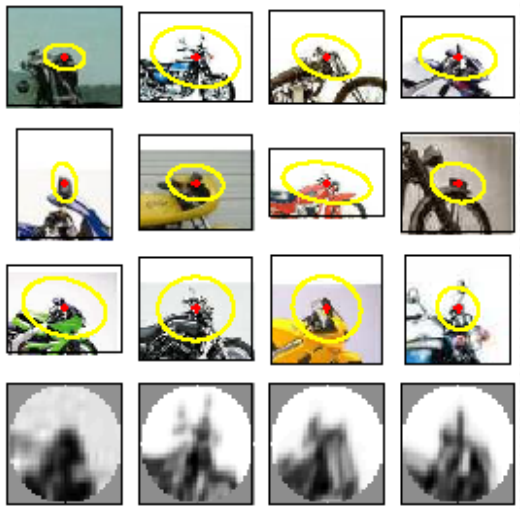


Figure from Sivic & Zisserman, ICCV 2003

Visual words

- They have also been used for describing scenes and objects for the sake of indexing or classification.

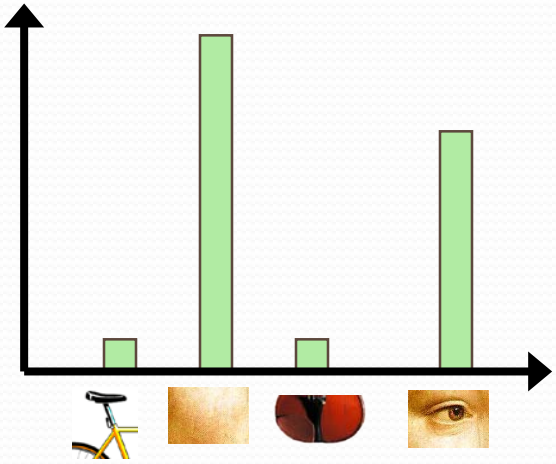


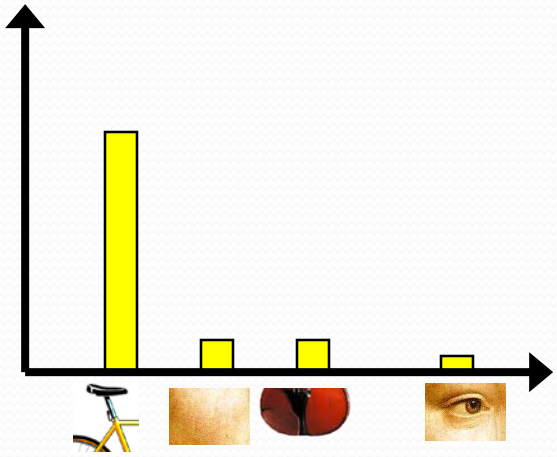


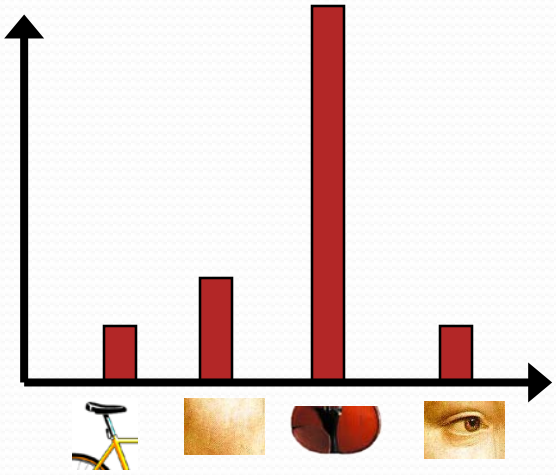
Sivic & Zisserman 2003; Csurka, Bray, Dance, & Fan 2004; many others.

Source credit: K. Grauman, B. Leibe

Object → Bag of ‘words’







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.

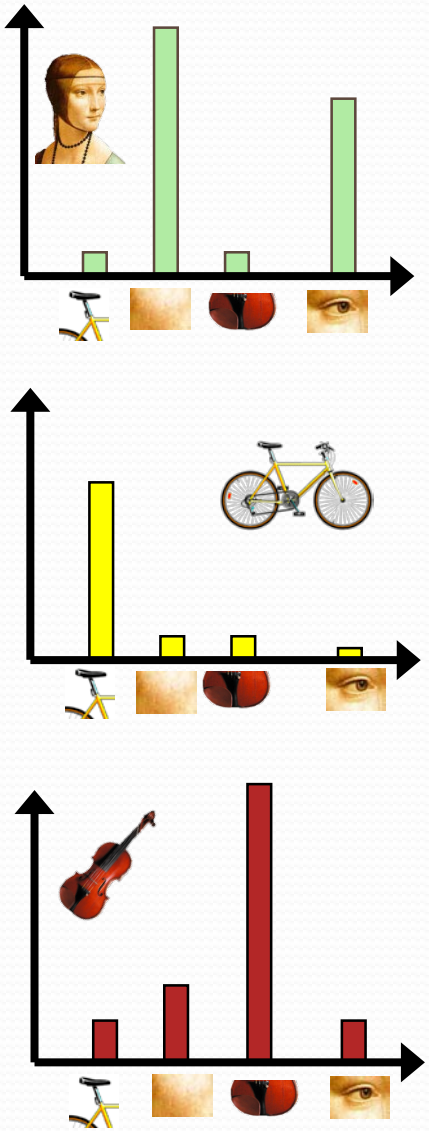
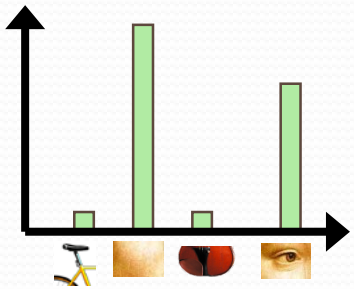


Image credit: Fei-Fei Li

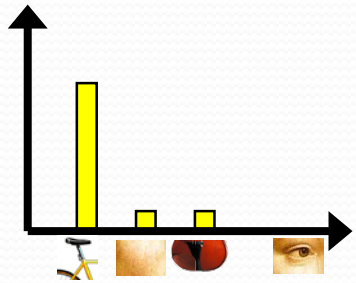
Comparing bags of words

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

[1 8 1 4]

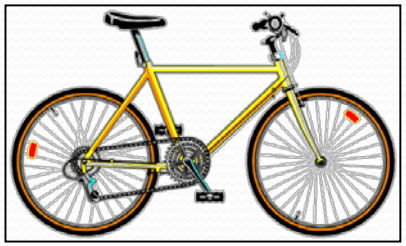


[5 1 1 0]



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

$\vec{d}_j$



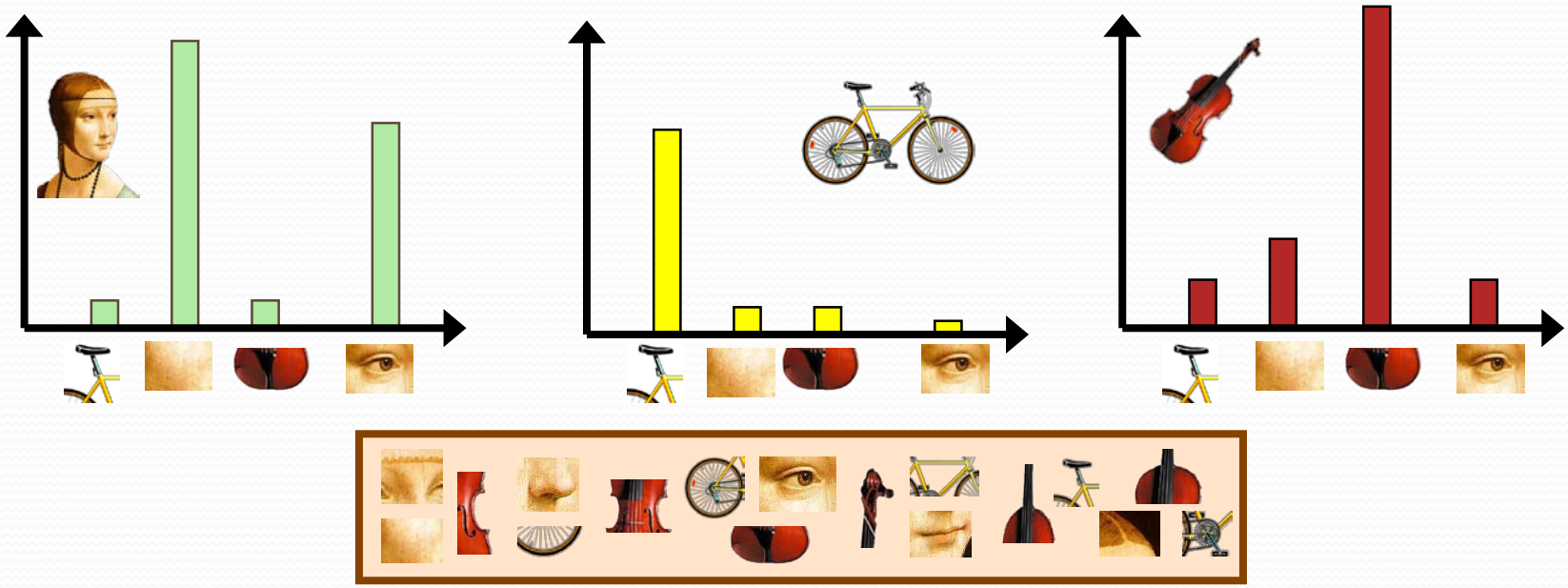
$\vec{q}$

Bag of Visual Words: online part

2 stages:

