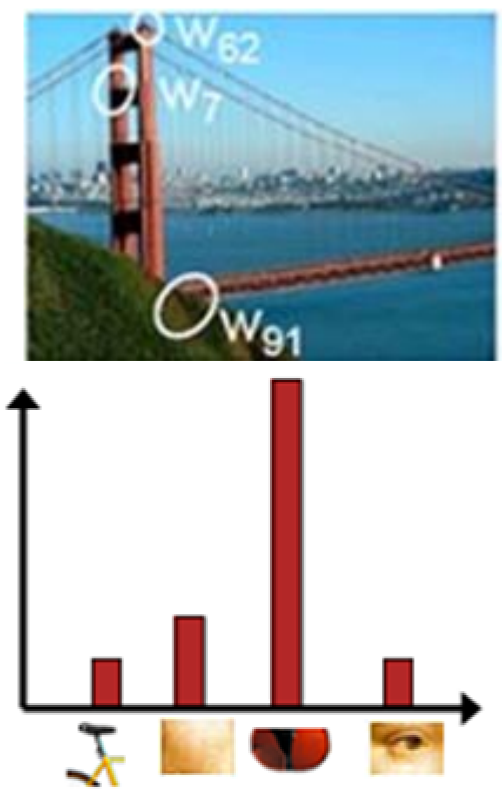
TDS3651
Visual Information Processing



Visual Words: Feature Indexing
Lecture 10

Faculty of Computing and Informatics
Multimedia University
created by Lai-Kuan, Wong
modified by Yuen Peng, Loh

Some graphics and materials are taken from UTexas at Austin CS376 & CS381V notes, Stanford Uni's CS131 notes and Gonzalez & Wood's DIP textbook

Lecture Outline

- Feature indexing – Why do it?
- “Visual words” concept
  - Bag of visual words
  - Inverted file index
  - Retrieval scoring
- Application for image retrieval

Matching local features

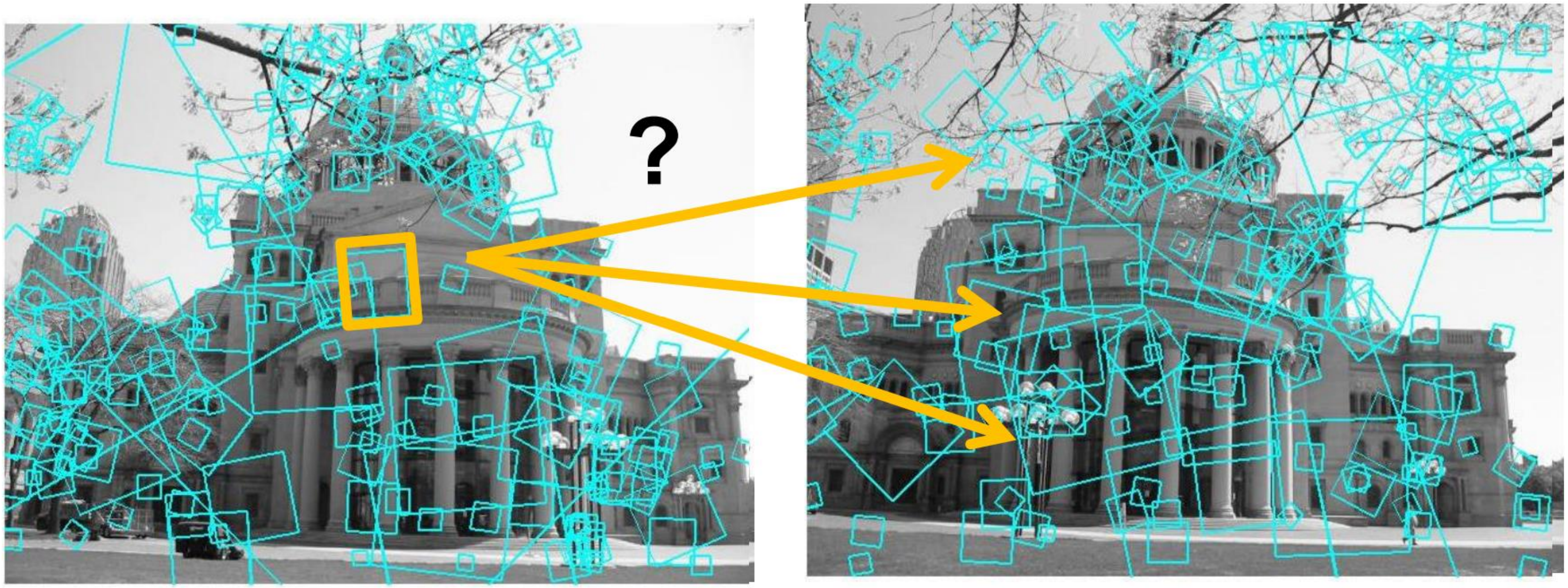
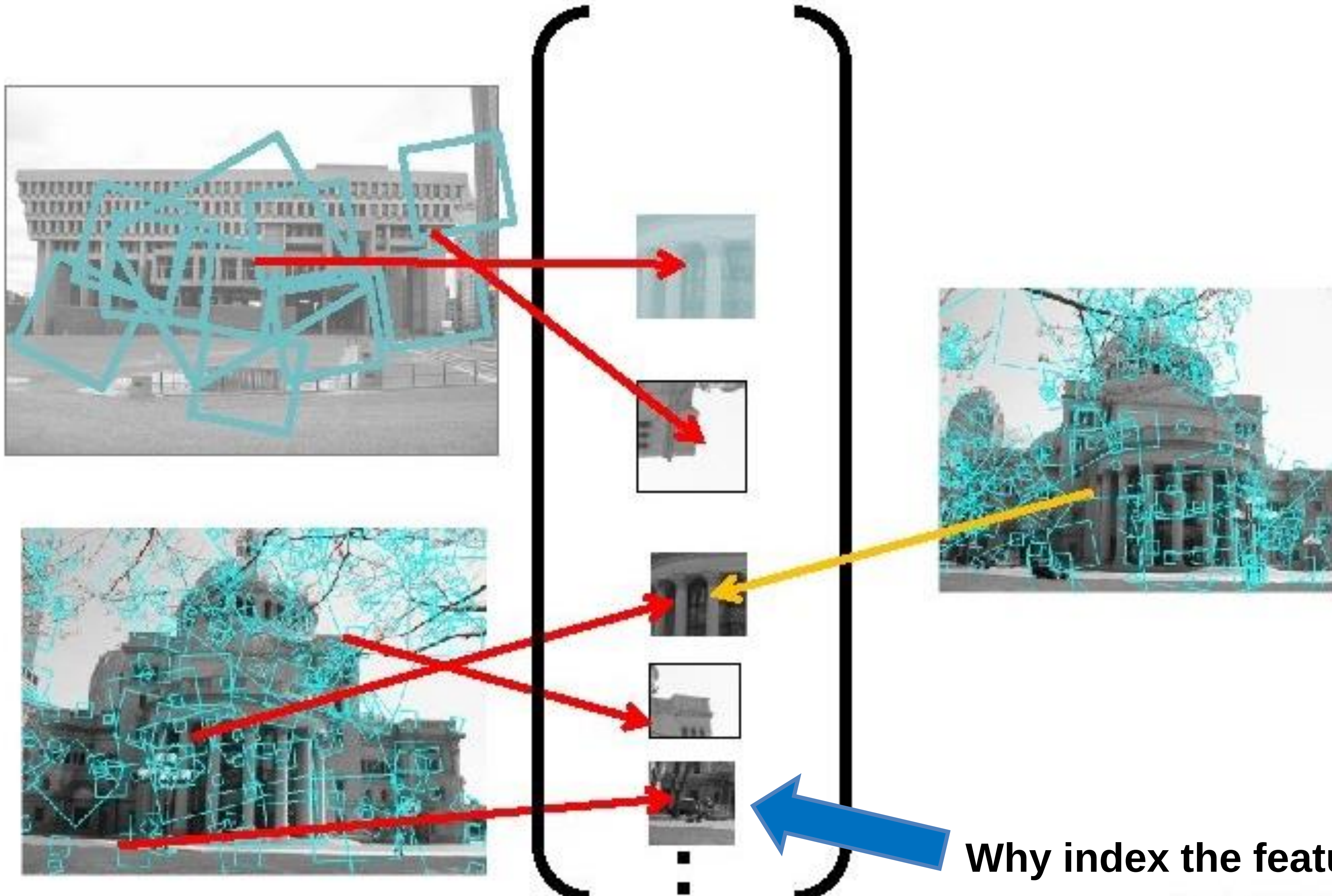


