# Content-Based Image Retrieval

## (以图搜图)

Jun Wu (邬俊)

School of Computer & Information Technology

Beijing Jiaotong University, Beijing, China

URL: http://faculty.bjtu.edu.cn/8620/

# A Multifaceted Definition of CV

- **Automatic *understanding* of image/video (pixels)**
  - **Interpretation**
    - ***Recognizing*** objects, people, scenes and activities from visual data
  - **Measurement**
    - Computing ***3D properties*** from visual data
  - **Search & management**
    - Finding out useful information from ***massive*** visual data

# Retrieving Specific Objects

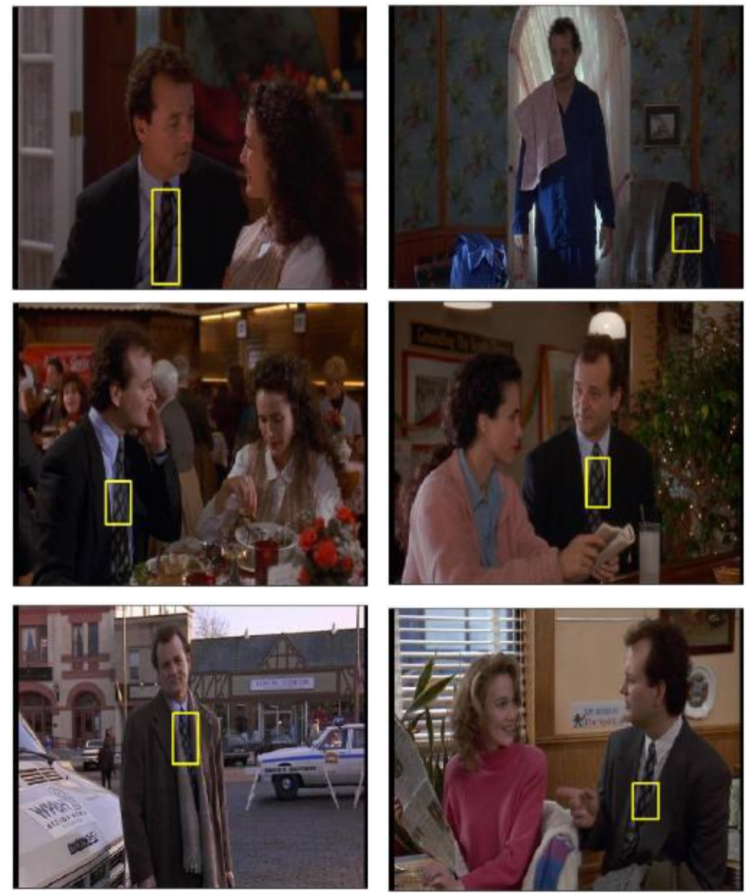
Search photos on the web for particular places

Find these landmarks ...in these images and 1M more

# Retrieving Specific Objects

Visual search in feature films

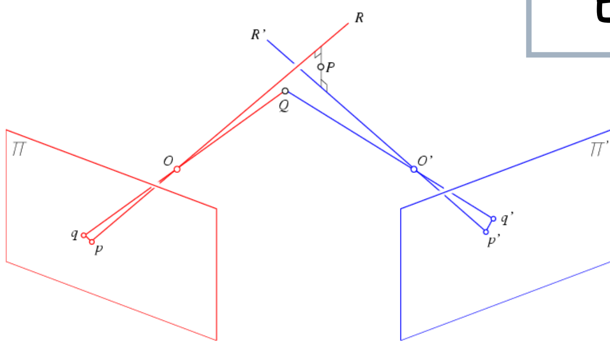
Visually defined query



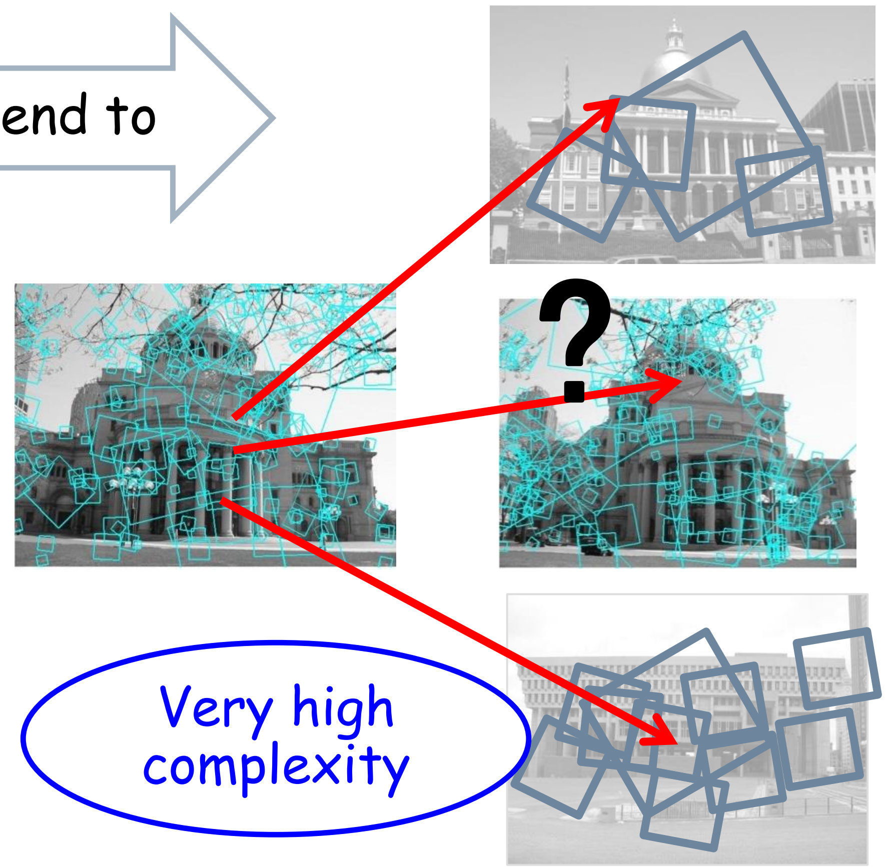
Retrieved frames

“Groundhog Day” [Rammis, 1993]

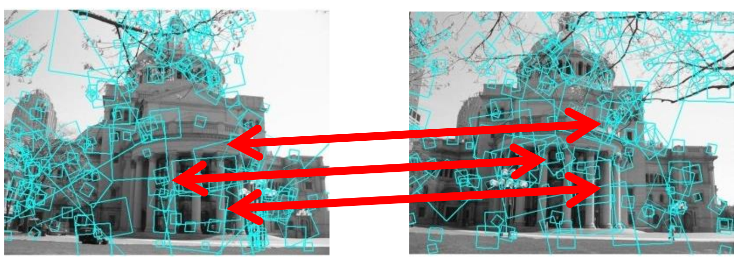
# Matching Local Features



Extend to



Very high complexity

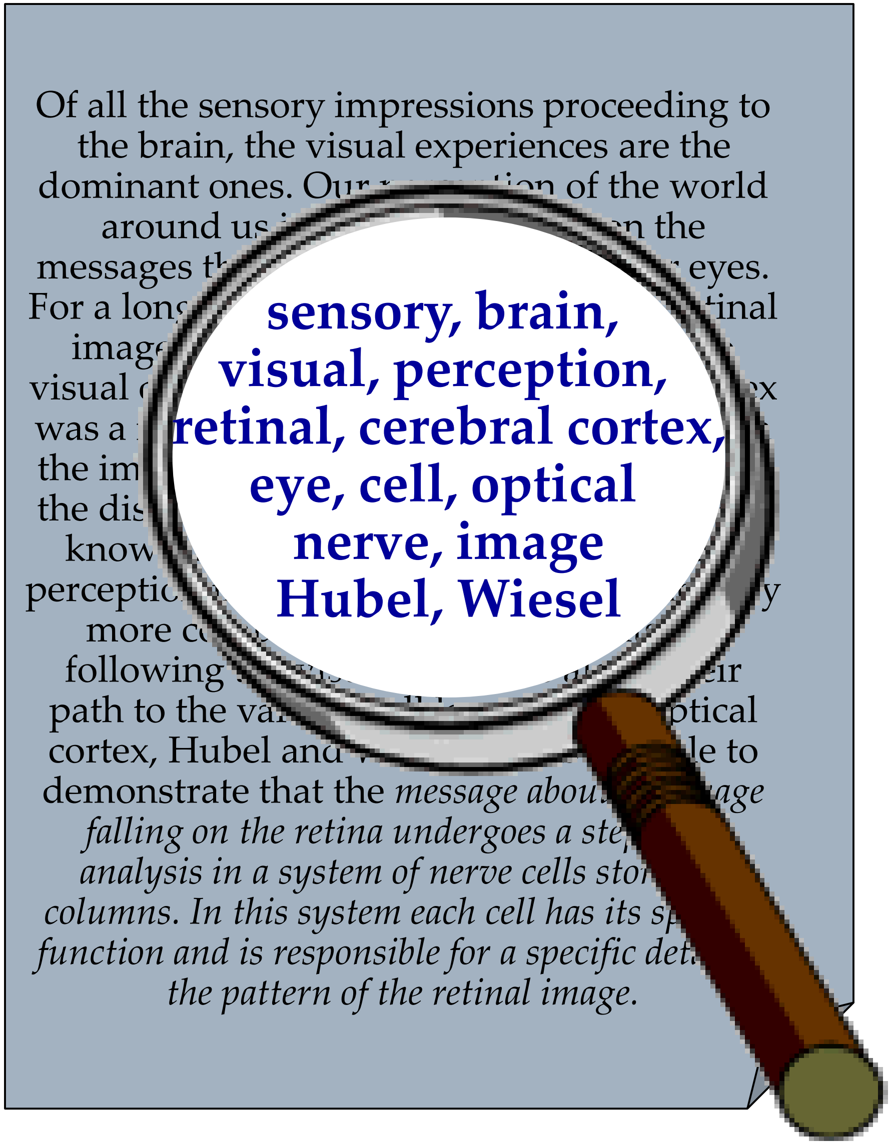


Multi-view matching for the same object

Search for matched views from an collection

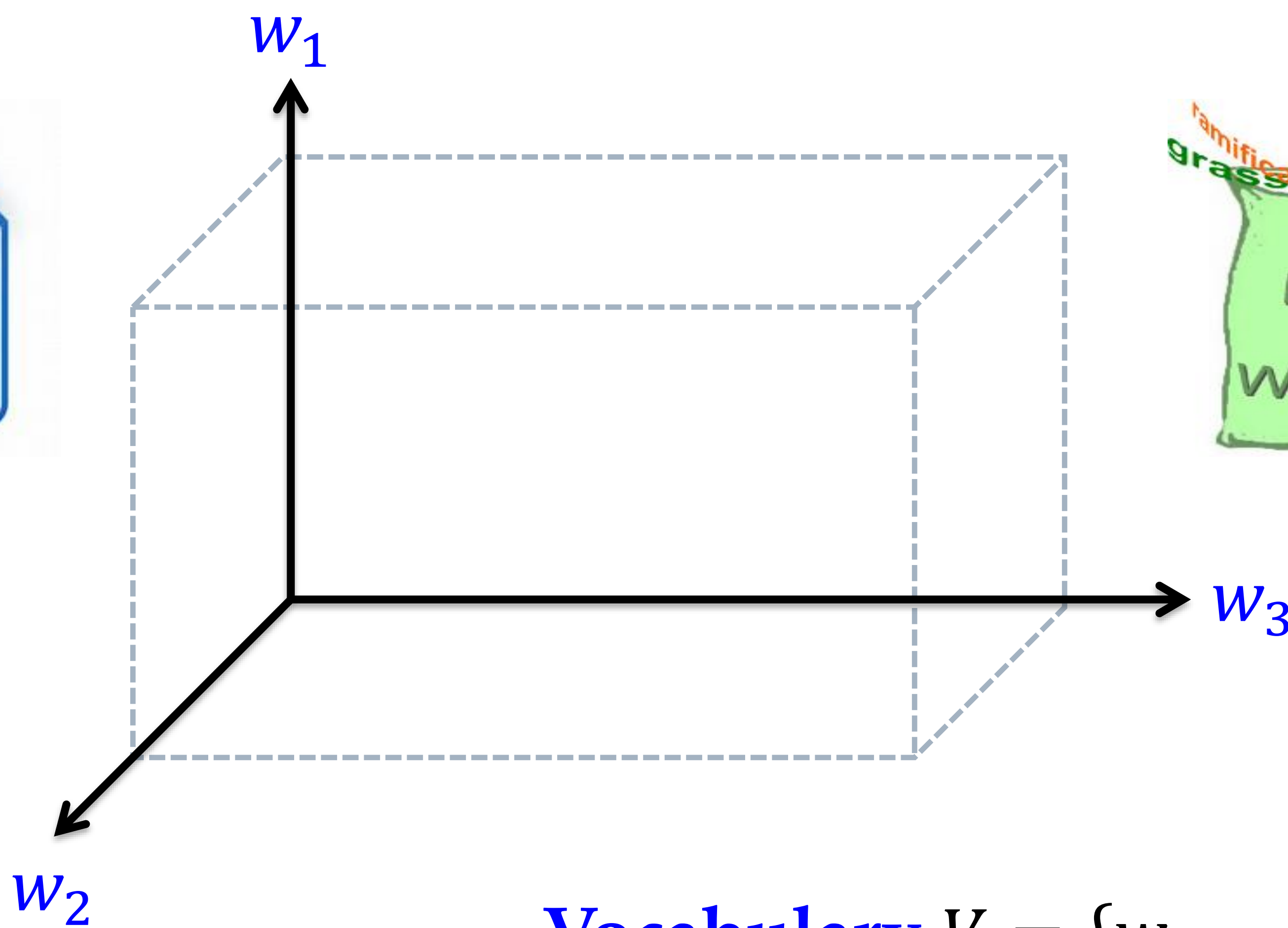
:

# Bag of Words Models


Of all the sensory impressions proceeding to
the brain, the visual experiences are the
sensory, brain,
visual, perception,
retinal, cerebral cortex,
eye, cell, optical
nerve, image
Hubel, Wiesel
the pattern of the retinal image.