- Coding: projection of each local descriptor => dictionary
- Pooling (projection aggregation): statistical summary / Visual Word

=> Image global index: likelihood to contain each Visual Word



Coding

Notations :

- Les données images : $\mathbf{X} = \left\{ x_j \in \mathbb{R}^d \right\}, j \in \{1; N\}$
- Les centres : $\mathbf{C} = \{C_m\}, m \in \{1; M\}$
- Coding :

$$f : \mathbb{R}^d \longrightarrow \mathbb{R}^M$$

$$x_j \longrightarrow f(x_j) = \alpha_j = \{\alpha_{m,j}\}, \quad m \in \{1; M\}$$

$$\mathbf{H} = \begin{array}{c} \\ c_1 \\ \\ c_m \\ \\ c_M \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & & x_j & & x_N \end{array} \\ \begin{bmatrix} \alpha_{1,1} & \cdots & \alpha_{1,j} & \cdots & \alpha_{1,N} \\ \vdots & & \vdots & & \vdots \\ \alpha_{m,1} & \cdots & \alpha_{m,j} & \cdots & \alpha_{m,N} \\ \vdots & & \vdots & & \vdots \\ \alpha_{M,1} & \cdots & \alpha_{M,j} & \cdots & \alpha_{M,N} \end{bmatrix} \end{array} \Rightarrow g\text{: } pooling$$

$$\Downarrow$$

$$f\text{: } cooding$$

Coding

- Hard assignement: the simplest (cf text retriveal)

Hard coding: $f = f_Q$ assigns a constant weight to its closest center:

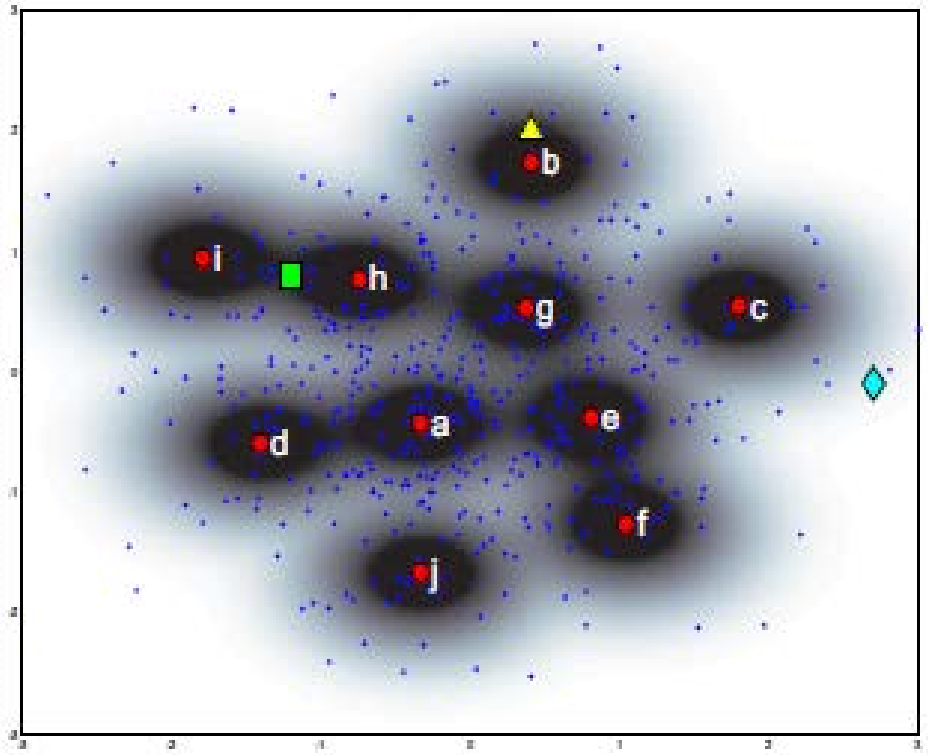
$$f_Q(x_j)[m] = \begin{cases} 1 & \text{if } m = \underset{k \in \{1;M\}}{\operatorname{argmin}} \|x_j - c_k\|^2 \\ 0 & \text{otherwise} \end{cases}$$

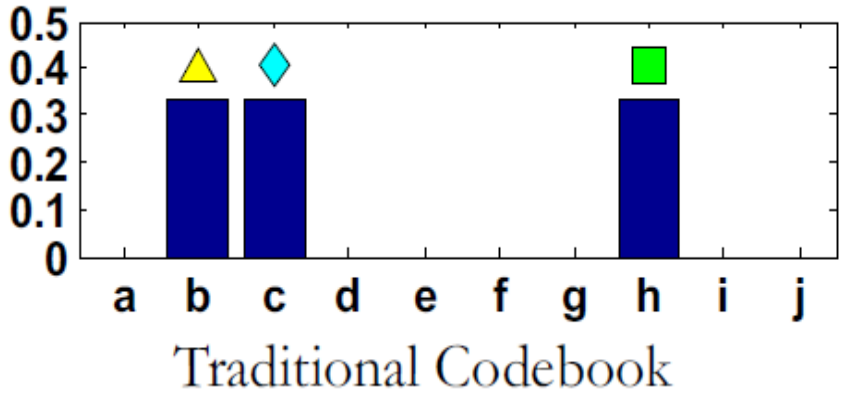
- To be more precise
  - Soft assignement
  - Sparse coding
  - Importance of the locality in coding

=> Image global index: likelihood to contain each Visual Word

Projection local signature => dictionary

- **The idea the simplest: hard assignement**
  - We seek the nearest cluster of the descriptor
  - We assign to this cluster a fixed weight (*e.g.* 1)





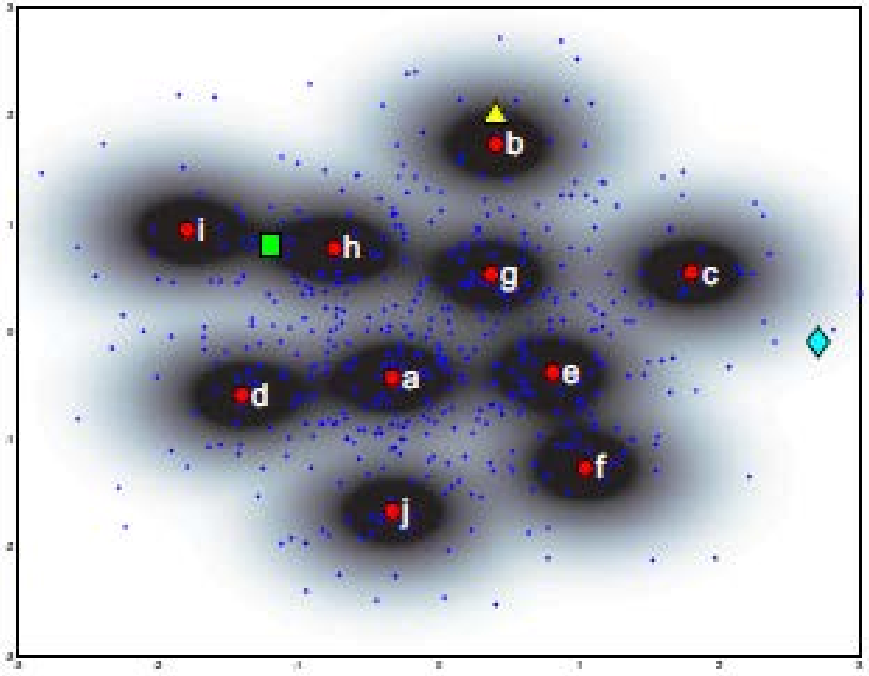
Projection local signature => dictionary