Image 1 Image 2

- To generate candidate matches, find patches that have the most similar appearance (e.g. lowest SSD)
- Simplest approach: Compare ALL, take the closest (or closest *k*, or within a threshold distance)

Indexing local features

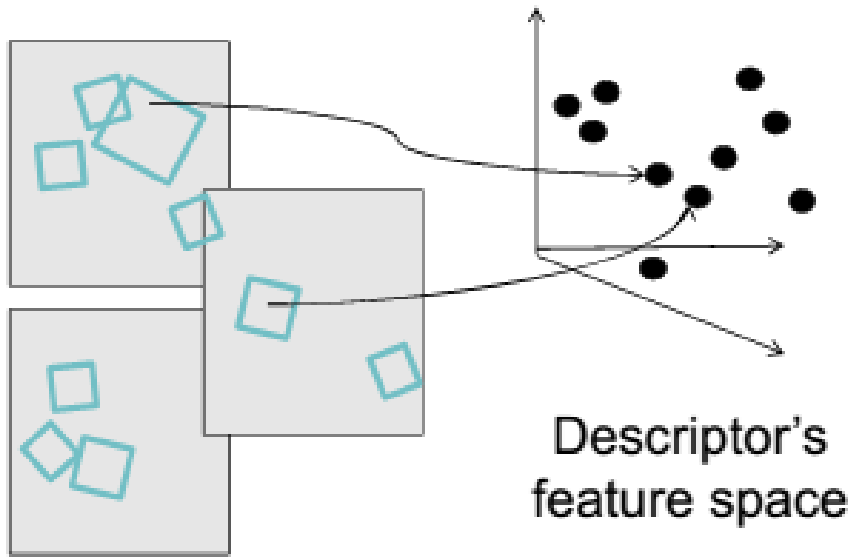


Why index the features?