China is forecasting a trade surplus of $90bn
(£51bn) to $100bn this year, a threefold
increase on 2004's $32bn. The Commerce
China, trade,
surplus, commerce,
exports, imports, US,
yuan, bank, domestic,
foreign, increase,
trade, value
further in value.

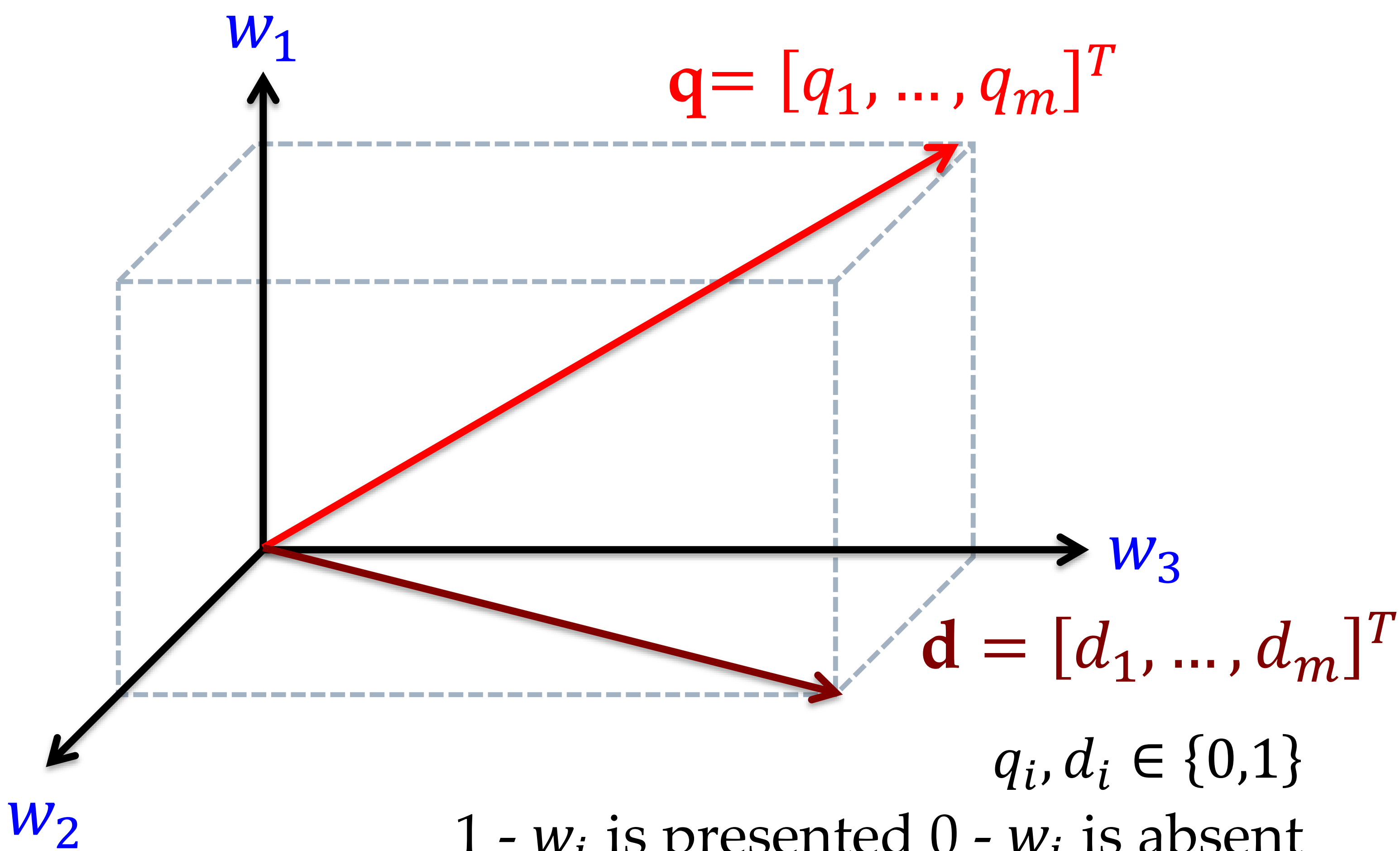
# Dimension Instantiation: BoW





**Vocabulary** $V = \{w_1, \cdots, w_m\}$

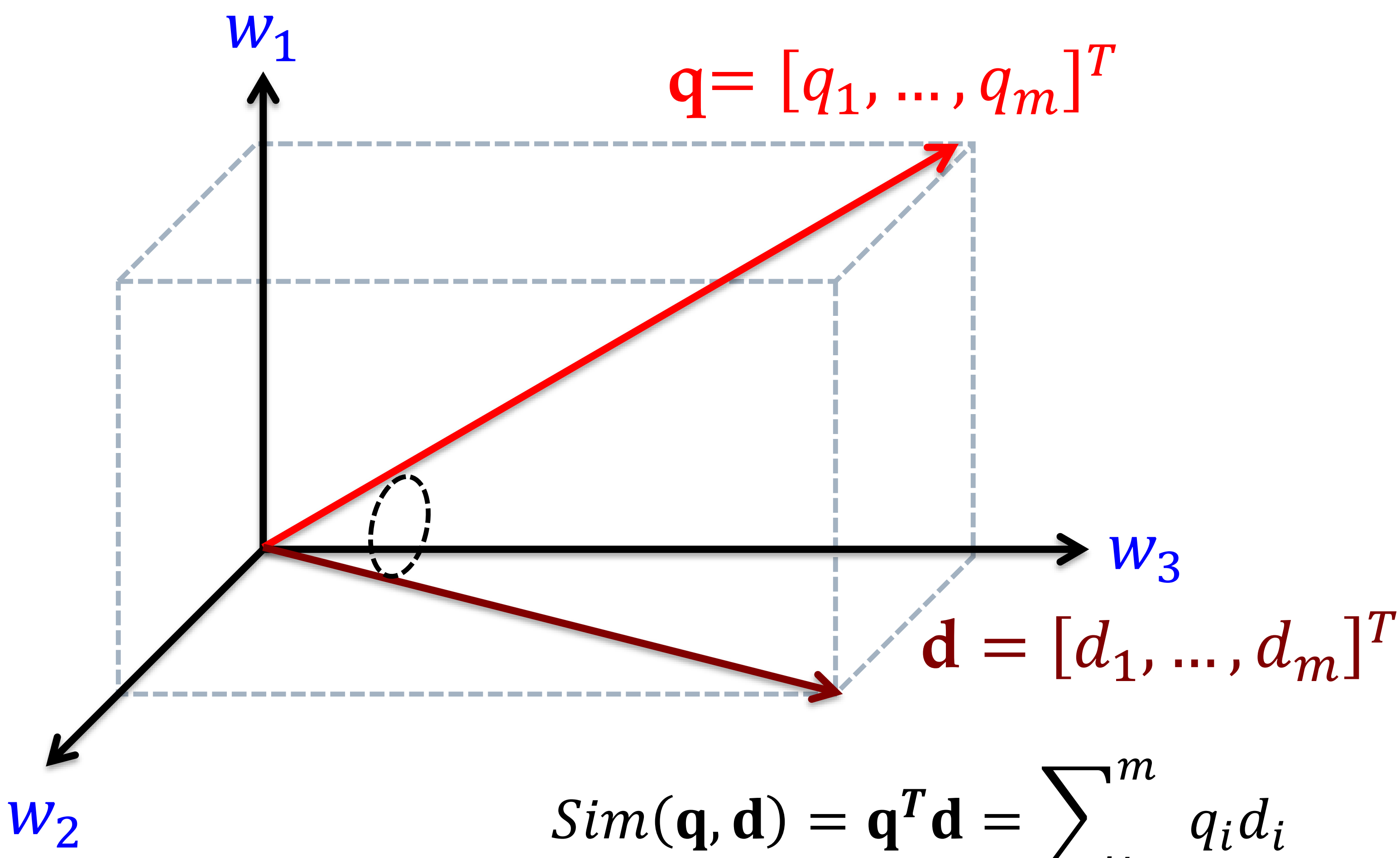
# Vector Placement: Bit Vector




$q_i, d_i \in \{0,1\}$

1 - $w_i$ is presented 0 - $w_i$ is absent

# Vector Placement: Bit Vector




$$Sim(\mathbf{q}, \mathbf{d}) = \mathbf{q}^T \mathbf{d} = \sum_{i=1}^{m} q_i d_i$$

# Simplest VSM

**BoW + Bit-Vector + Dot-Product**

$$\mathbf{q} = [q_1, \dots, q_m]^T$$

$$\mathbf{d} = [d_1, \dots, d_m]^T$$

$q_i, d_i \in \{0,1\}$

$1 - w_i$ *is present*; $0$ - $w_i$ *is absent*

$$Sim(\mathbf{q}, \mathbf{d}) = \mathbf{q}^T \mathbf{d} = \sum_{i=1}^{m} q_i d_i$$

**What does this ranking function intuitively capture?**
**Is this a good ranking function?**

# How Would You Rank These Documents?

**Query** = "news about presidential campaign"

Ideal Rank

| | | |
|---|---|---|
| **d1** | … news about … | **d4 +** |
| **d2** | … news about organic food campaign… | **d3 +** |
| **d3** | … news of presidential campaign … | **d1 0** |
| **d4** | … news of presidential campaign …<br>… presidential candidate … | **d2 -** |

# Ranking Using the Simplest VSM

Query = "news about presidential campaign"

d3 | … news of presidential campaign …

V= {news, about, presidential, campaign, food …. }

$$q=(\boxed{1,}\quad 1,\quad \boxed{1,}\quad \boxed{1,}\quad 0,\quad \ldots)$$

$$d3=(\boxed{1,}\quad 0,\quad \boxed{1,}\quad \boxed{1,}\quad 0,\quad \ldots)$$

**f(q,d3)= 1*1+1*0 +1*1 +1*1+0*0+ … = 3**

\# of distinct query words **matched** in a document

# Is the Simplest VSM Effective?

**Query** = "news about presidential campaign" Score

| | | |
|---|---|---|
| **d1** | … news about … | f(q,d1)=2 |
| **d2** | … news about organic food campaign… | f(q,d2)=3 |
| **d3** | … news of presidential campaign … | f(q,d3)=3 |
| **d4** | … news of presidential campaign …<br>… presidential candidate … | f(q,d4)=3 |

# Speculation

- How to achieve $f(q,d4) > f(q,d3) > f(q,d2)$

| | | |
|---|---|---|
| **d2** | … news about organic food campaign… | $f(q,d2)=3$ |
| **d3** | … news of presidential campaign … | $f(q,d3)=3$ |
| **d4** | … news of presidential campaign …<br>… presidential candidate … | $f(q,d4)=3$ |

1. Matching "presidential" more times deserves more credit (Comparing d3 & d4)
2. Matching "presidential" is more important than matching "about" (Comparing d2 & d3)

# Improved VSM with TF Weighting

- $\mathbf{q} = [q_1, \ldots, q_m]^T$ — $q_i$: count of $w_i$ in query
- $\mathbf{d} = [d_1, \ldots, d_m]^T$ — $d_i$: count of $w_i$ in document

$$Sim(\mathbf{q}, \mathbf{d}) = \mathbf{q}^T \mathbf{d} = \sum_{i=1}^{m} q_i d_i$$

- What does this ranking function intuitively capture?
- Does it fix the problems of the simplest VSM?

Query = "news about presidential campaign"

d2 | ... news about organic food campaign... | f(q,d2)=3

q= (1, 1, 1, 1, 0, ...)
d2= (1, 1, 0, 1, 1, ...)

d3 | ... news of presidential campaign ... | f(q,d3)=3

q= (1, 1, 1, 1, 0, ...)
d3= (1, 0, 1, 1, 0, ...)

d4 | ... news of presidential campaign ... ... presidential candidate ... | **f(q,d4)=4**

q= (1, 1, 1, 1, 0, ...)
d4= (1, 0, 2, 1, 0, ...)