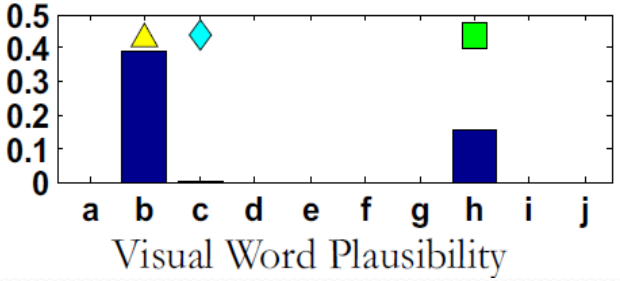
- **More sophisticated: soft assignement**

	Best Candidate	Multiple Candidates
Constant Weight	Traditional Codebook	Codeword Uncertainty
Kernel Weighted	Codeword Plausibility	Kernel Codebook

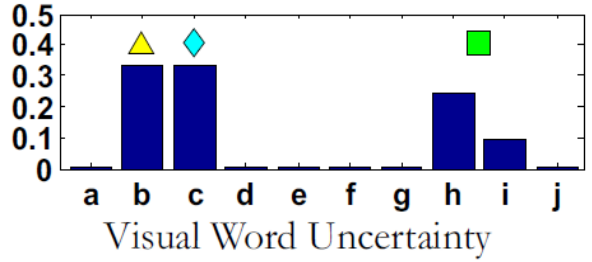
TABLE I

THE RELATIONSHIP BETWEEN VARIOUS FORMS OF CODEWORD AMBIGUITY AND THEIR PROPERTIES.

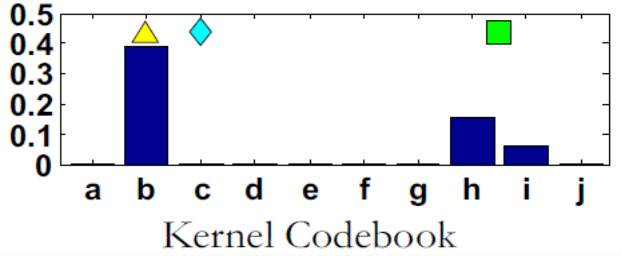




Visual Word Plausibility *signifies that an image feature may not be close enough to warrant representation by any relevant codeword in the vocabulary*



Visual Word Uncertainty *indicates that one image region may distribute probability mass to more than one codeword.*



Projection local signature => dictionary

- **Soft vs hard assignement**
  - In practice, the gain of the soft / hard is not always clear
  - Only the strategy based on *Uncertainty* improves the performance of classification
- **Other approach: sparse coding**
  - We approximate each local descriptor as a linear combination of a subset of $\boldsymbol{x}_i$ dictionary words = $D\,\alpha$ $\boldsymbol{x}_i = D\,\alpha$

Projection local signature => dictionary

- **Sparse coding**
  - We approximate each local descriptor as a linear combination of a subset of $\boldsymbol{x}_i$ dictionary words = $D\ \alpha$
    $\boldsymbol{x}_i = D\ \alpha$
    - α weight, D dictionary
    - We force each $\boldsymbol{x}_i$ to be represented only by a small number of visual words => sparsity

$$\alpha_i = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} L(\alpha, \mathrm{D}) \triangleq \|x_i - \mathrm{D}\alpha\|_2^2 + \lambda\|\alpha\|_1$$

Projection local signature => dictionary

- **Sparse coding vs VQ: hard assignement standard**
  - SC: Sparse Coding = majority of $\alpha_i = 0$
  - LLC: Local LinearCoding = the words representing the input must be close (locality)
  - Open question: relevance of these minimization criteria on the error of reconstruction in classification ?

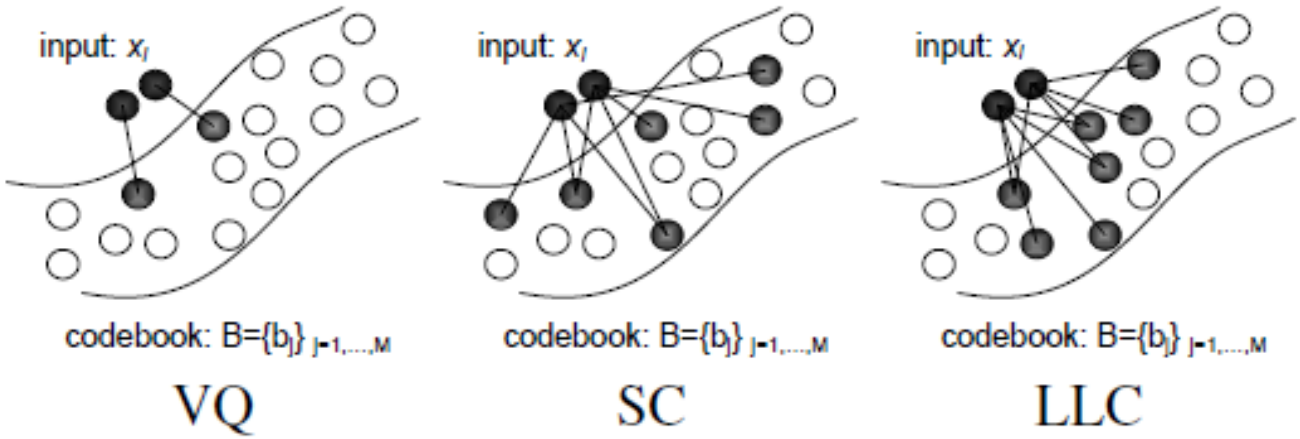
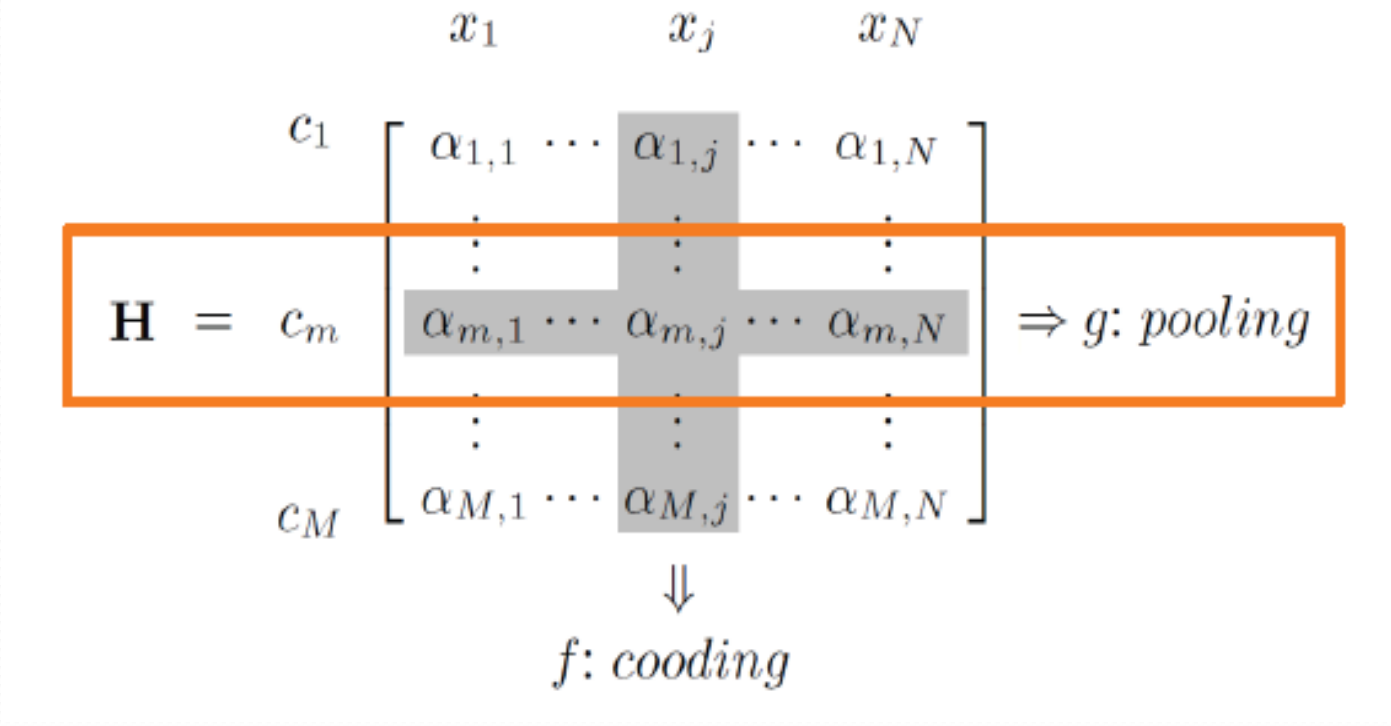