Kristen Grauman

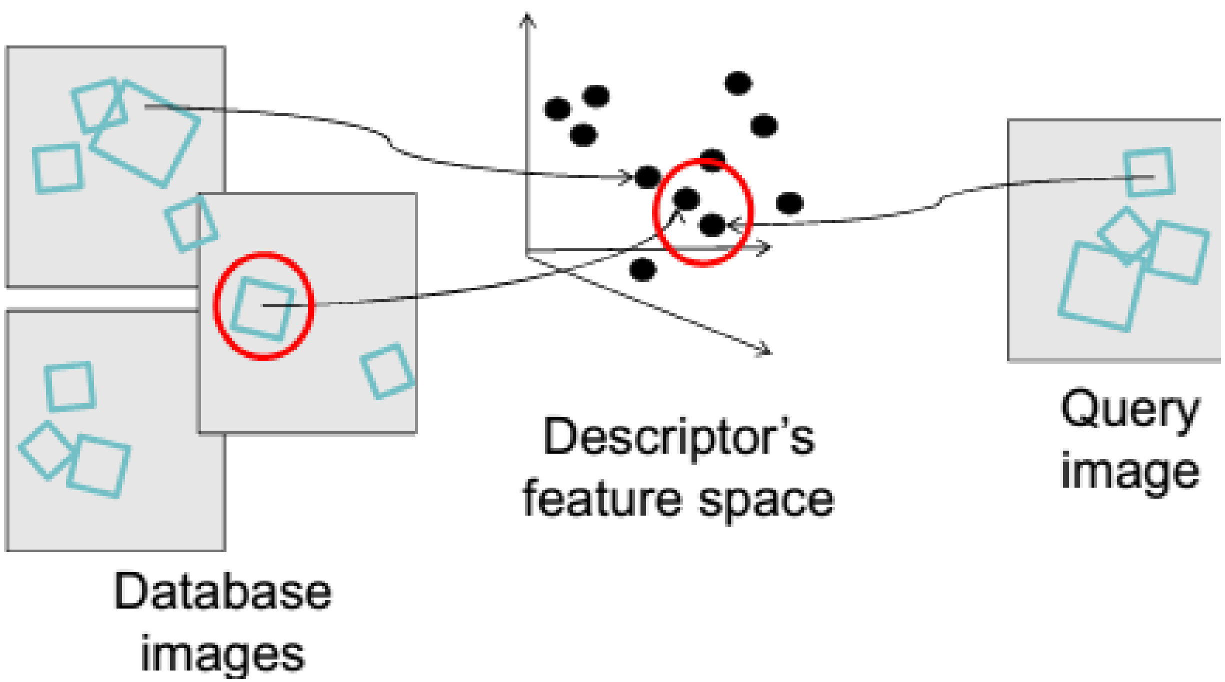
Indexing Local Features

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



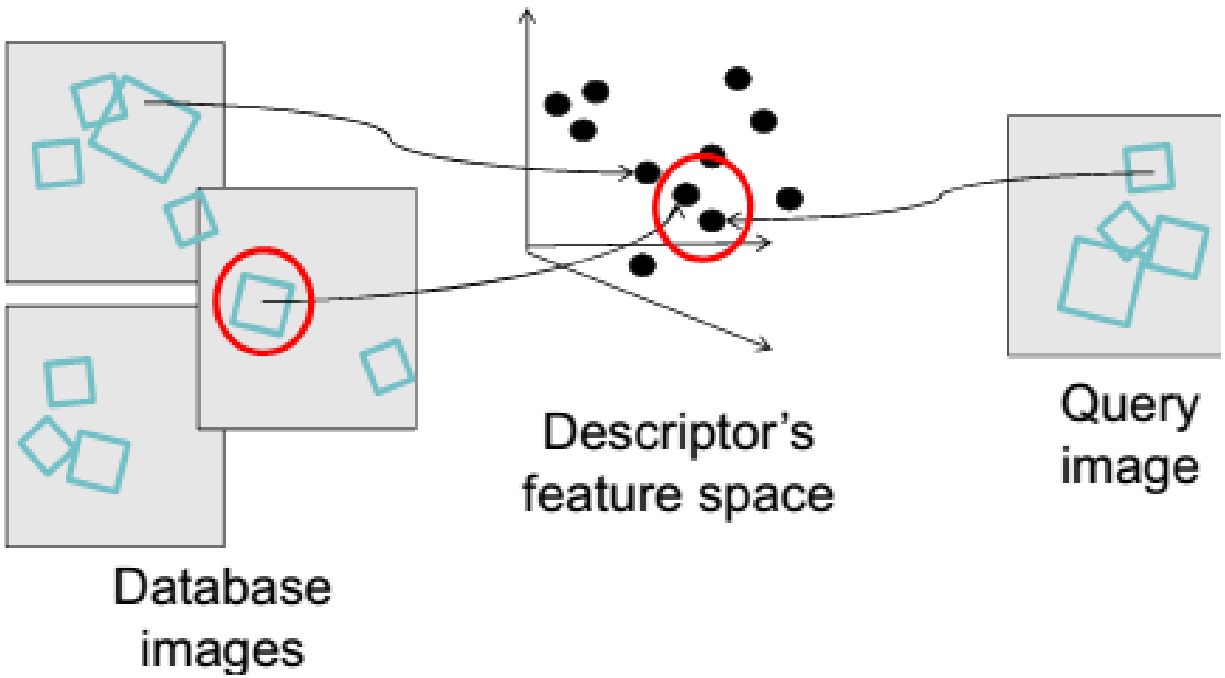
Indexing Local Features

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



Indexing Local Features

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



Indexing Local Features

- For text documents, an efficient way to find all **pages** on which a **word** occurs is to use an **index**…
- Here, we want to find all **images** in which a **feature** occurs.
- To use this idea, we need to map our features to this “**index**” of visual words.

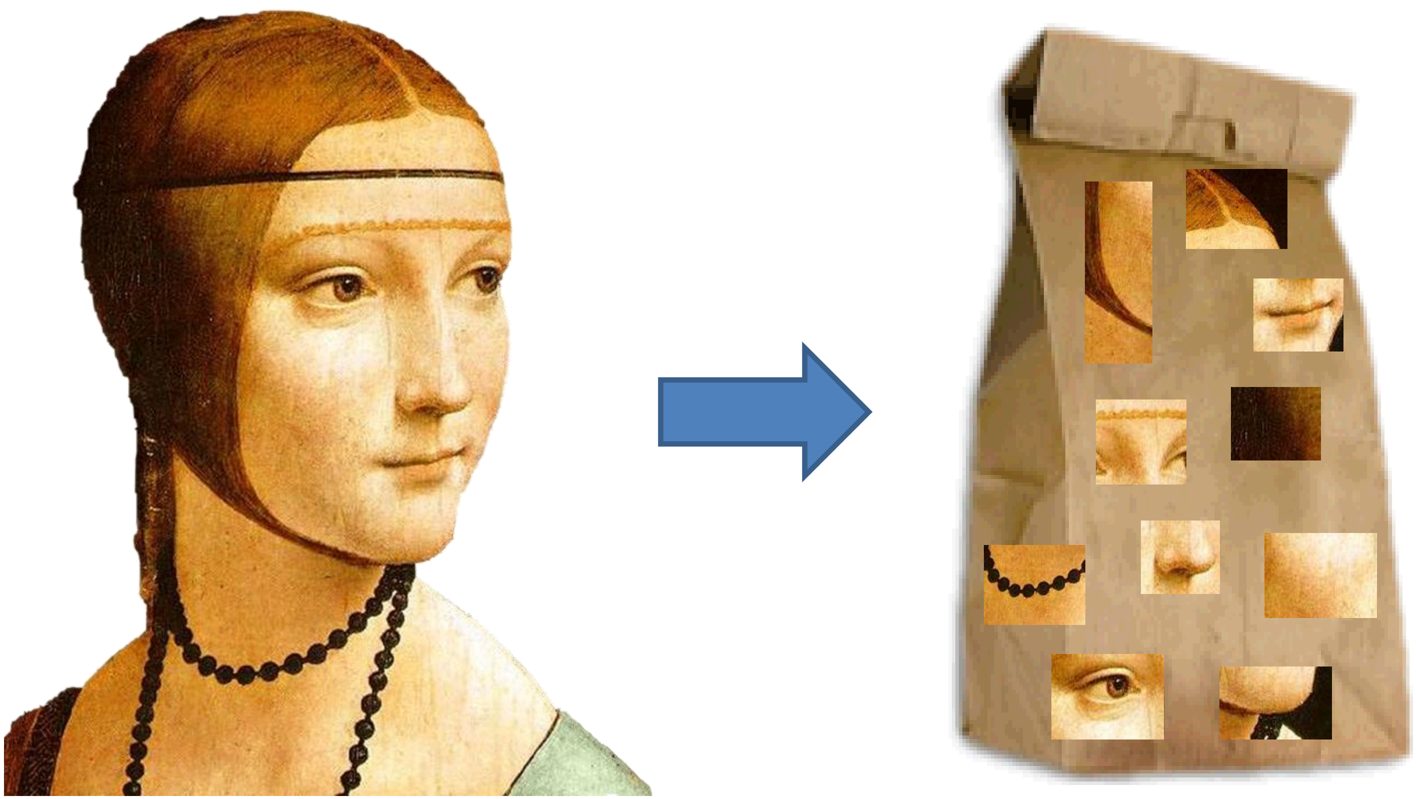
Text retrieval vs. image search

- So, what makes the problems similar, or different?

A. Different because images and texts have different dimensions.
B. Different because images describe visual details while text describe high level concepts.
C. Similar because we can have a dictionary for image features like a dictionary for text words.
D. Similar because image features are the same vectors as text word vectors.