# How to Fix Problem 2 ("presidential" vs. "about")

**d2** … news about organic food campaign…

**d3** … news of presidential campaign …

V= {news, about, presidential, campaign, food .... }

q= (1, 1, 1, 1, 0, …)
d2= (1, 1, 0, 1, 1, …)

f(q,d2)<3

q= (1, 1, 1, 1, 0, …)
d3= (1, 0, 1, 1, 0, …)

f(q,d3)>3

# Further Improved VSM with TF-IDF

$\mathbf{q} = [q_1, \dots, q_m]^T$

$\mathbf{d} = [d_1, \dots, d_m]^T$

- $q_i$: count of $w_i$ in query
- $d_i$: $c(w_i, \mathbf{d}) * \boldsymbol{IDF(w_i)}$

$$Sim(\mathbf{q}, \mathbf{d}) = \mathbf{q}^T \mathbf{d} = \sum_{i=1}^{m} q_i d_i$$

- **Adding Inverse Document Frequency (IDF)**
- Does it fix the problems of the simplest VSM?

# IDF Weighting: Penalizing Popular Terms

**IDF: M/k**

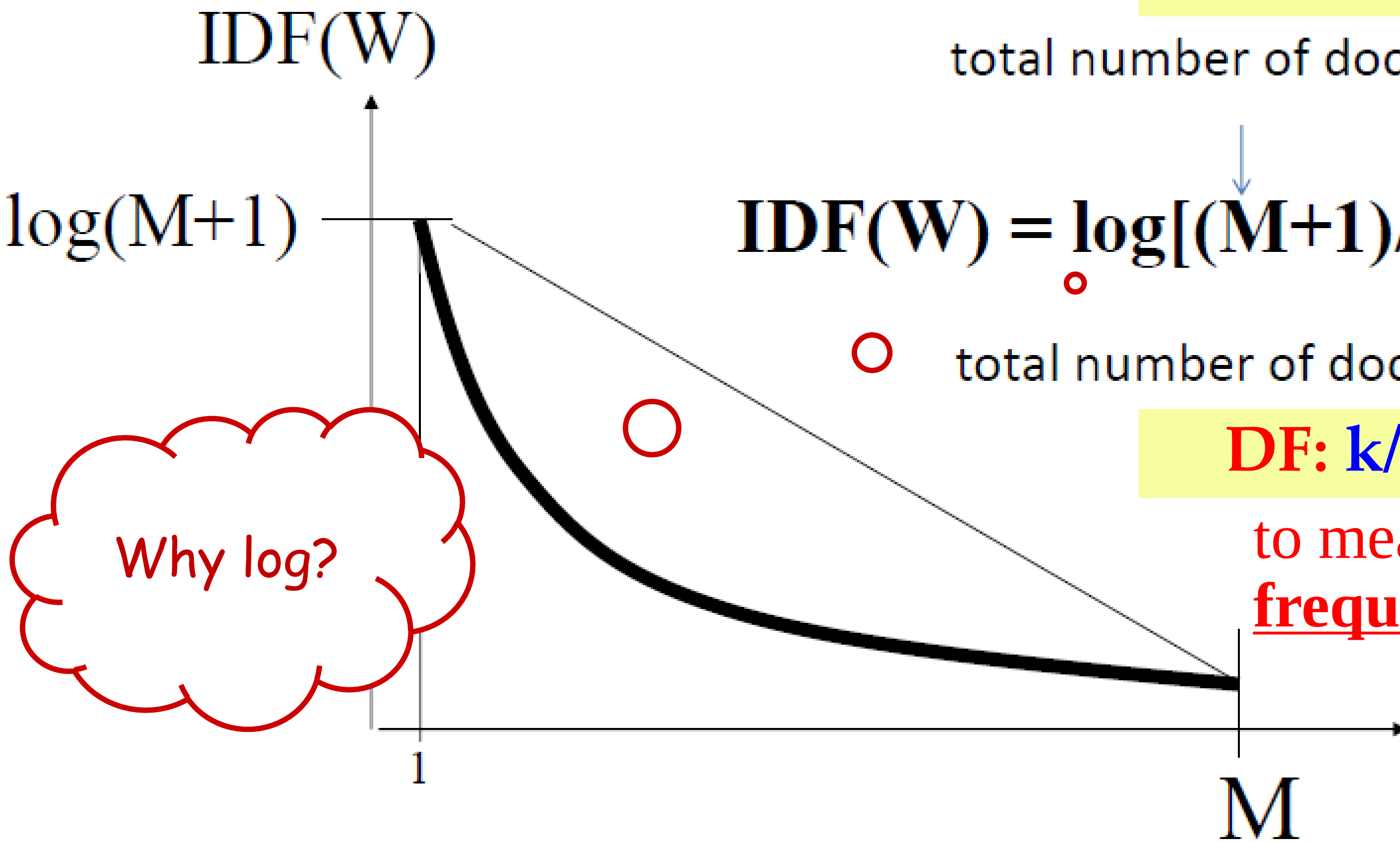
total number of docs in collection

$$IDF(W) = \log[(M+1)/k]$$

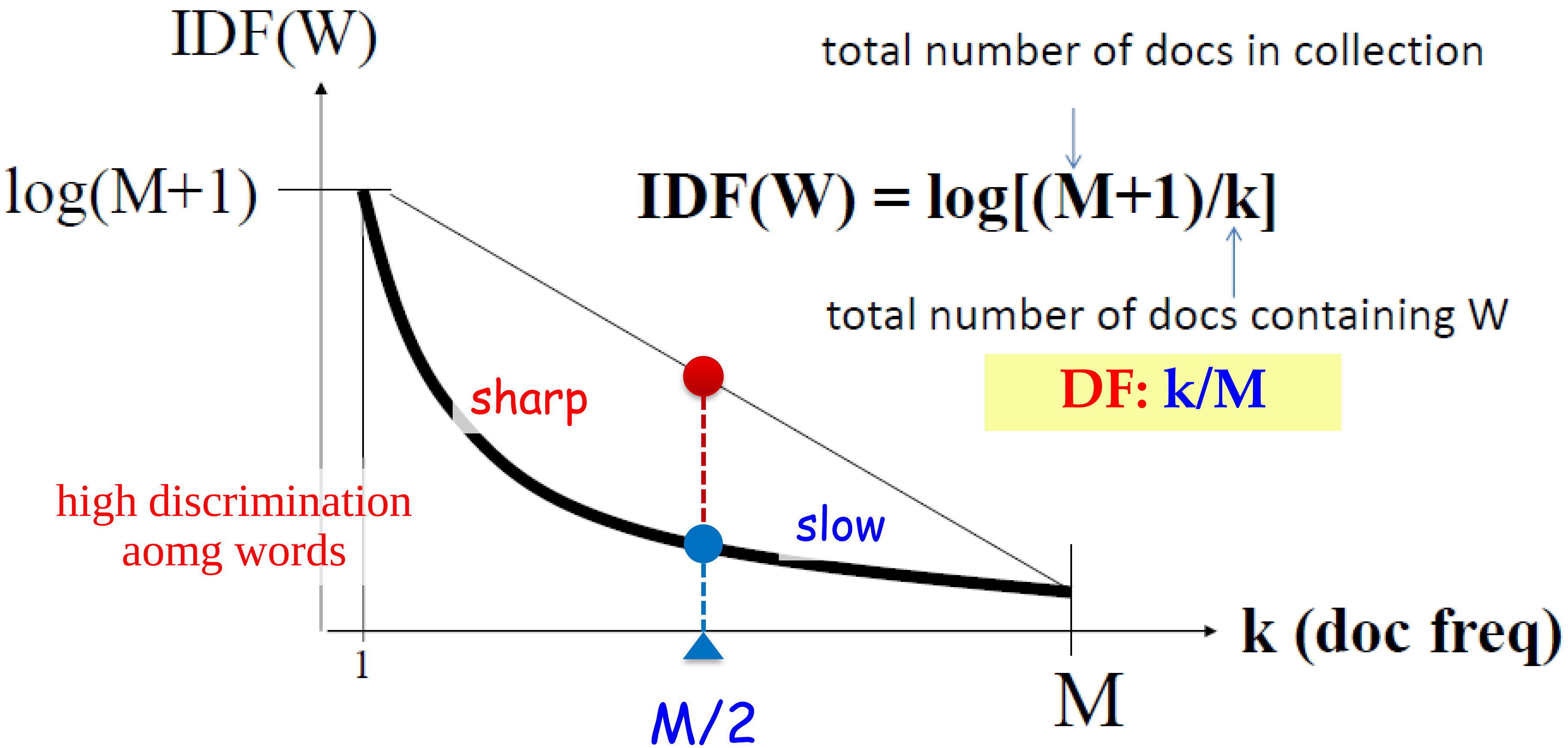
total number of docs containing W

**DF: k/M**

to measure the **use frequency** of a word



# IDF Weighting: Penalizing Popular Terms



# Solving Problem 2 ("presidential" vs. "about")

**d2** … news about organic food campaign…

**d3** … news of presidential campaign …

| | | | | | | |
|---|---|---|---|---|---|---|
| V= | {news, | about, | presidential, | campaign, | food | …. } |
| **IDF(W)=** | **1.5** | **1.0** | **2.5** | **3.1** | **1.8** | |
| q= | (1, | 1, | 1, | 1, | 0, | …) |
| d2= | (1*1.5, | 1*1.0 | 0, | 1*3.1, | 0, | …) |
| q= | (1, | 1, | 1, | 1, | 0, | …) |
| d3= | (1*1.5, | 0, | 1*2.5 | 1*3.1, | 0, | …) |

**f(q,d2) = 5.6 < f(q,d3)=7.1**

# How Effective is VSM with TF-IDF?

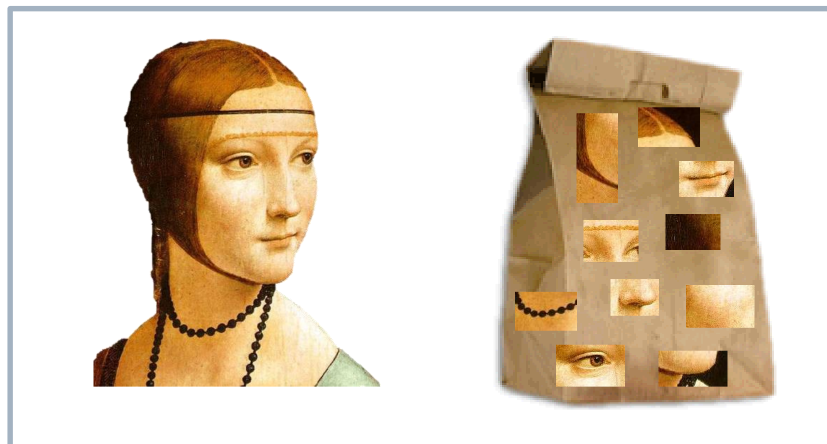
**Query** = "news about presidential campaign"

| | Document | Score |
|---|---|---|
| **d1** | … news about … | f(q,d1)=2.5 |
| **d2** | … news about organic food campaign… | f(q,d2)=5.6 |
| **d3** | … news of presidential campaign … | f(q,d3)=7.1 |
| **d4** | … news of presidential campaign …<br>… presidential candidate … | f(q,d4)=9.6 |

# Bag-of-VisualWord (Features)

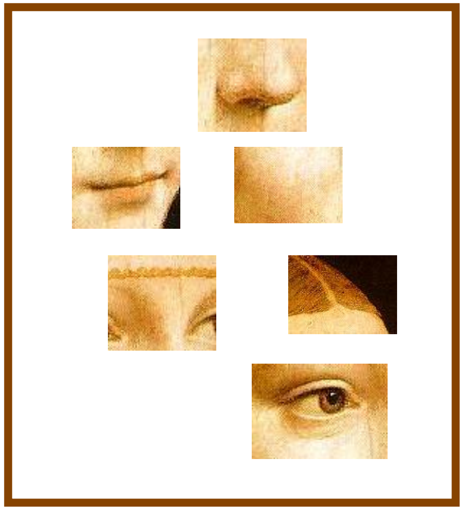

Bag-of-Word

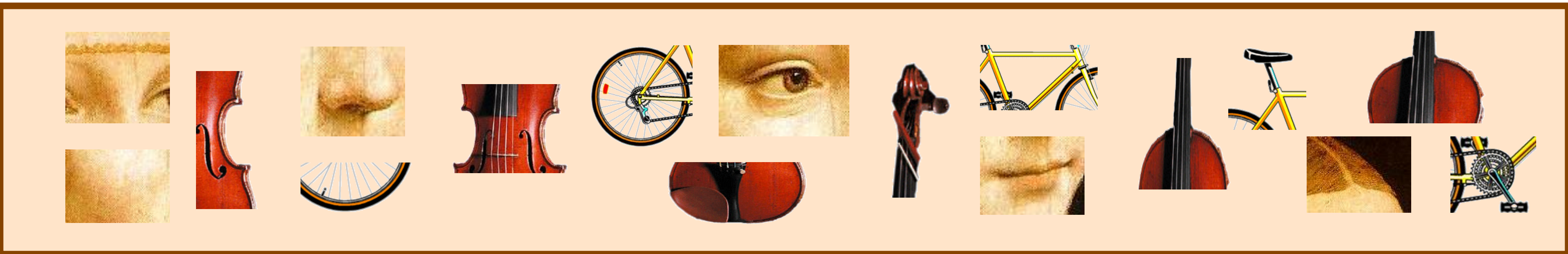
Bag-of-Visual-Word



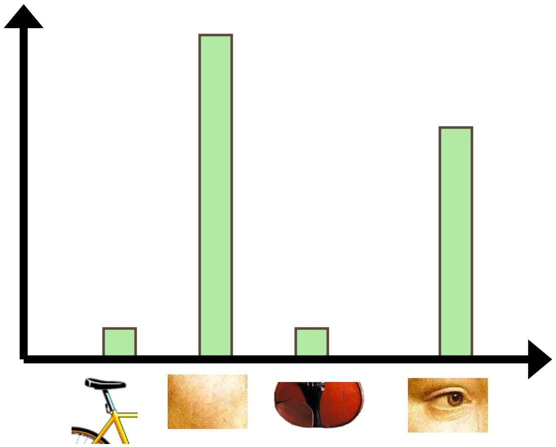
Step-1: Extract local features from each image