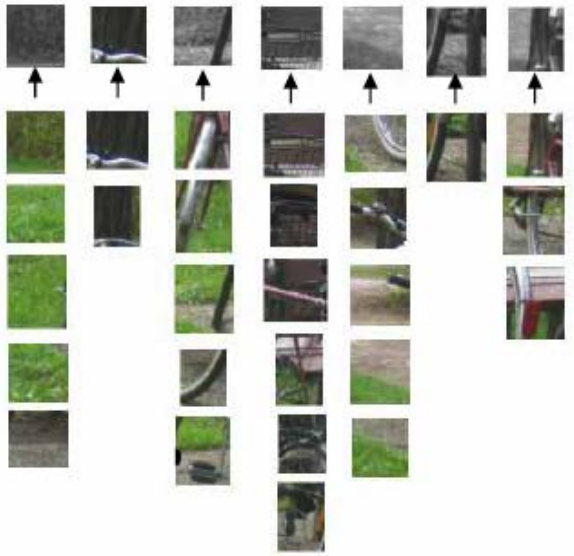
Figure 2. Comparison between VQ, SC and LLC. *The selected basses for representation are highlighted in red*

Pooling: projection aggregation => image global index



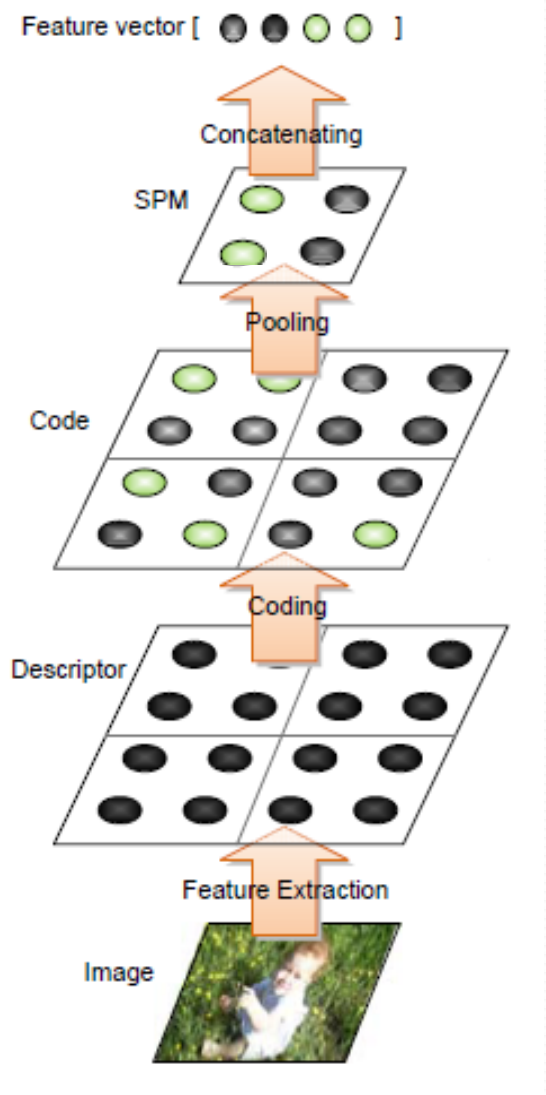
Pooling: projection aggregation => image global index

- Global index: likelihood for the image to contain each visual word
- Different ways to summarize the projections
  - Sum pooling: the most classical BoW approach
    - From text retrieval: count word occurrences in the



Pooling: projection aggregation => image global index

- Global index: likelihood for the image to contain each visual word
- Different ways to summarize the projections
  - Max pooling: we keep the max value of the projection for each visual word
    - Interesting for sparse / soft coding: reduce the impact of the noise
    - Somehow linked to biological inspiration (cortex)

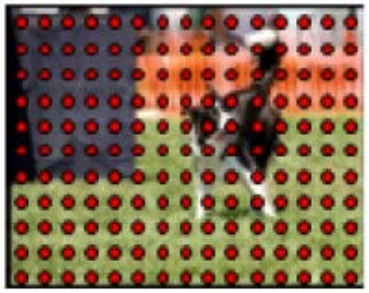


Pooling: projection aggregation => image global index

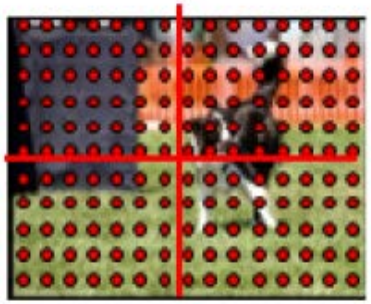
- BoW: histogram: spatial information about the location of the local descriptors is lost
  - how to store this information?
- Apply the BoW approach in different sub-regions of the image
  - Concatenation of BoWs ?
  - Get a similarity measure combining all levels
    - Spatial Pyramid Matching [LAZEBNIK06]
    - Extension to the spatial domaine of the pyramid match kernel[GRAUMAN05]

Projection aggregation => image global index

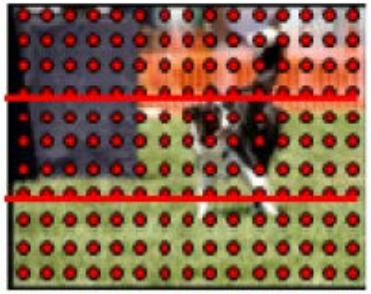
1X1

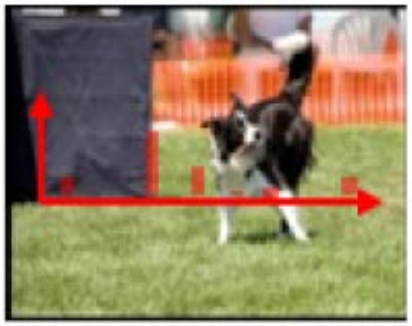


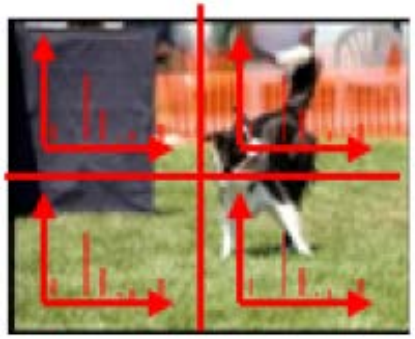
2X2

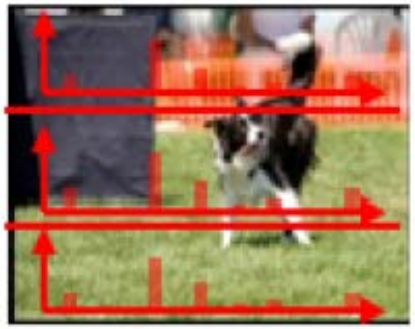


3X1









Two extensions of the BoW

- Spatial Pyramid (Lazebnik, Schmid & Ponce)

  Pyramid in image space, quantize features

  ⇒**Limit the global invariance:**

  S(  faible

  [CVPR 06: Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories]
  Slide credits: Svetlana Lazebnik

- Pyramid Match Kernel (Grauman & Darrell)

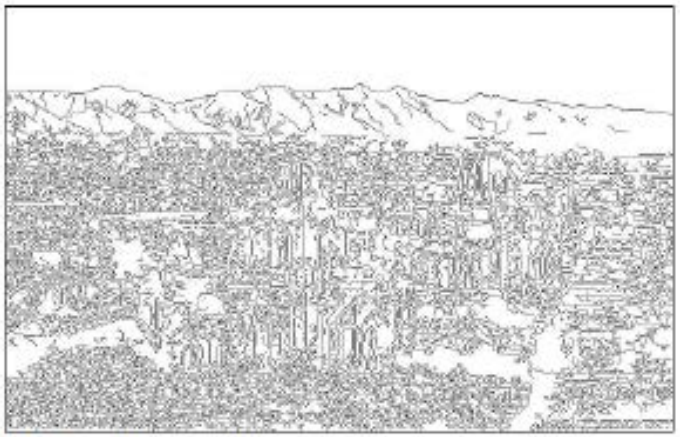
  Pyramid in feature space, ignore location