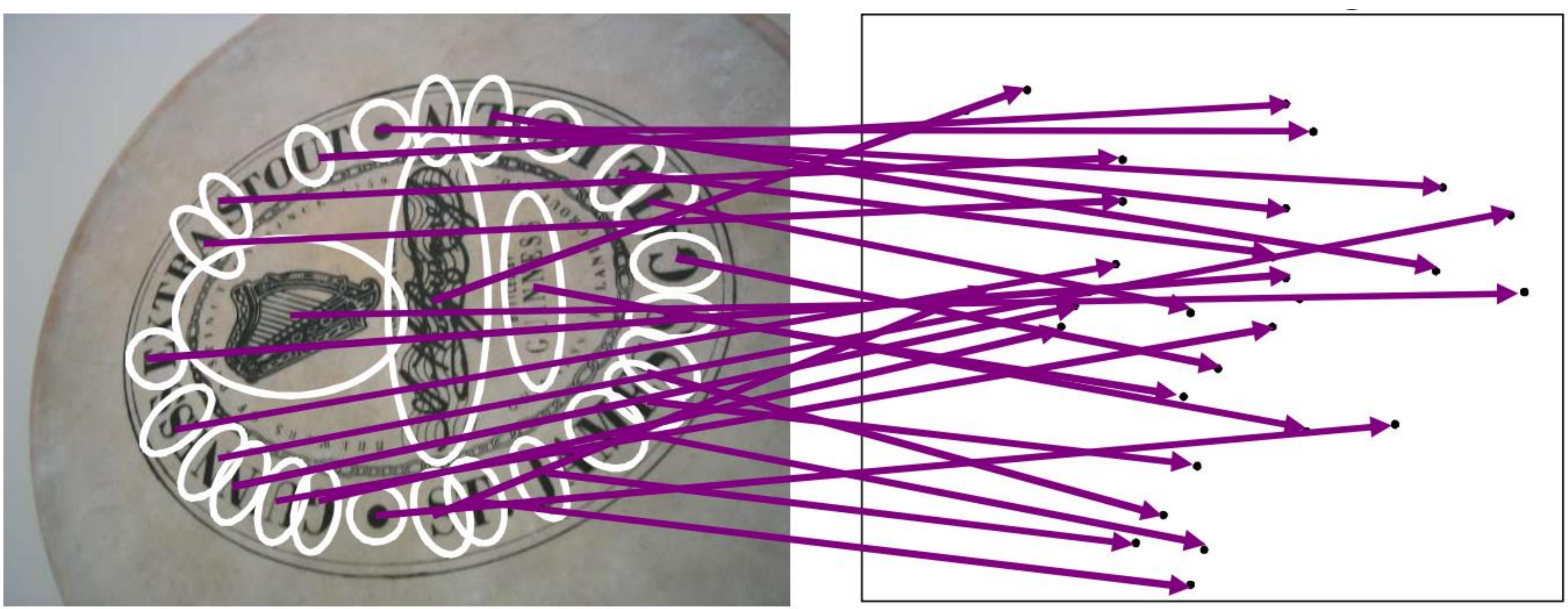
Concept of Visual Words

Visual words



Visual words

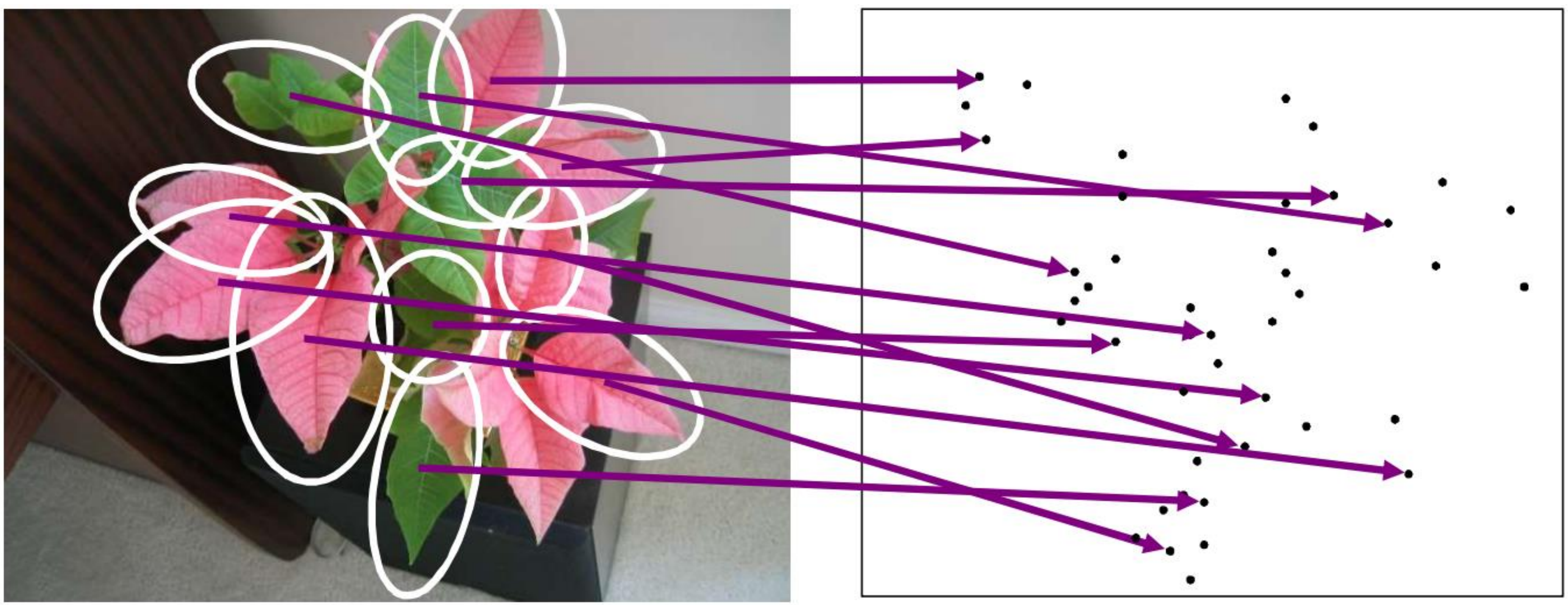
- Extract some local features from a number of images…



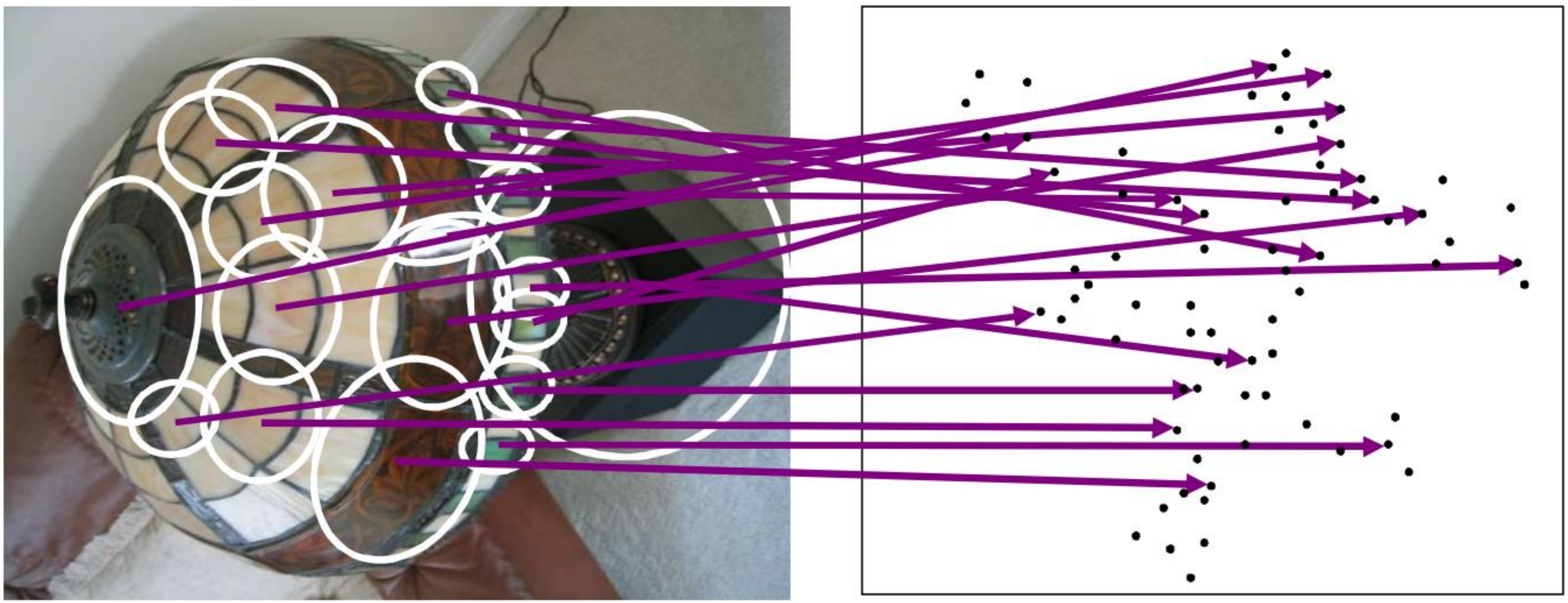
e.g. SIFT descriptor space: each point is 128-dimensional