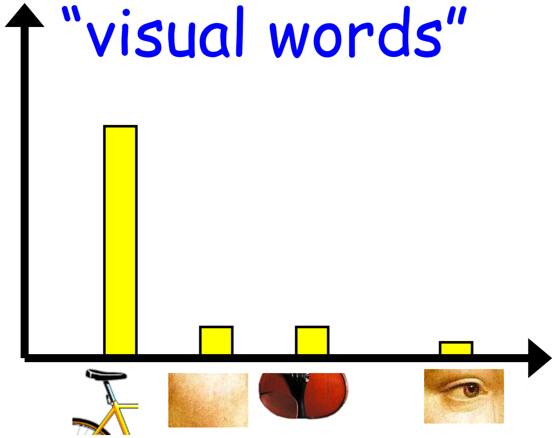
# Step-2: Learn "visual vocabulary" based on extracted local features

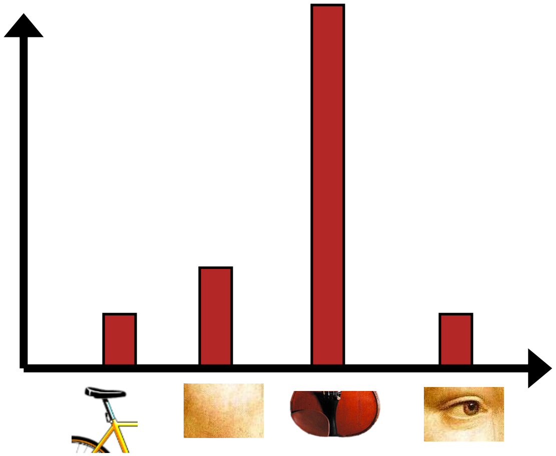


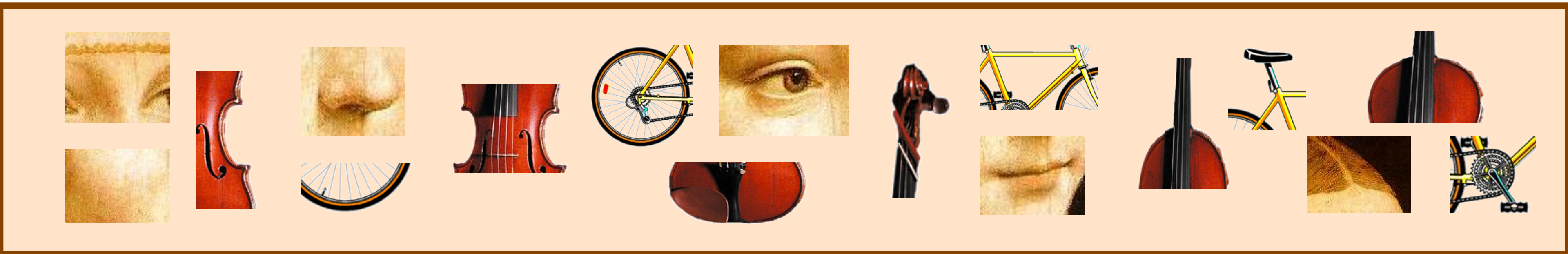



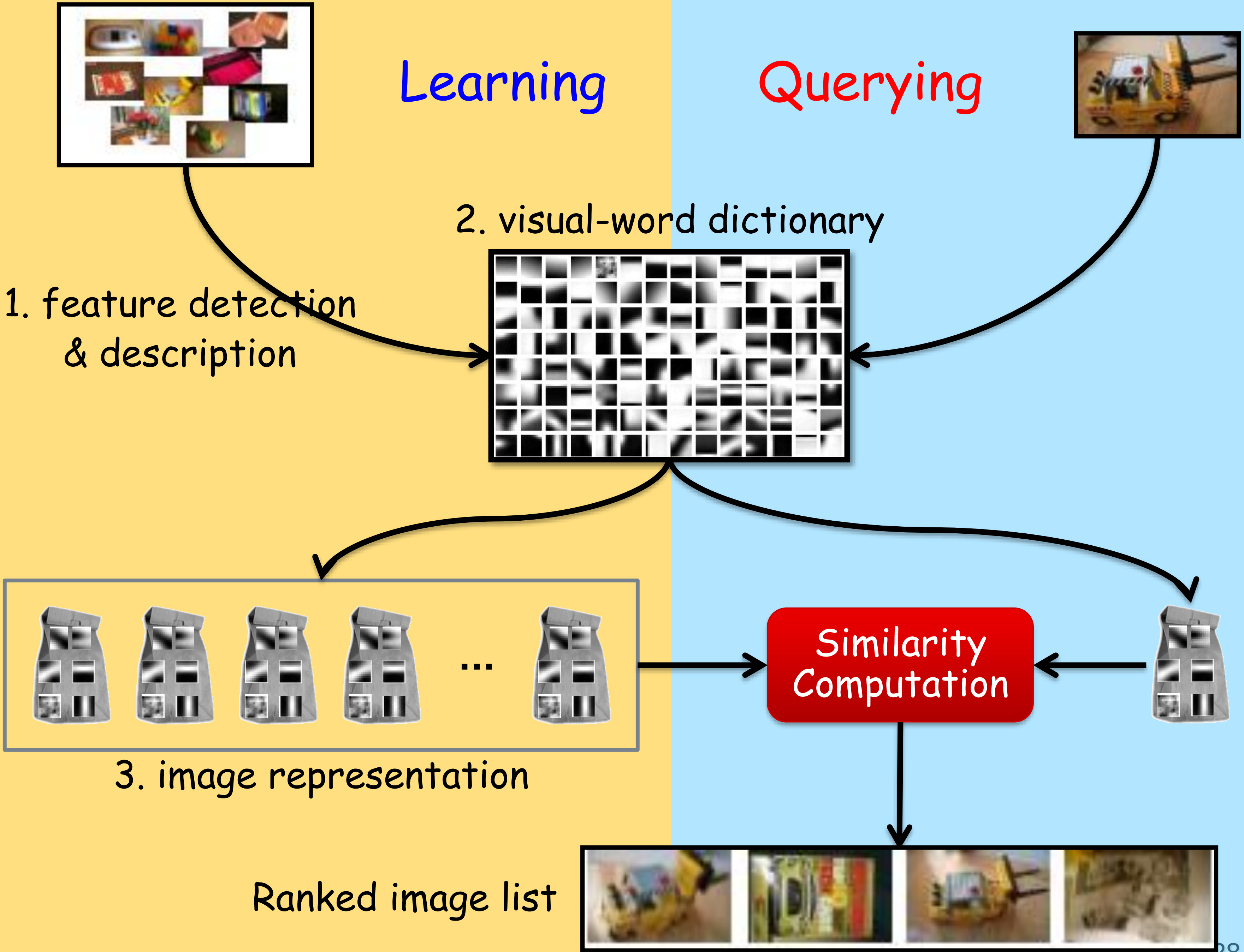
Step-3: Represent images by frequencies of "visual words"







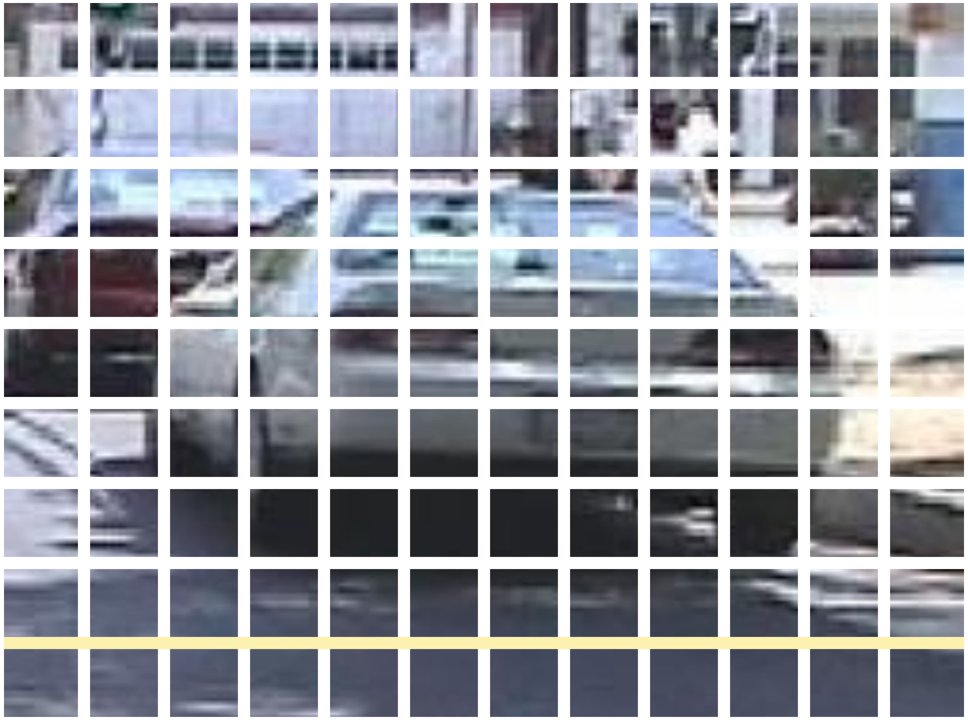




Learning
Querying
1. feature detection
& description
2. visual-word dictionary
3. image representation
...
Similarity Computation
Ranked image list