Spatial Pyramid

- Extract interest point descriptors (dense scan)
- Construct visual word dictionary
- Build spatial histograms
- Train an SVM

Spatial Pyramid

- **Extract interest point descriptors (dense scan)**
- Construct visual word dictionary
- Build spatial histograms
- Train an SVM



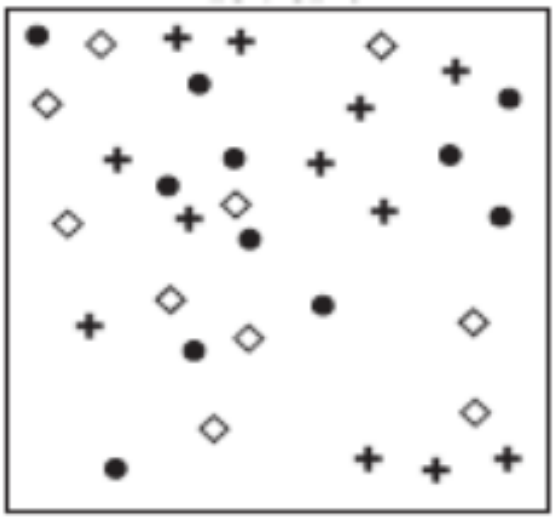
Weak (edge orientations)

OR

Strong (SIFT)

Spatial Pyramid

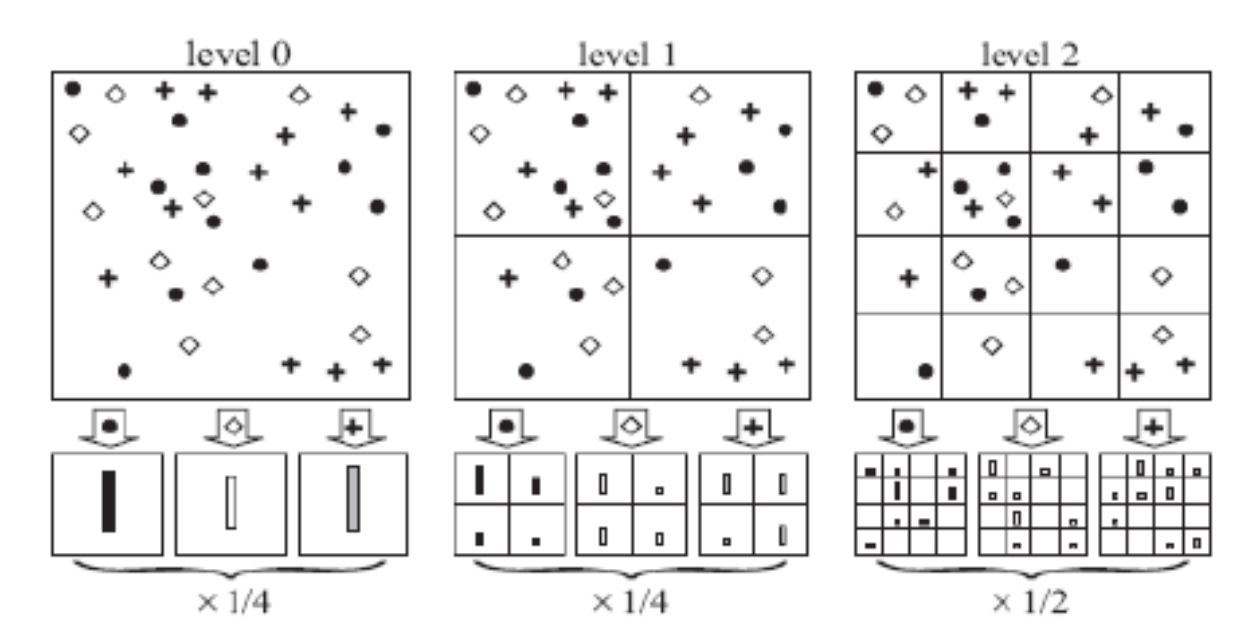
- Extract interest point descriptors (dense scan)
- **Construct visual word dictionary**
- Build spatial histograms
- Train an SVM



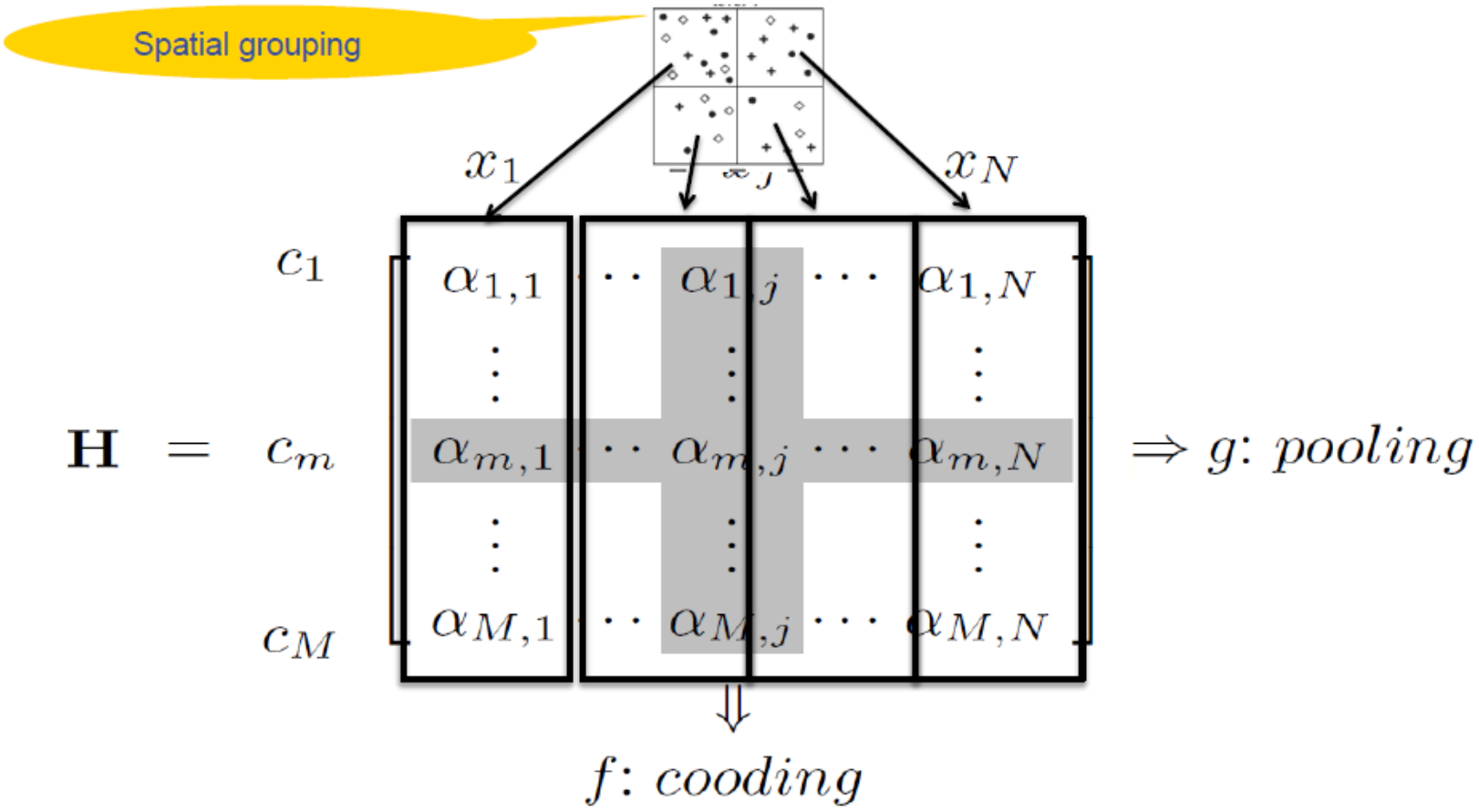
- Vector quantization
- Usually K-means clustering
- Vocabulary size (16 to 400)

Spatial Pyramid

- Extract interest point descriptors (dense scan)
- Construct visual word dictionary
- **Build spatial histograms**
- Train an SVM