Visual words

- Of course, it is impossible to visualize 128-dimensions! So, most of the time we show it in 2-D or 3-D only…

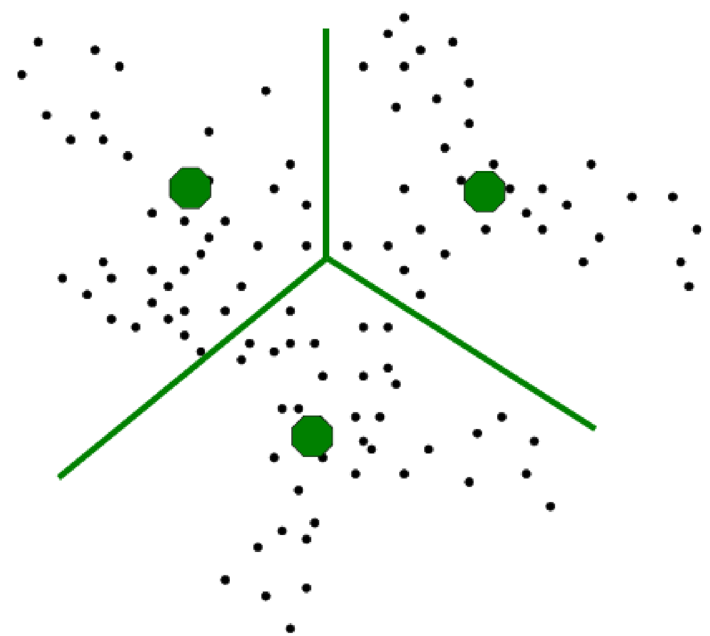


Visual words



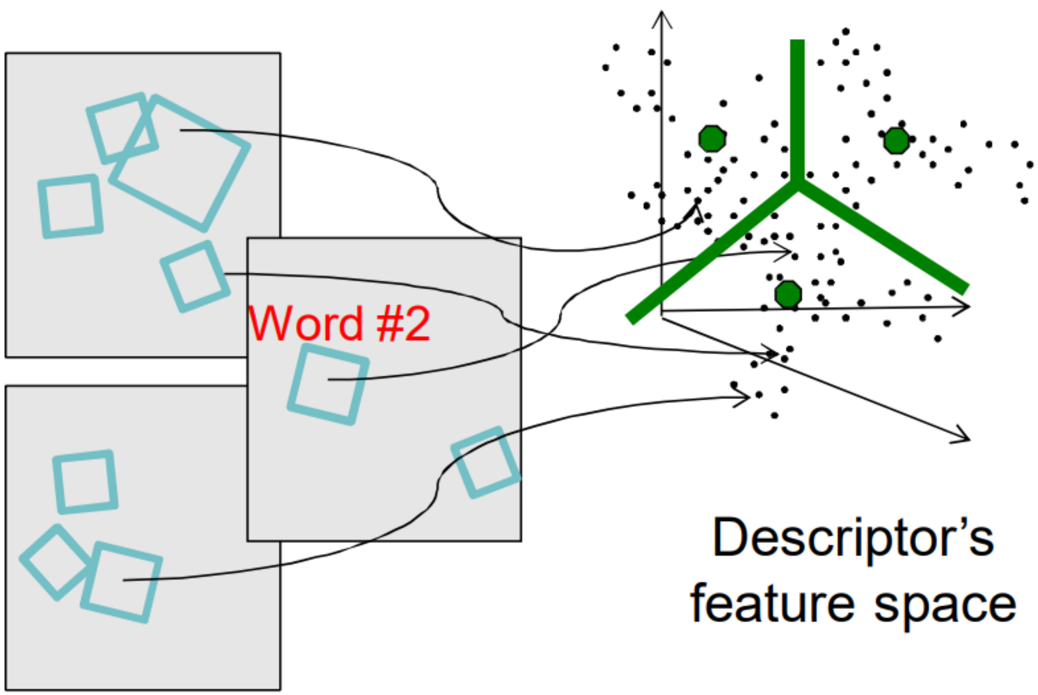
Visual Words

- Next, we want to try group these points (each 128-dimensions) into groups which will reflect distinctive characteristics
- Solution: Use a clustering technique such as k-means



Quantizing the feature space

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

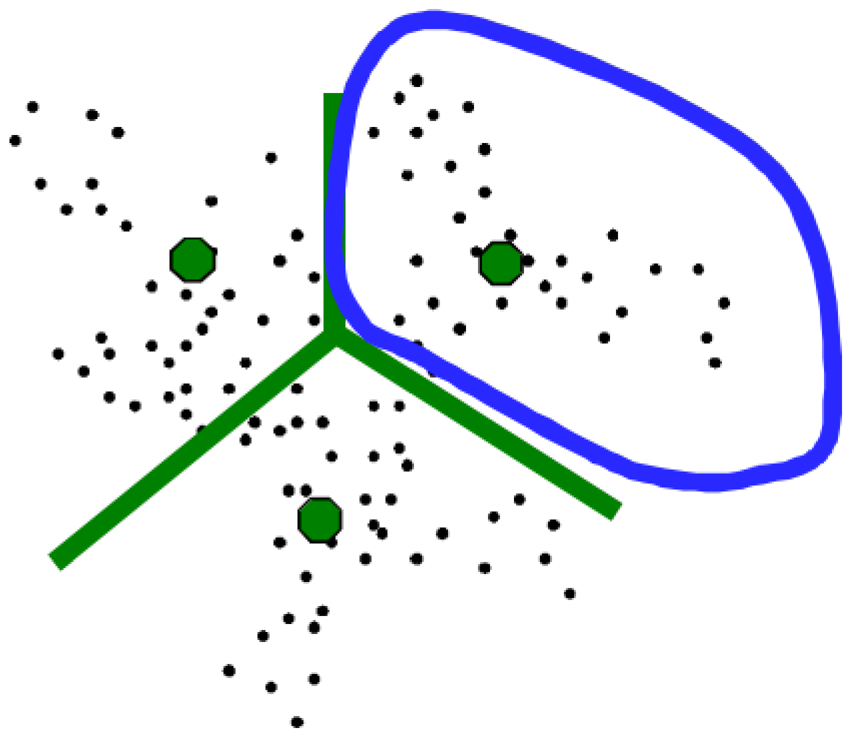


- **Clustering**: Let cluster centers be the representative of the "**words**"
- **Quantization:** Determine which word to assign to each image descriptor by finding the closest cluster center