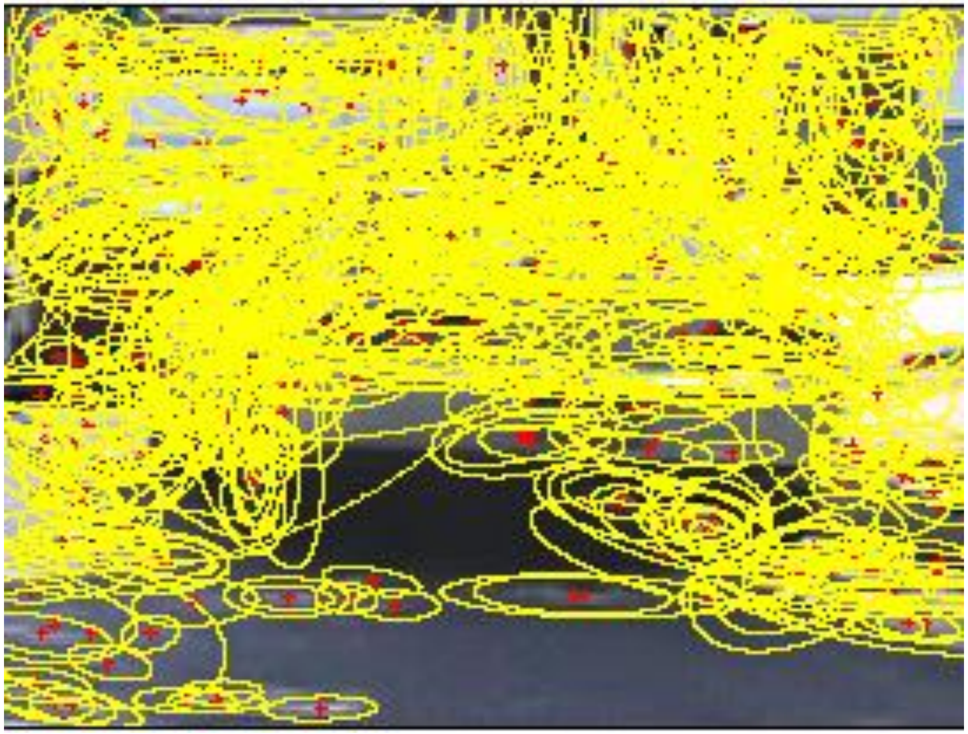
# Step-1: Feature extraction

- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005



# Step-1: Feature extraction

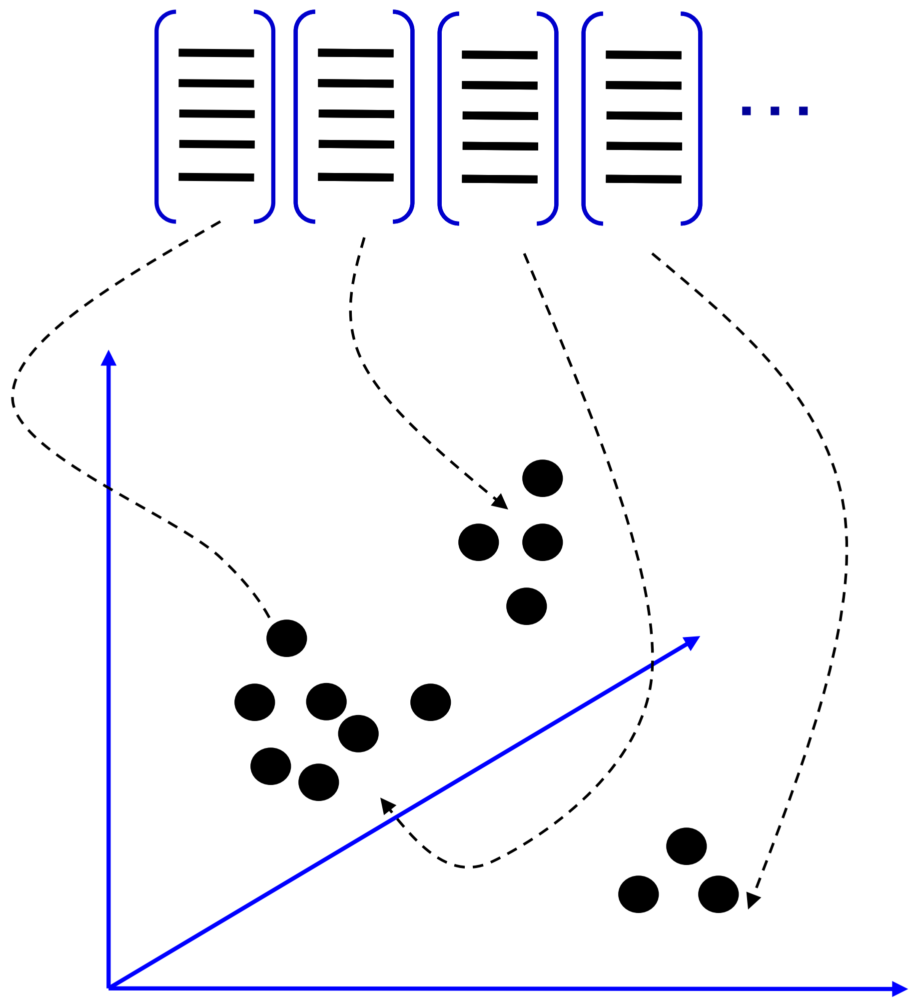
- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic et al. 2005



# Step-1: Feature extraction

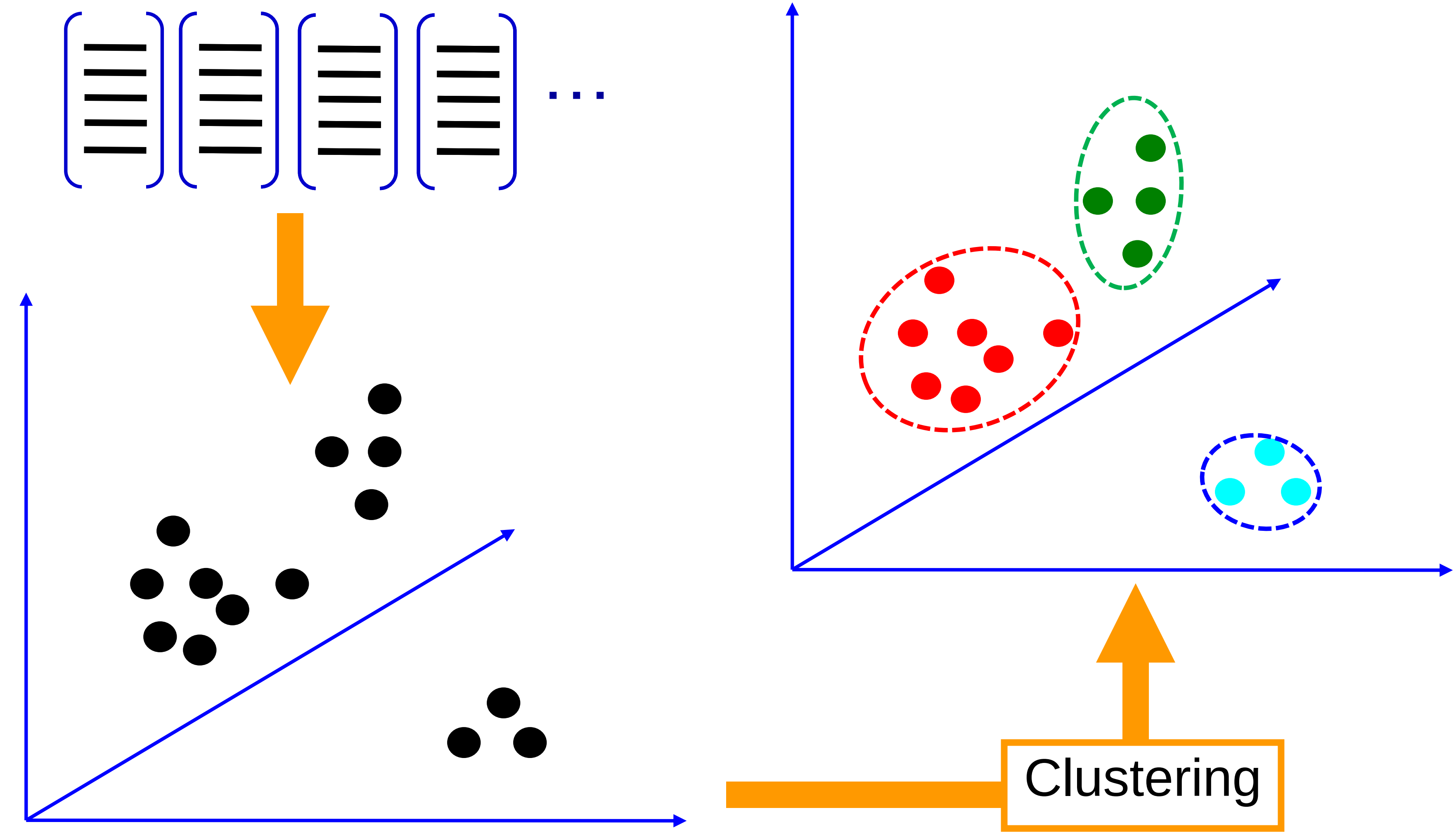
- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic et al. 2005
- Other methods
  - Random sampling (Vidal-Naquet & Ullman, 2002)
  - Segmentation-based patches (Barnard et al. 2003)

# Step-2: Learning the visual vocabulary

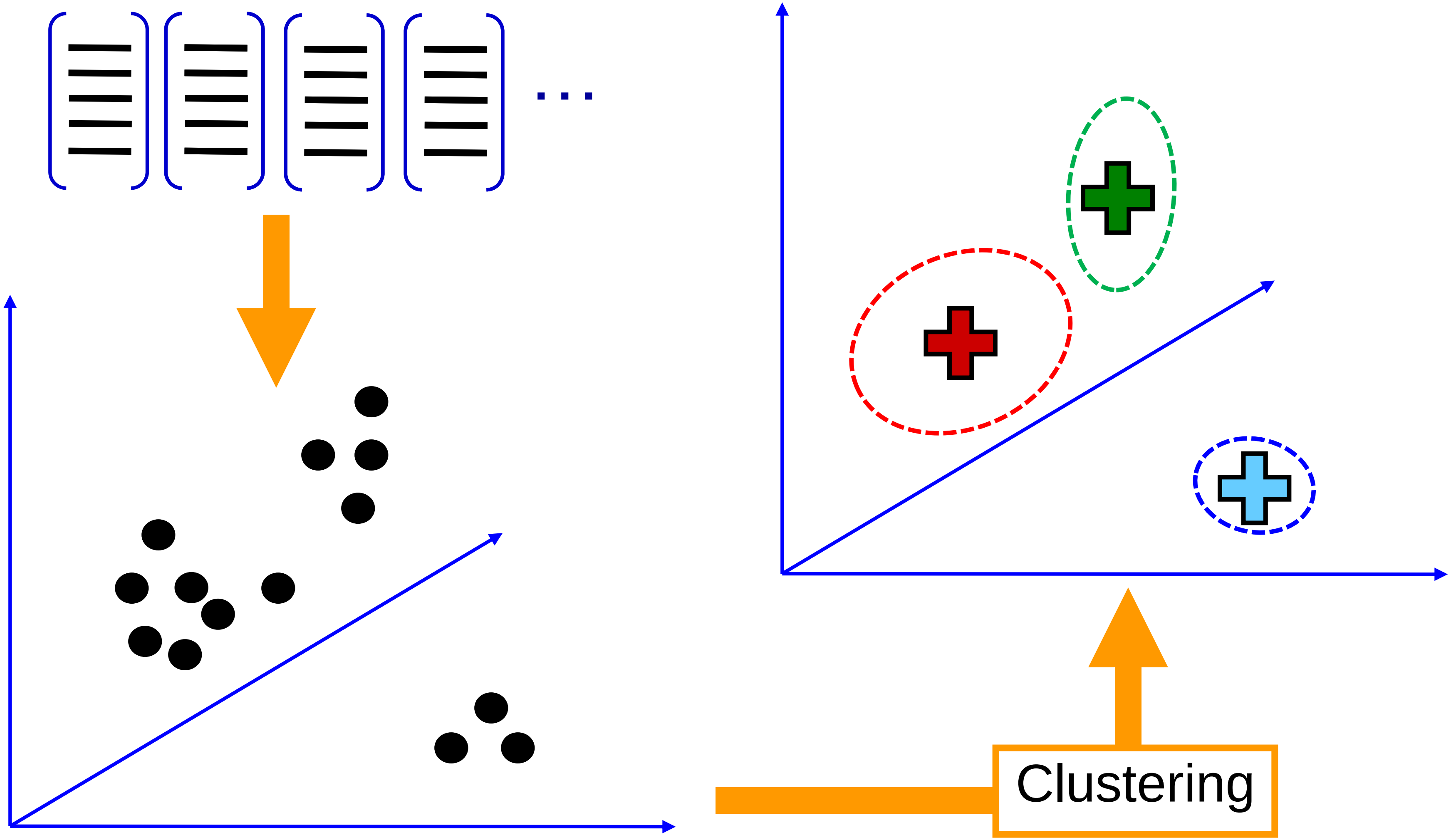



SIFT descriptor for each local image path

# Step-2: Learning the visual vocabulary



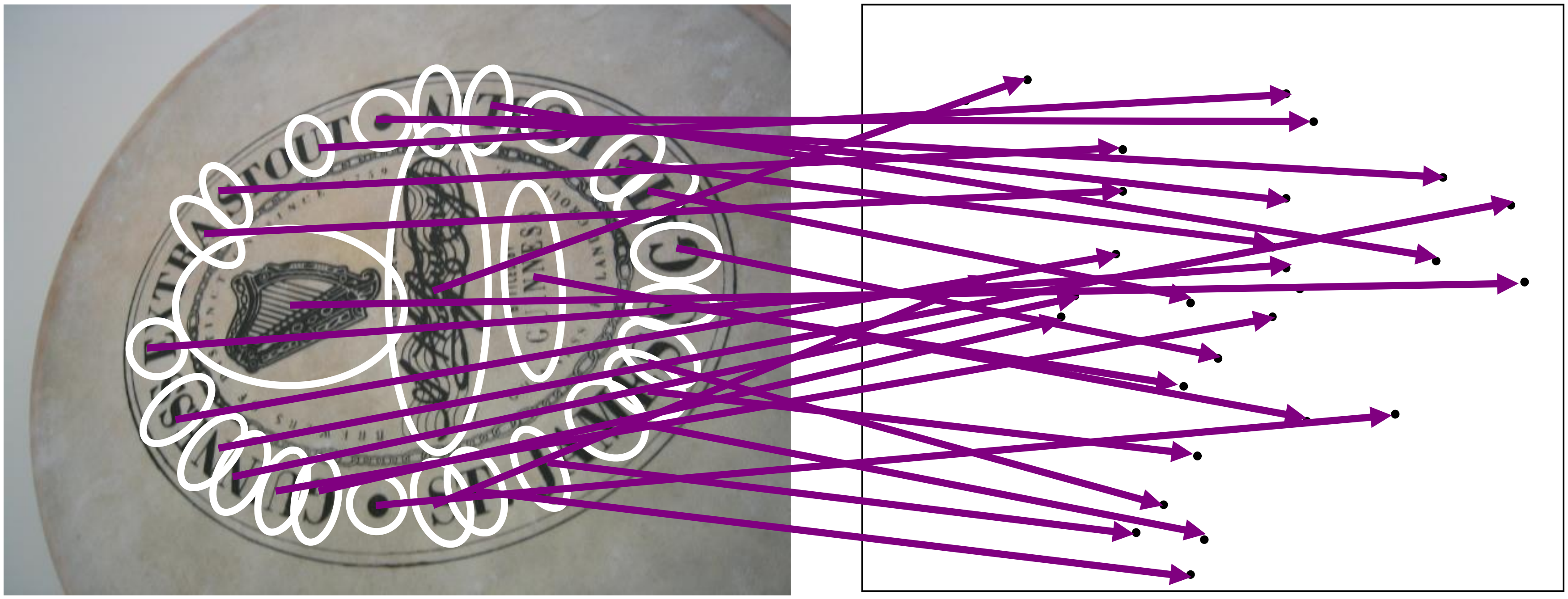
# Step-2: Learning the visual vocabulary