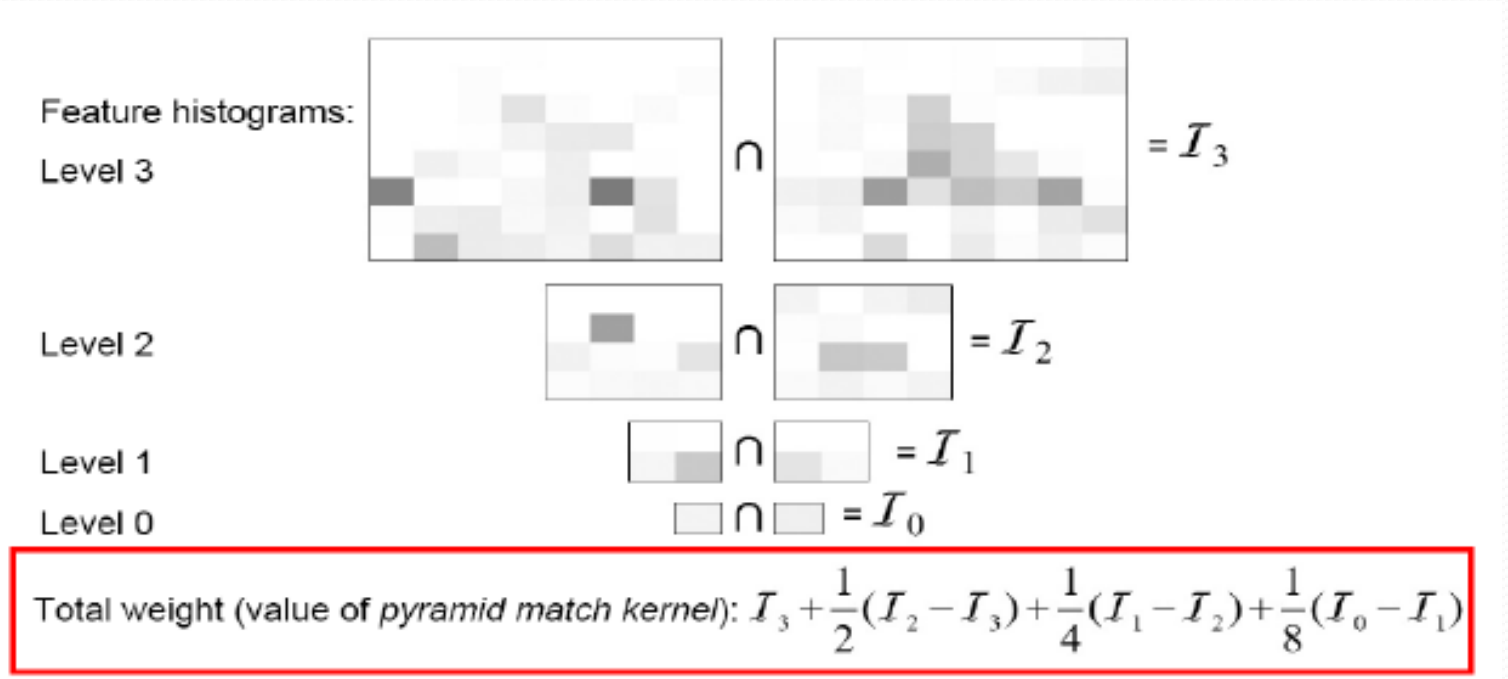
Spatial Pyramid



=> Break the global invariance because the pyramide is fixed

Spatial Pyramid

- Extract interest point descriptors (dense scan)
- Construct visual word dictionary
- Build spatial histograms
- **Train an SVM**



Two extension of BoW

- Spatial Pyramid (Lazebnik)

  *Pyramid in **image** space, quantize features*

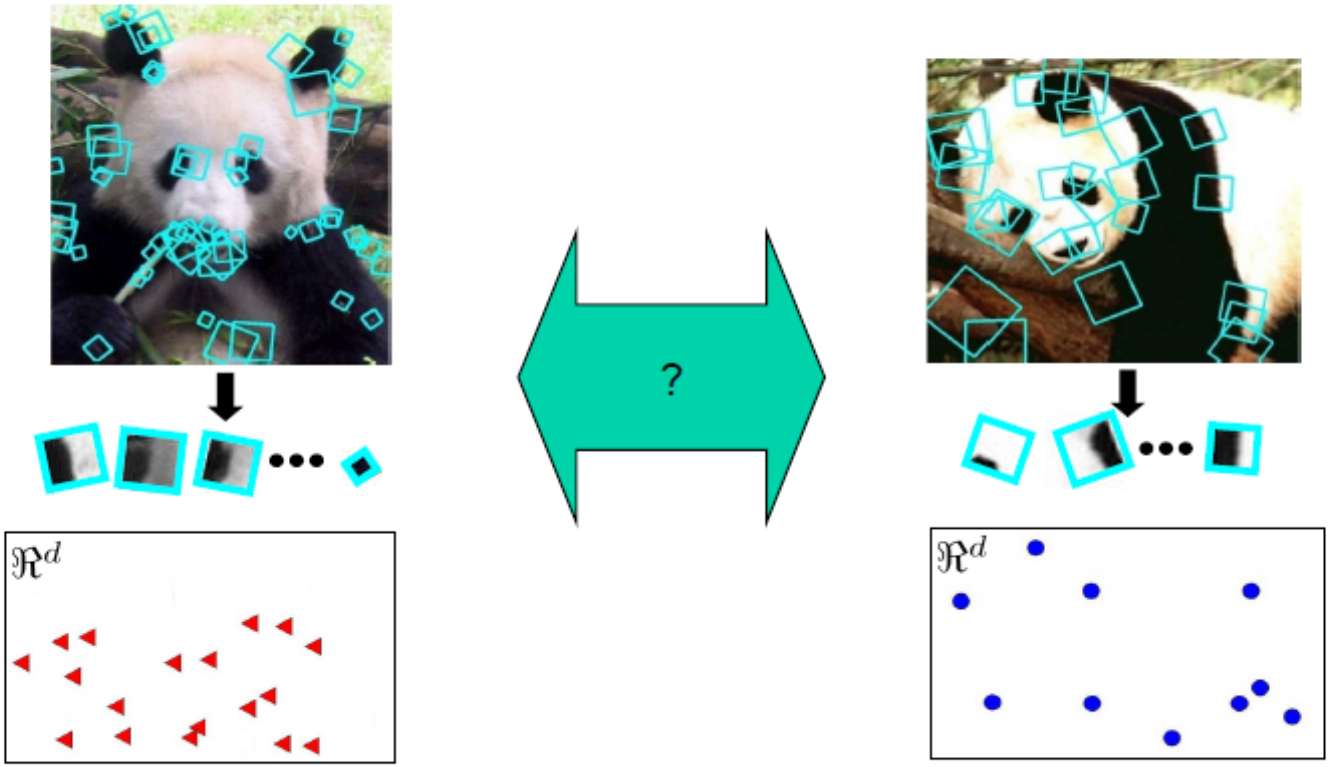
- **Pyramid Match Kernel (Grauman & Darrell)**

  *Pyramid in **feature** space, ignore location*

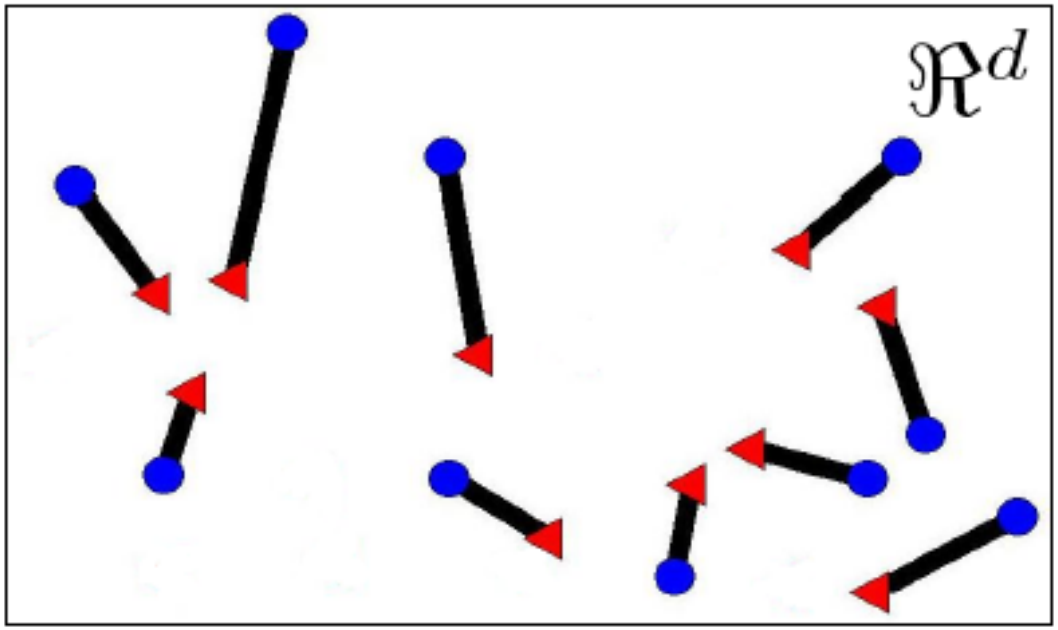
  => Idée de "corriger" les imperfections du dico

How to compare Sets of Features?