Quantizing the feature space

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

  Look how similar they are after performing clustering



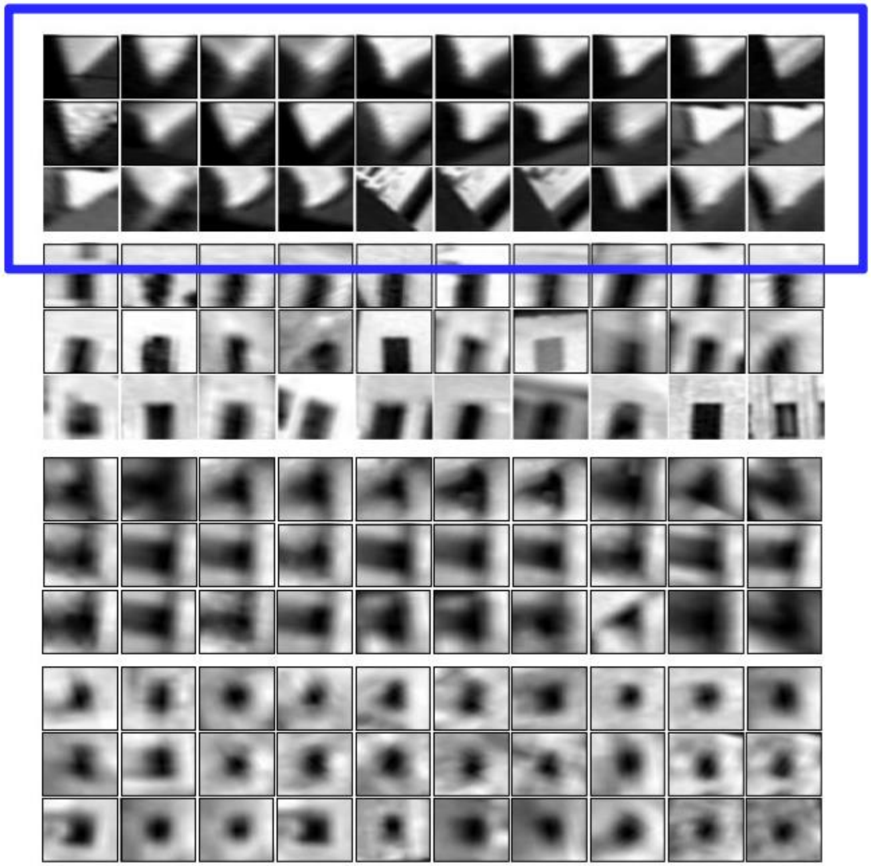
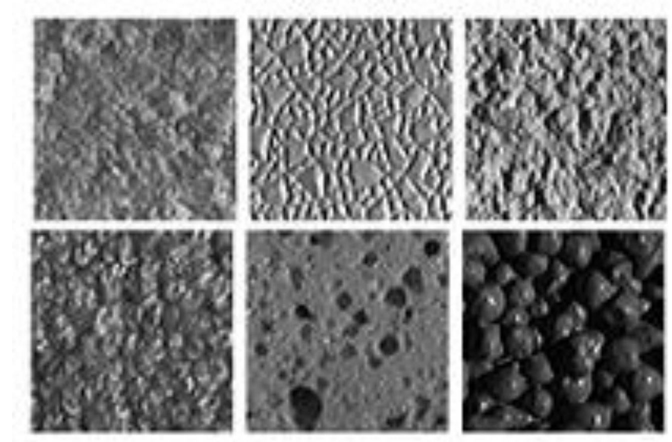
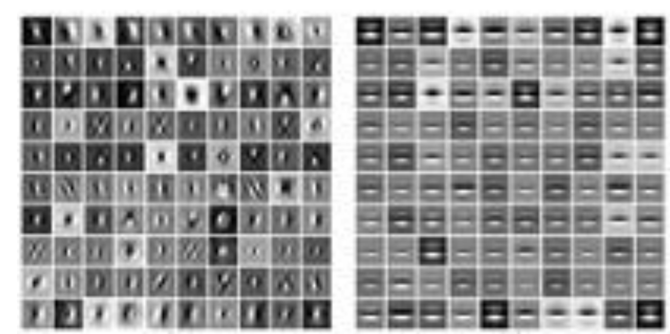


Figure from Sivic & Zisserman, ICCV 2003

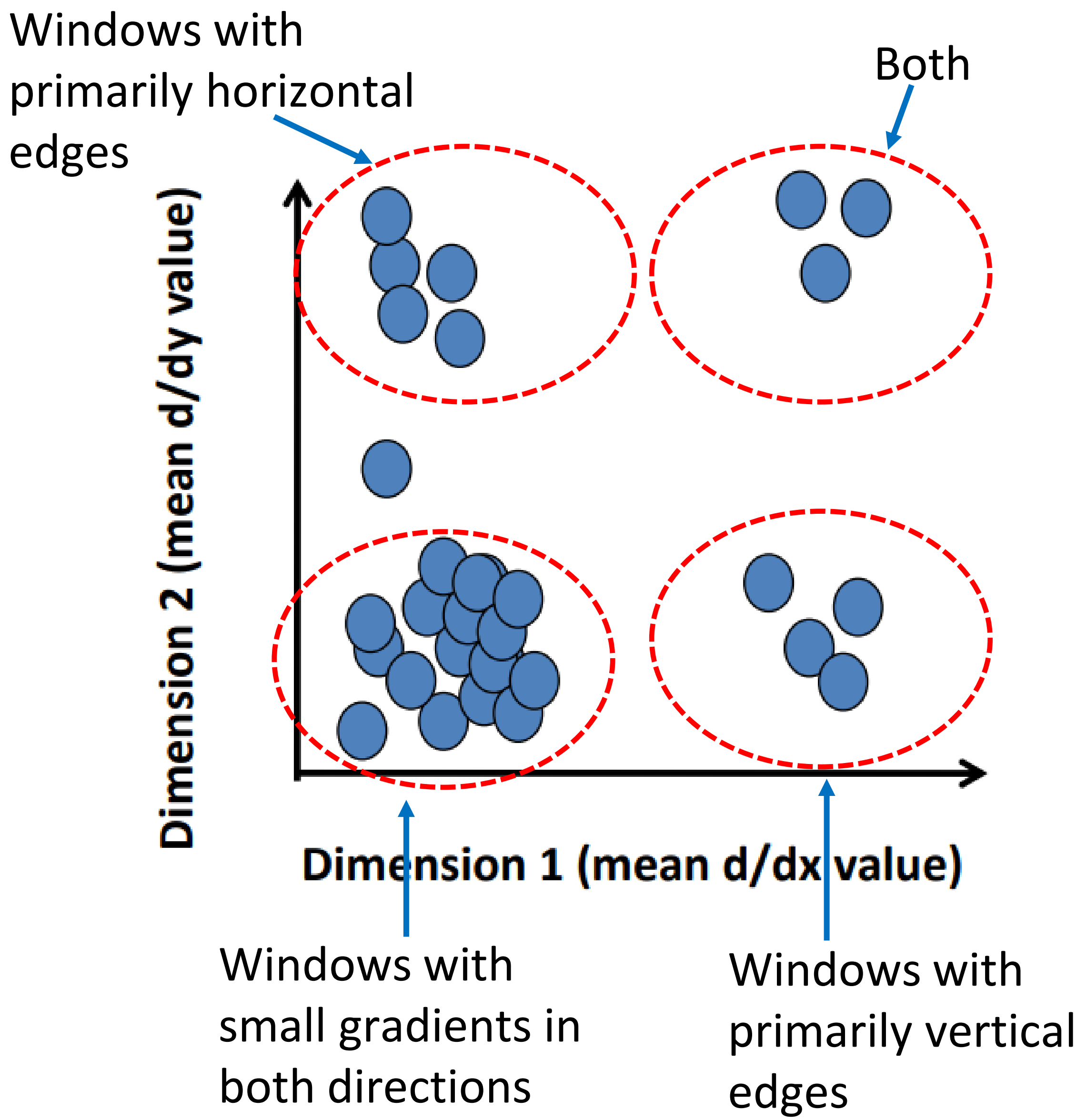
Visual words and "textons"