# From Clustering to Vector Quantization

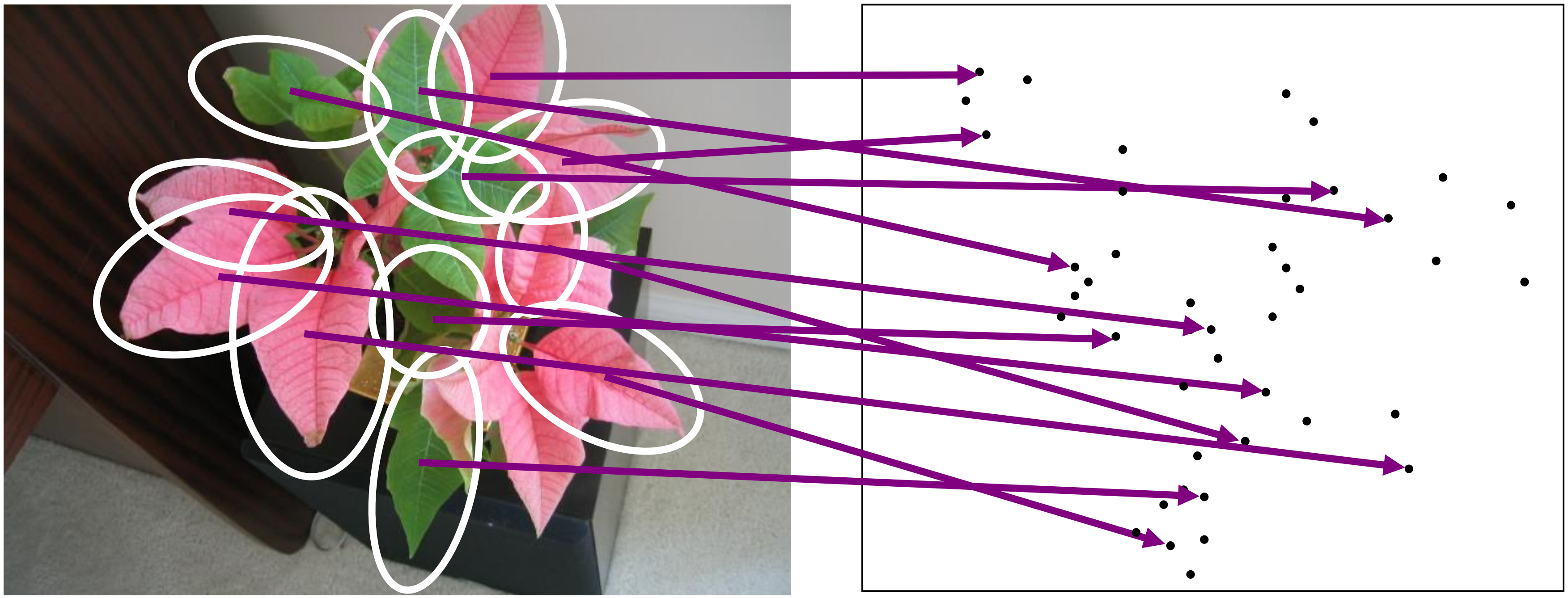
- **Clustering is a common method for learning a visual vocabulary or codebook**
  - Each cluster center produced by k-means becomes a codevector **(Quantization)**
  - Codebook can be learned on separate training set
  - Provided the training set is sufficiently representative, the codebook will be "universal"
  - **Codebook = visual vocabulary**
  - **Codevector = visual word**