- Each instance is unordered set of vectors
- Varying number of vectors per instance



Correspondence-based Match

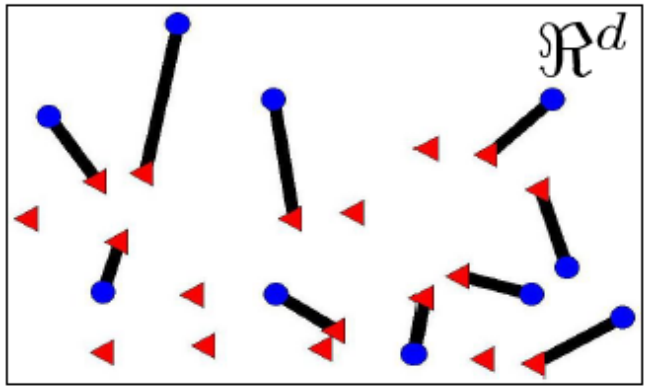


Explicit search for correspondences…

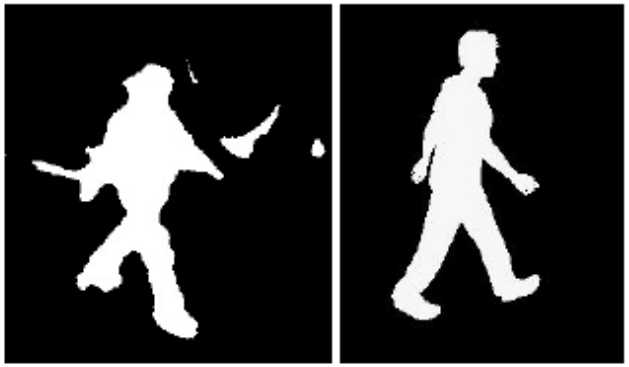
$$\max_{\pi:\mathbf{X}\to\mathbf{Y}} \sum_{\mathbf{x}_i\in\mathbf{X}} \mathcal{S}(\mathbf{x}_i, \pi(\mathbf{x}_i))$$

Partial Matching

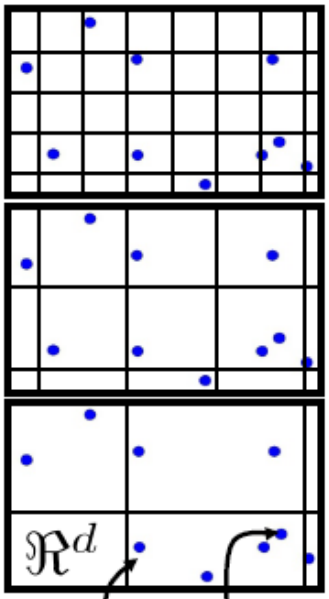
Compare sets by computing a *partial matching* between their features.

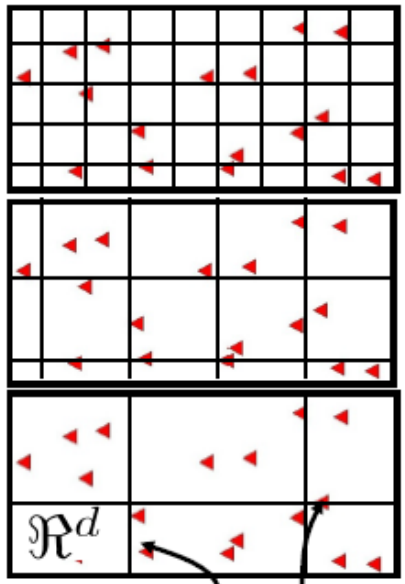


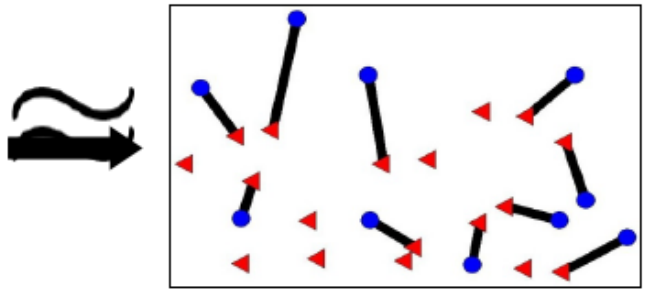
$$\max_{\pi:\mathbf{X}\to\mathbf{Y}} \sum_{\mathbf{x}_i\in\mathbf{X}} \mathcal{S}(\mathbf{x}_i, \pi(\mathbf{x}_i))$$