- First explored in texture and material representations
- **Texton** = cluster center of filter responses over collection of images
- Describe textures and materials based on distribution of prototypical texture elements





Recall: Texture Representation

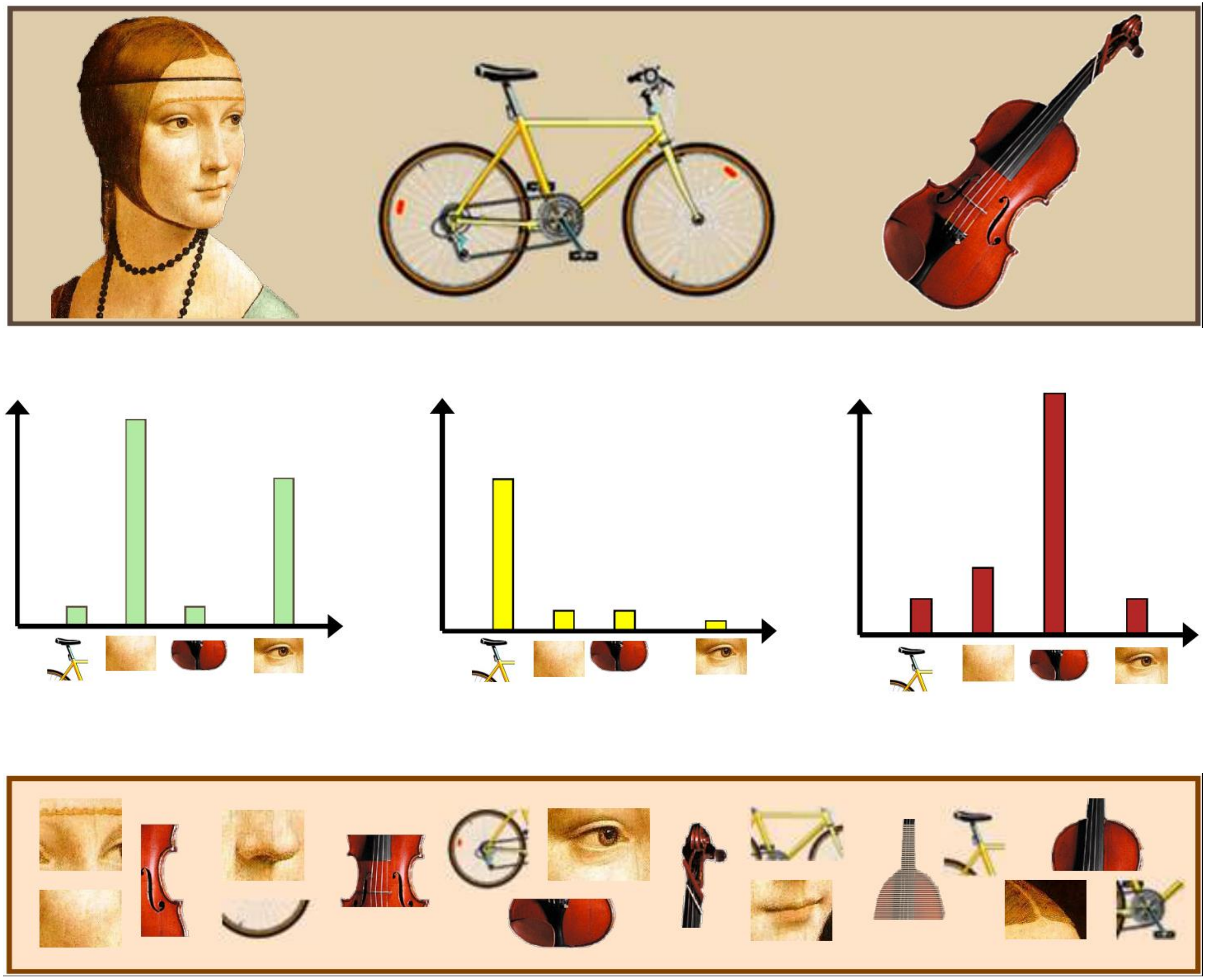


	Mean d/dx value	Mean d/dy value
Win. #1	4	10
Win. #2	18	7
⋮	⋮	⋮
Win. #9	20	20

•
•
•

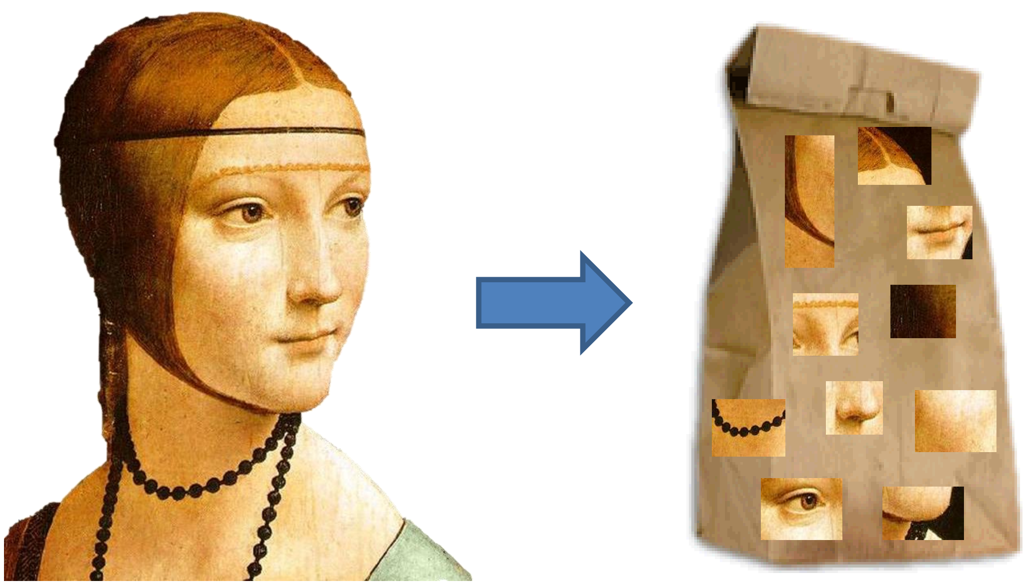
statistics to summarize patterns in small windows

Whole image in terms of its "parts"



Bag of visual words

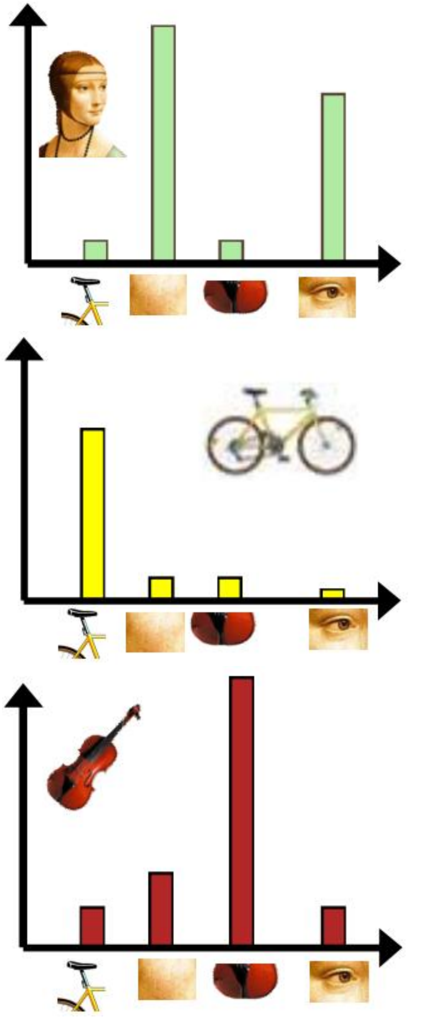
- Summarize entire image based on its distribution (histogram) of visual word occurrences
- Analogous to “bag of words” concept commonly used in documents