# Extract local features from images

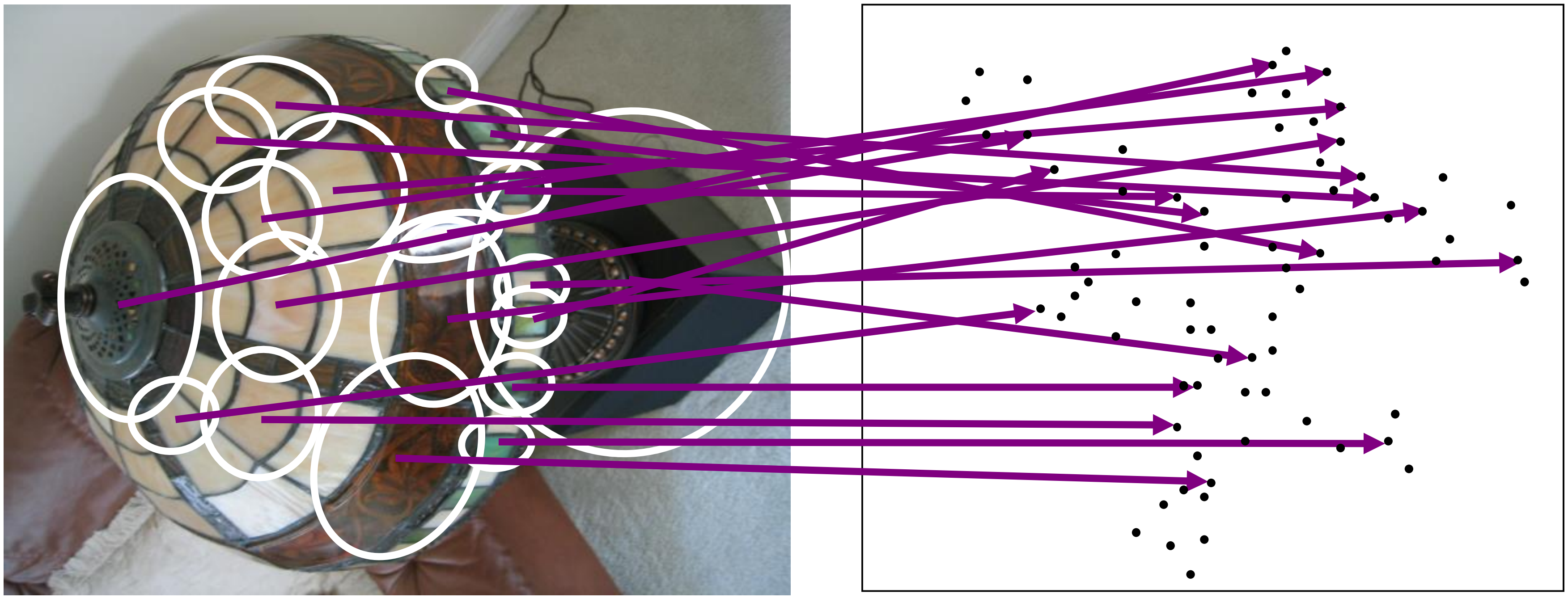


e.g., SIFT descriptor space:
each point is 128-dimensional

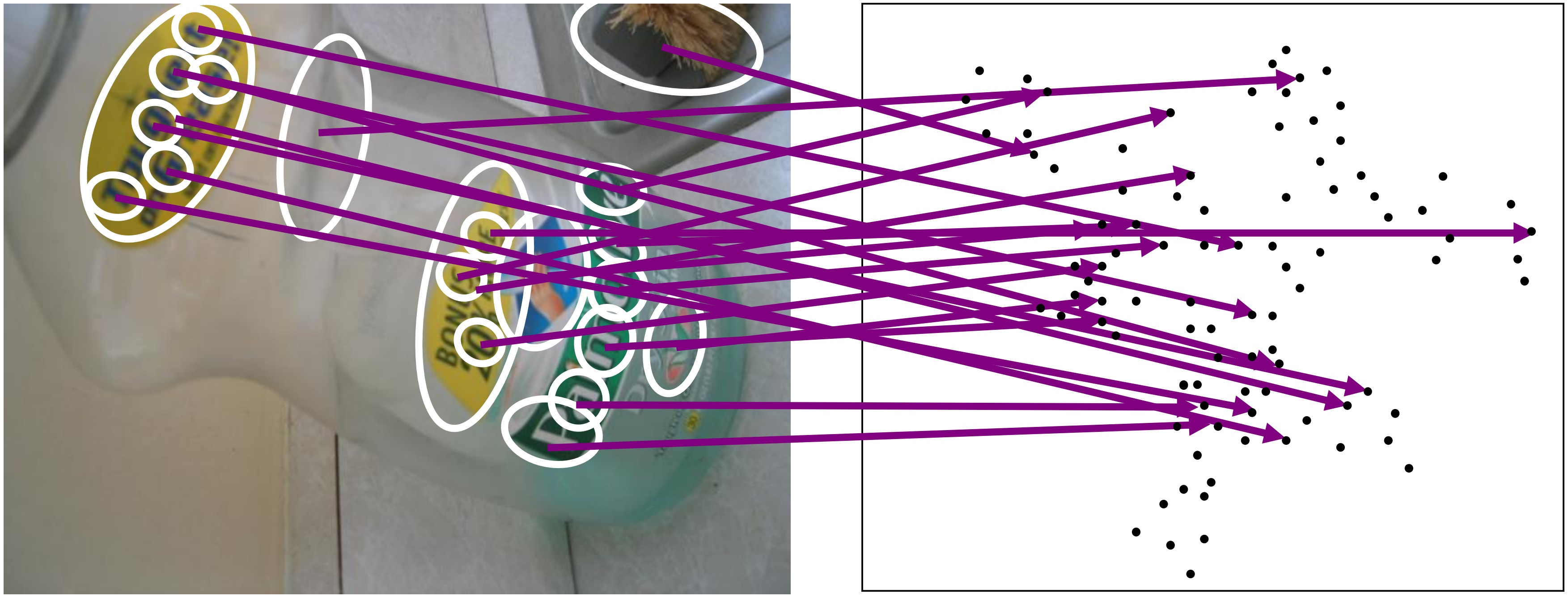
# Extract local features from images

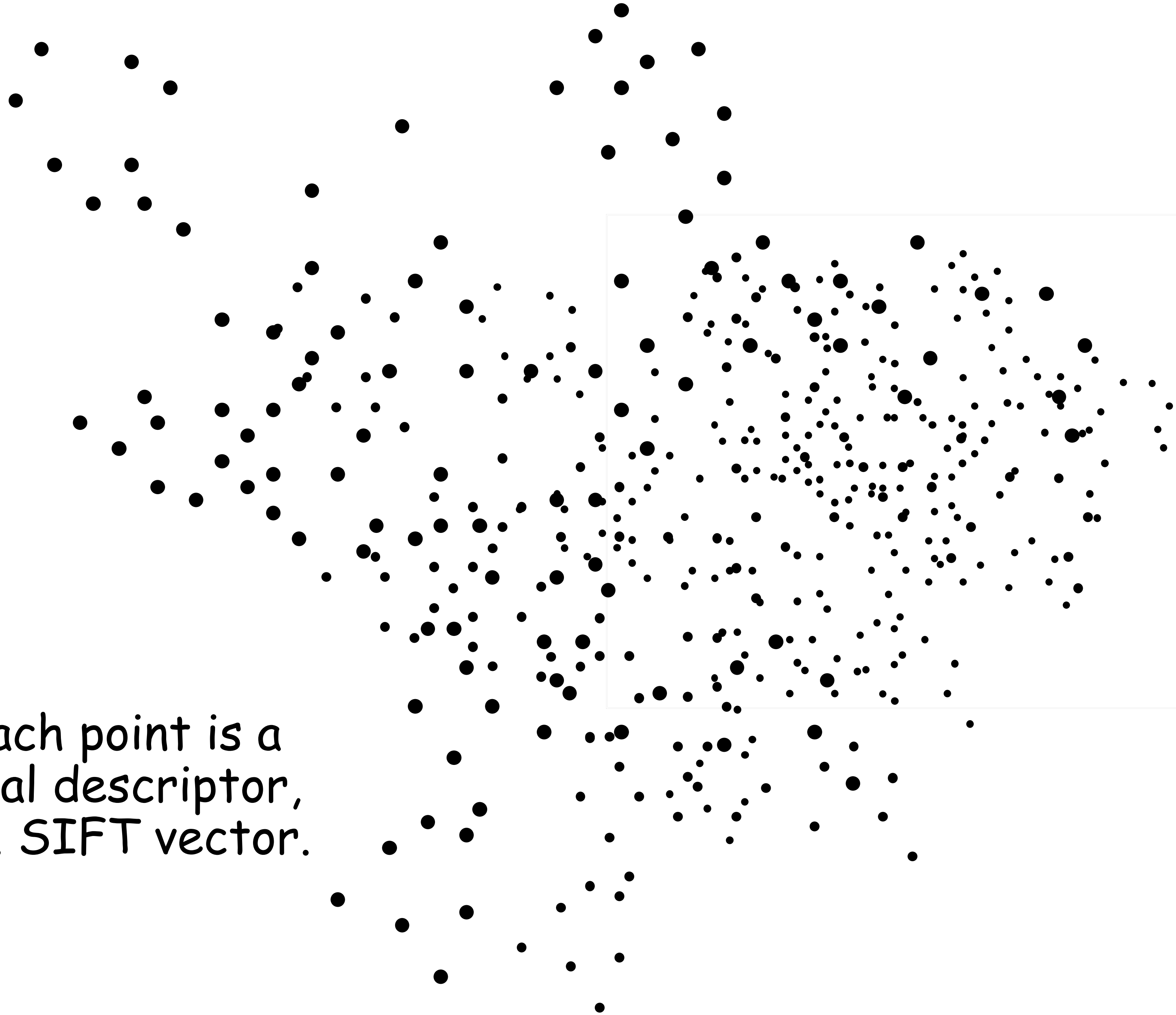


# Extract local features from images

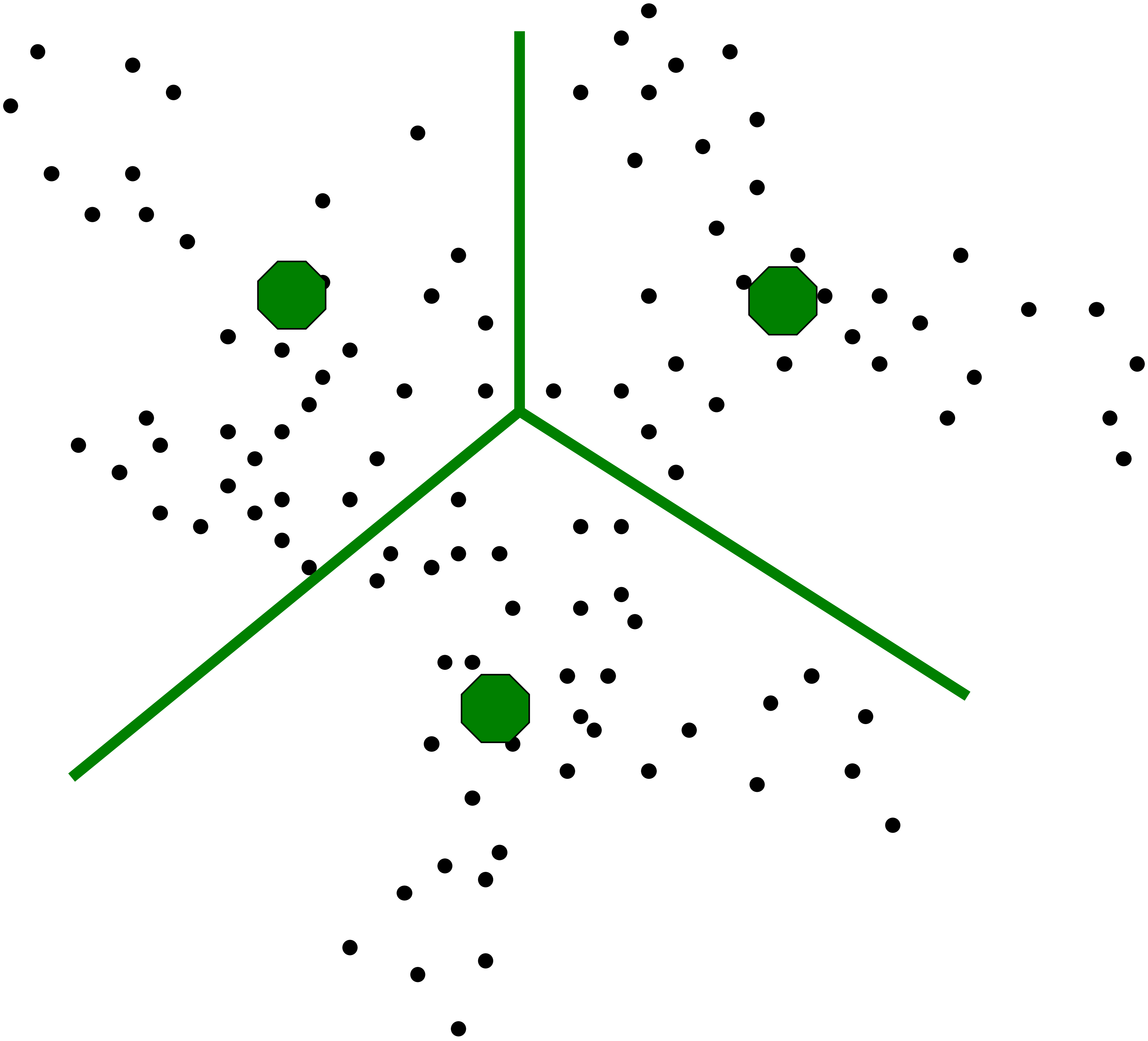


# Extract local features from images



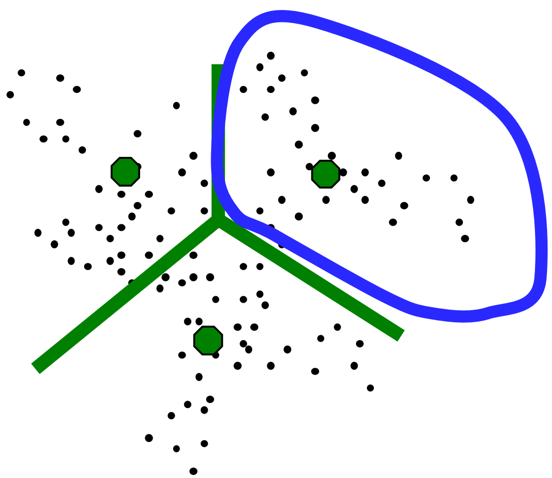


Each point is a local descriptor, e.g. SIFT vector.



# Visual words

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



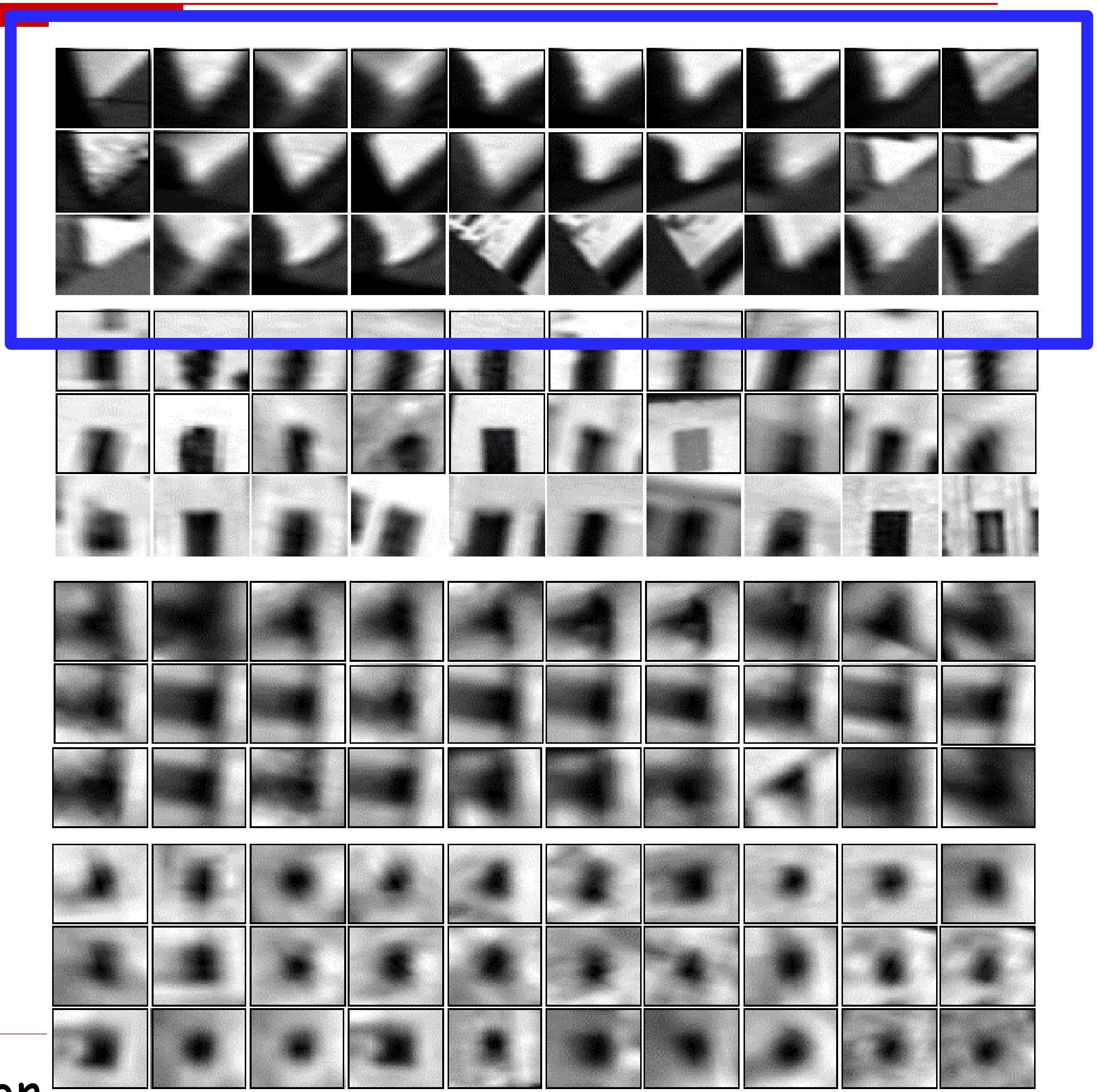
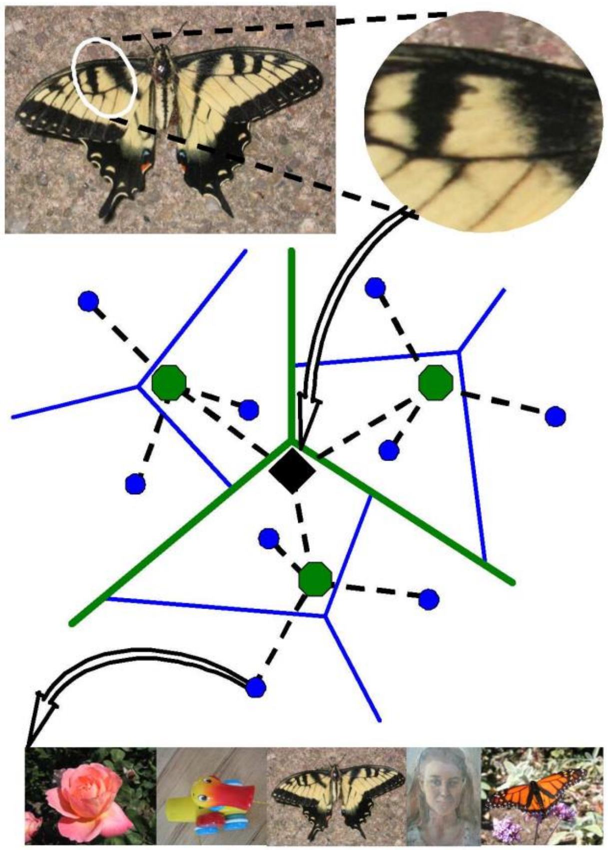


Figure from Sivic & Zisserman, ICCV 2003

Computer Vision Foundation

# Visual vocabularies: Issues

- How to choose vocabulary size?
  - Too small: visual words not representative of all patches
  - Too large: quantization artifacts, overfitting
- Computational efficiency
  - Vocabulary trees (Nister & Stewenius, 2006)



# Step-3: Image representation

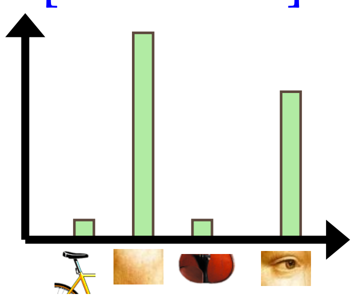
Align image paths with codewords


Computer Vision Foundation

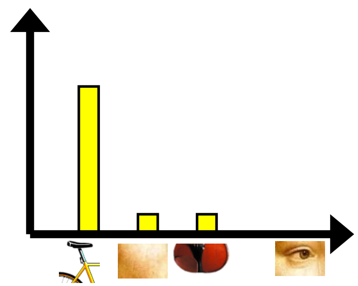
# Step-4: Comparing bags of words

- Rank images by normalized inner product between their (possibly weighted) occurrence counts---*nearest neighbor* search

[1 8 1 4]

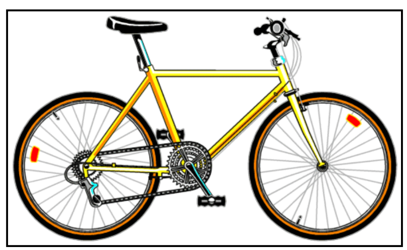


[5 1 1 0]