Pyramid Matching





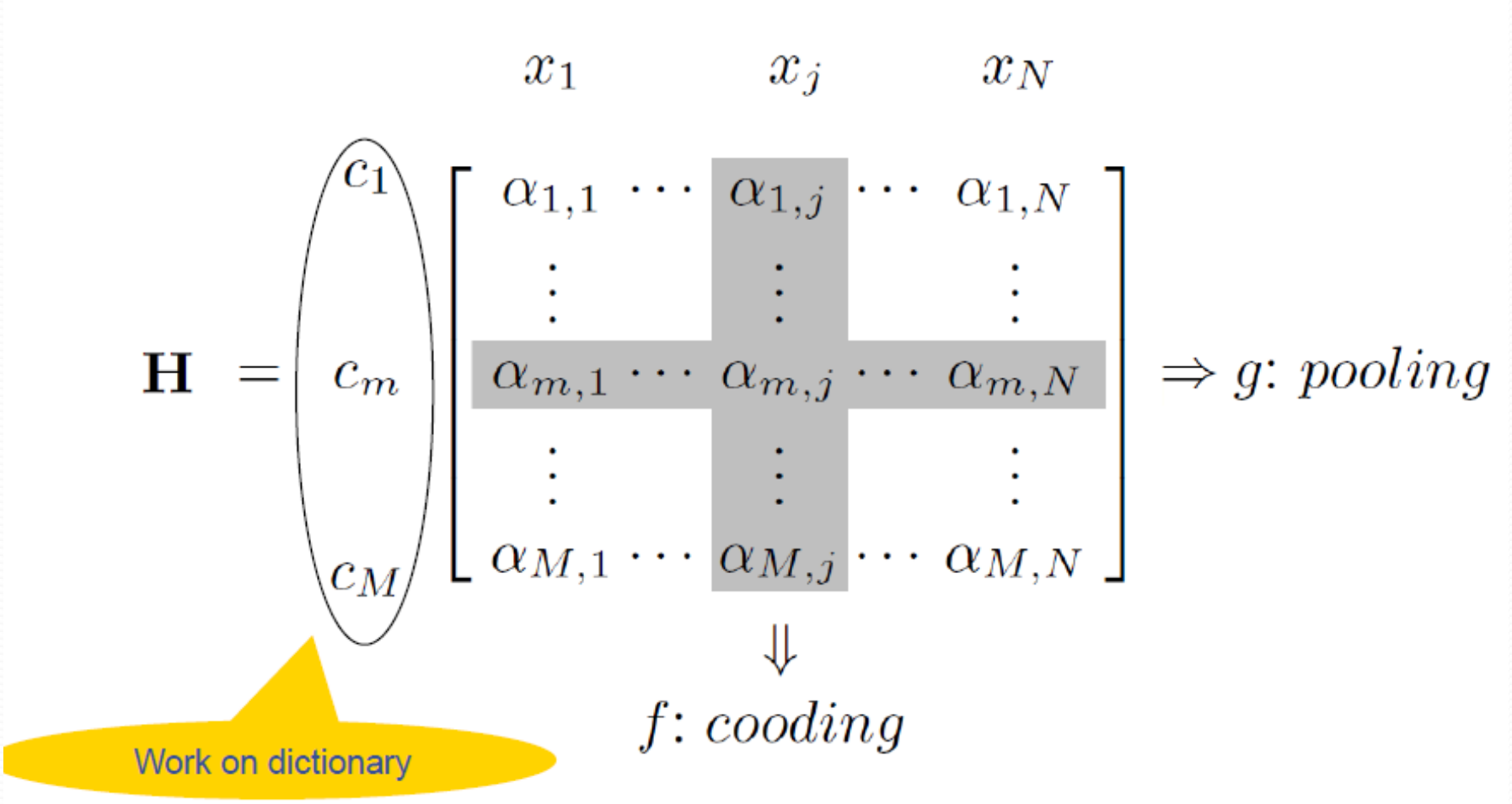


optimal partial matching

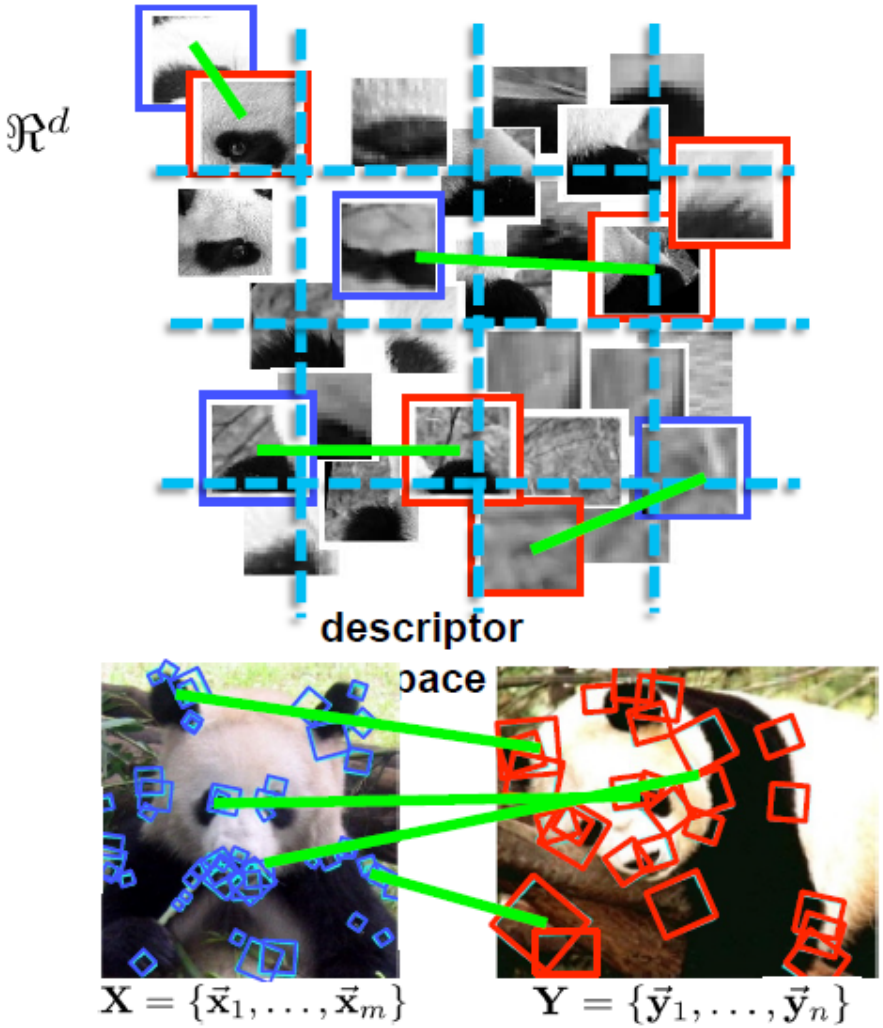
$$\max_{\pi:\mathbf{X}\to\mathbf{Y}} \sum_{\mathbf{x}_i\in\mathbf{X}} \mathcal{S}(\mathbf{x}_i, \pi(\mathbf{x}_i))$$

$\mathbf{X} = \{\vec{\mathbf{x}}_1, \ldots, \vec{\mathbf{x}}_m\}$ $\mathbf{Y} = \{\vec{\mathbf{y}}_1, \ldots, \vec{\mathbf{y}}_n\}$

Pyramid Matching

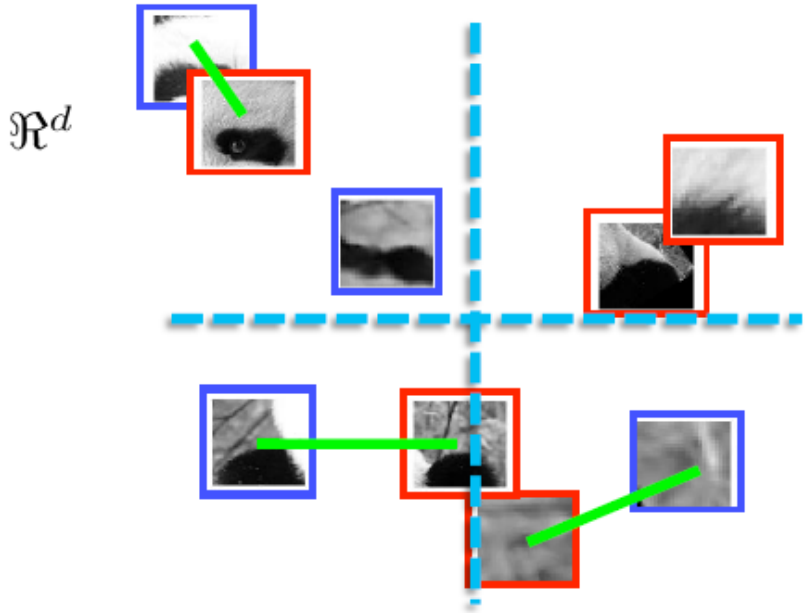


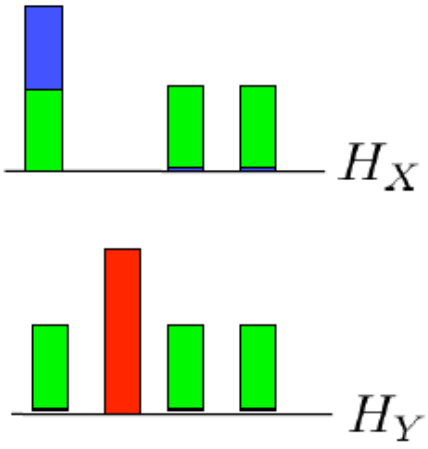
Pyramid Matching: Main Idea



Feature space partitions serve to "match" the local descriptors within successively wider regions.

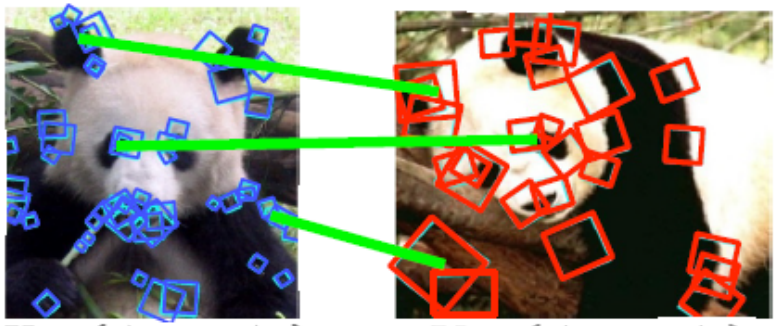
Pyramid Matching: Main Idea





$$\mathcal{I}(H_X, H_Y) = \sum_j \min\left(H_X(j), H_Y(j)\right)$$
$$= 3$$

Histogram intersection counts number of possible matches at a given partitioning.



$\mathbf{X} = \{\vec{\mathbf{x}}_1, \ldots, \vec{\mathbf{x}}_m\}$ $\mathbf{Y} = \{\vec{\mathbf{y}}_1, \ldots, \vec{\mathbf{y}}_n\}$

Pyramid Match Kernel

$$K_\Delta(X,Y) = \sum_{i=0}^{L} \underbrace{2^{-i}}_{\text{measures difficulty of a match at level } i} \underbrace{\mathcal{I}\left(H_X^{(i)}, H_Y^{(i)}\right) - \mathcal{I}\left(H_X^{(i-1)}, H_Y^{(i-1)}\right)}_{\text{number of newly matched pairs at level } i}$$

- For similarity, weights inversely proportional to bin size (or may be learned)
- Normalize these kernel values to avoid favoring large sets

[Grauman & Darrell, ICCV 2005]

Pyramid Match Kernel