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

$\vec{d}_j$



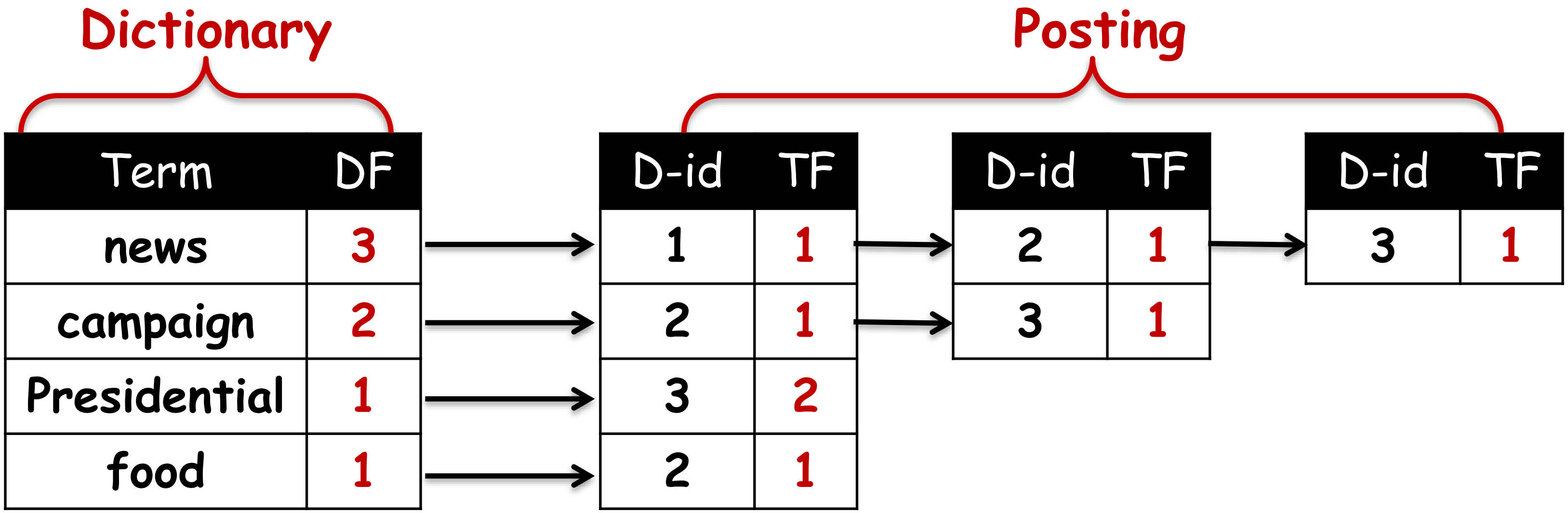
$\vec{q}$

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

for vocabulary of $V$ words

# Inverted indexing for faster BoW similarity computation

# An Example of Inverted Index

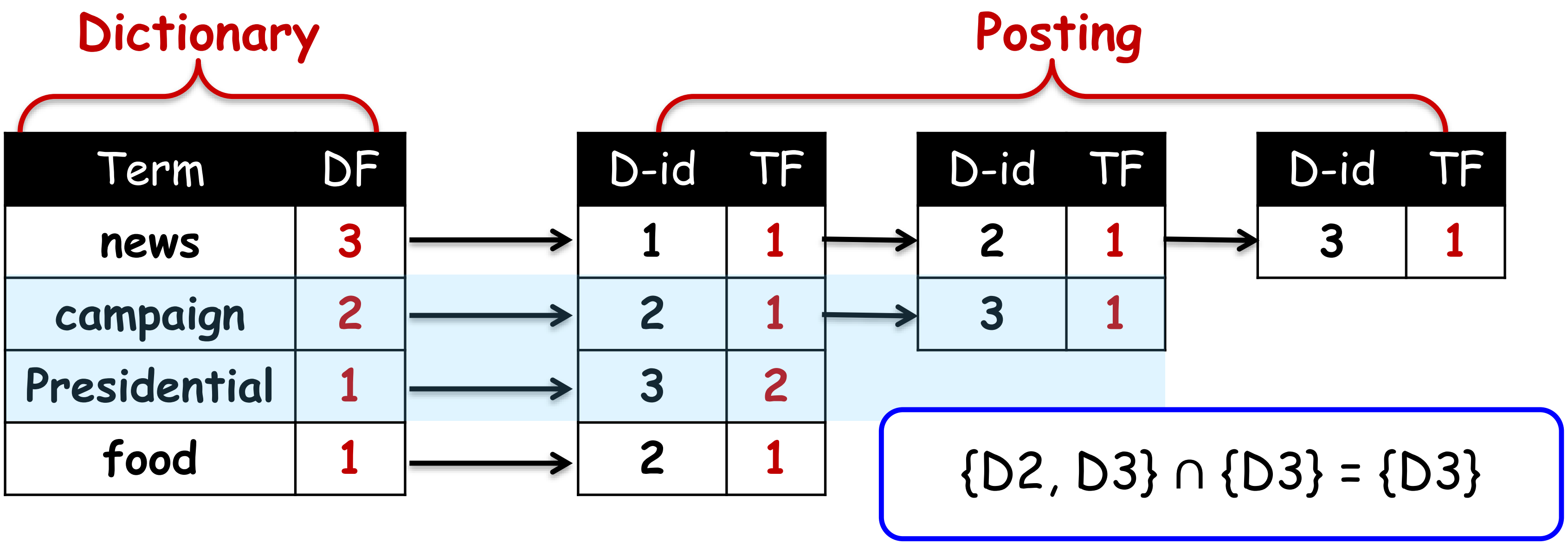



D1 … news about

D2 … news about organic food campaign…

D3 … news of presidential campaign … presidential candidate …

# An Example of Inverted Index




- Transform **Multi-term query** into **Boolean** operation
  - Match term "A" AND/OR term "B"
  - E.g. query="presidential campaign"

# Inverted indexing for BovW


New query image

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

…

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

How to score the retrieval results?

# Scoring retrieval quality

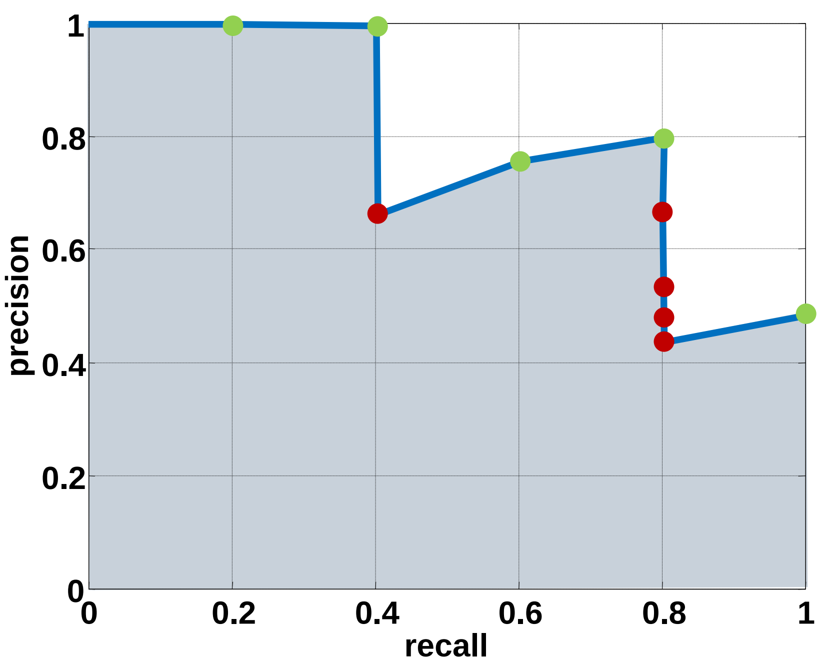
Query

Database size: 10 images

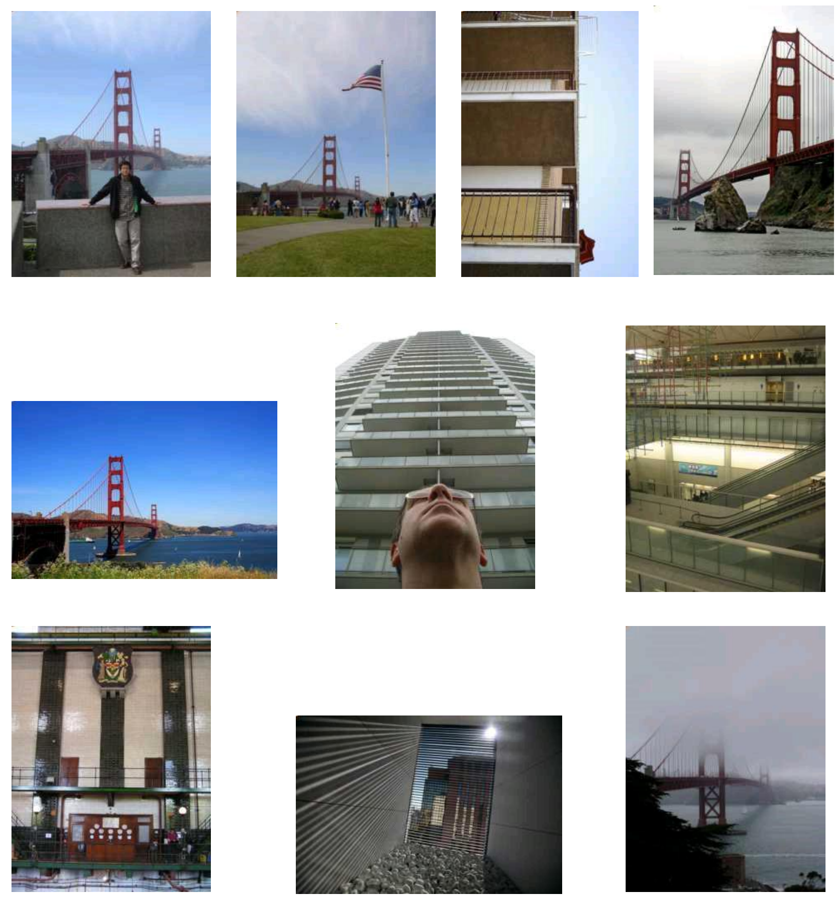
Relevant (total): 5 images

precision = #relevant / #returned

recall = #relevant / #total relevant




Results (ordered):



Slide credit: Ondrej Chum

# Summary on BoW Model

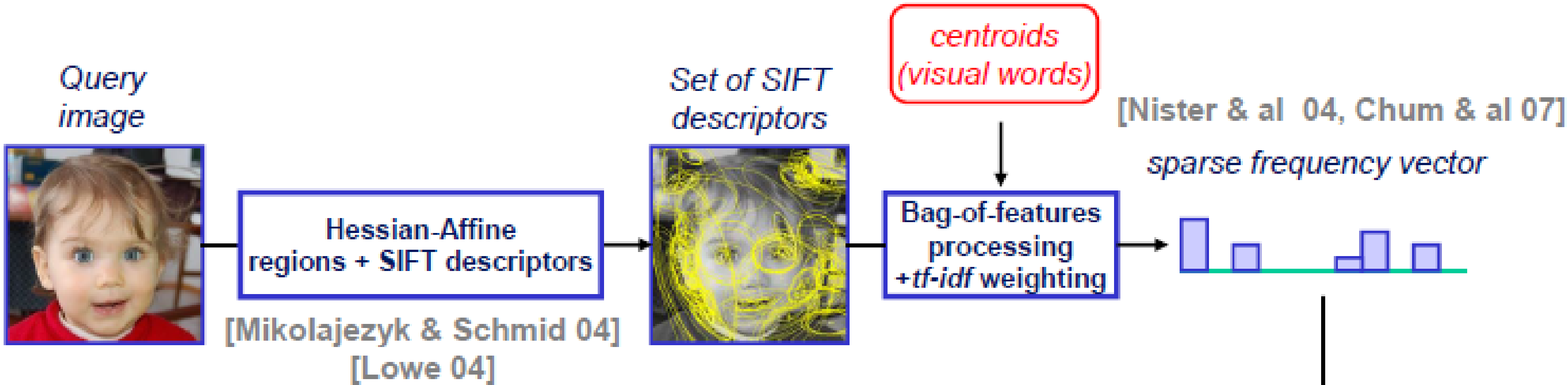
**Positives**

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

**Negatives**

- background and foreground mixed when bag covers whole image
- optimal vocabulary formation remains unclear

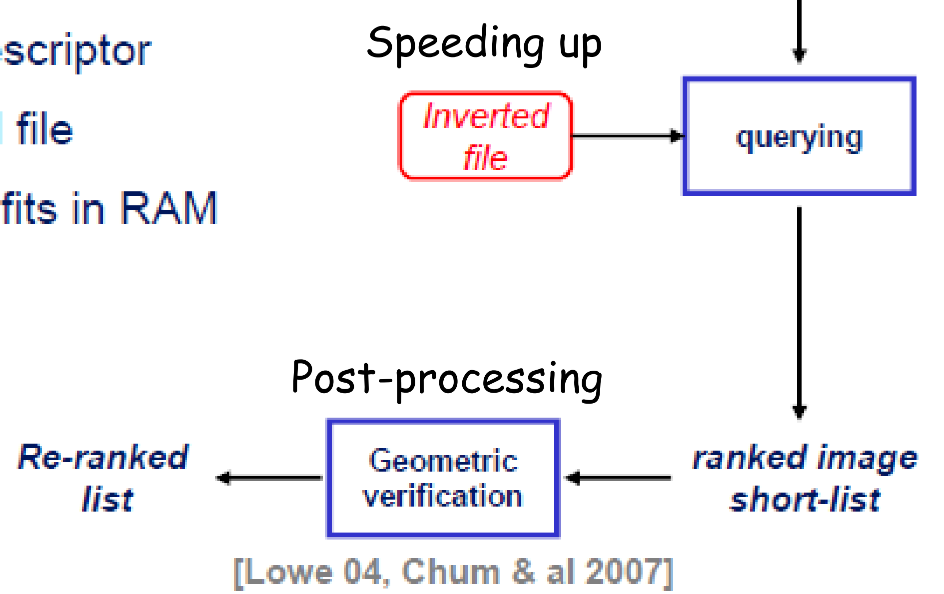
# State-of-the-Art BovW Flowchart for Search




Visual Words

- 1 word (index) per local descriptor
- only images ids in inverted file

⇒ 8 GB for a million images, fits in RAM



# Search on 10.2 M Image Database

Demo of Similar Image Search