Bag of visual words

- Summarize entire image based on its distribution (histogram) of visual word occurrences
- Analogous to “bag of words” concept commonly used in documents

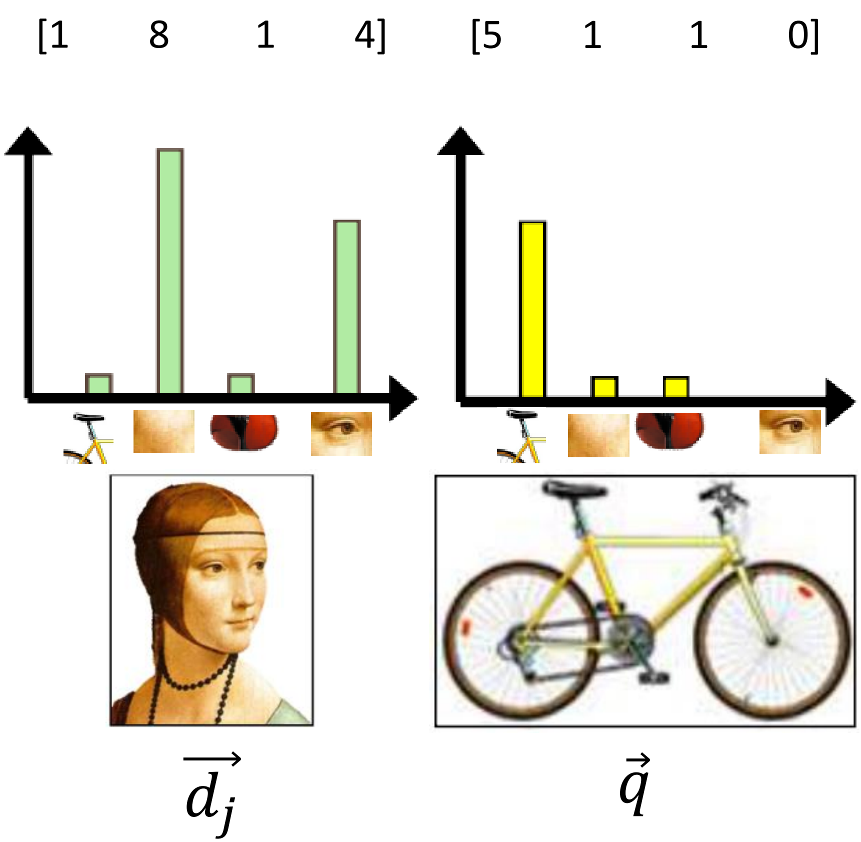
Visual words



Bag of visual words

Comparing bag of words

- Rank frames by normalized scalar (dot) product between their (possibly weighted) occurrence counts
  – nearest neighbour search for similar images



Cosine similarity

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

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

for vocabulary of *V* words

Visual vocabulary formation

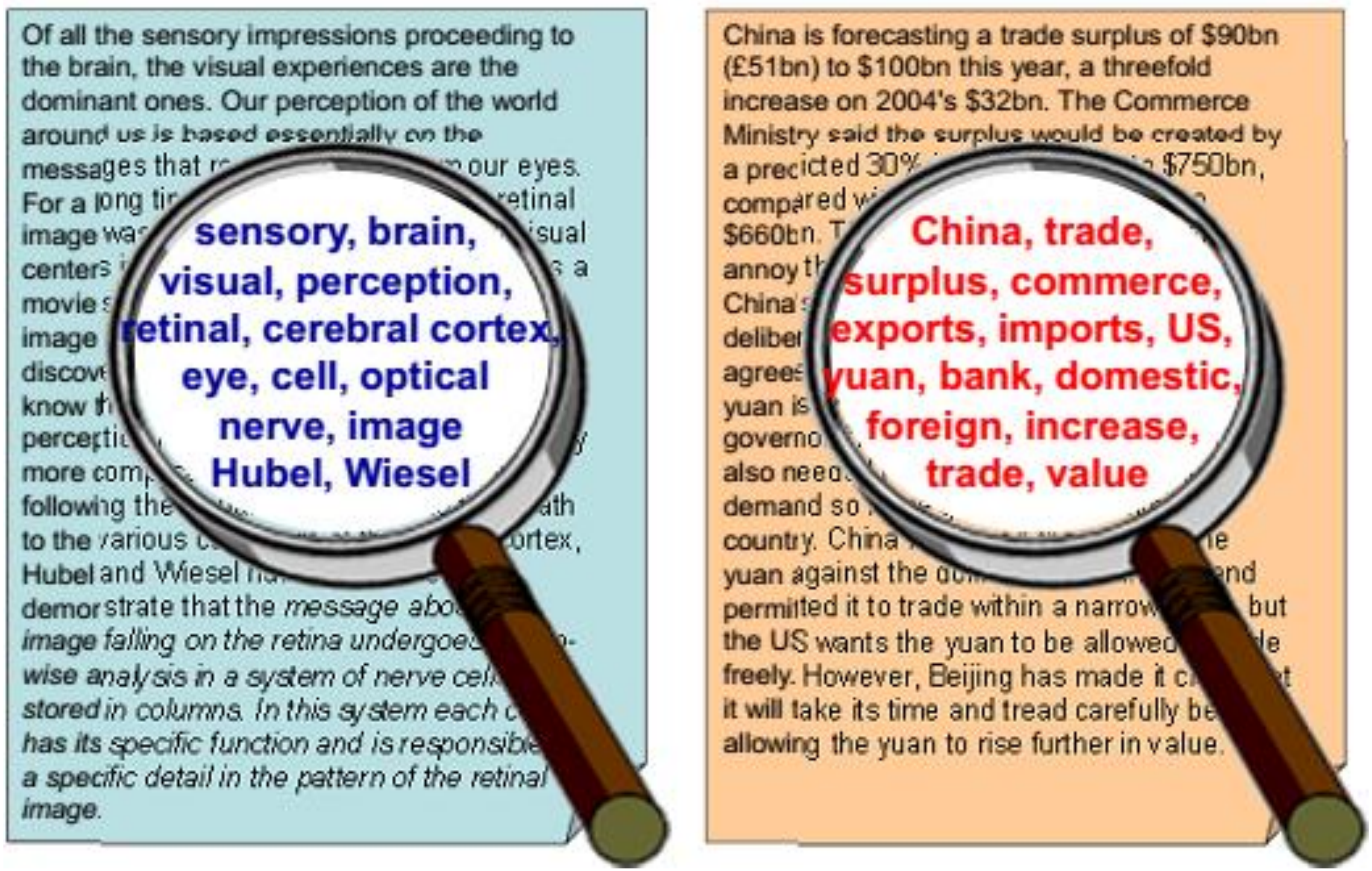
- Issues:
  - Sampling strategy: where to extract features?
    - A complex image can contain features in irrelevant portions of the image
  - Clustering / quantization algorithm
    - What are some problems of k-means?
  - Vocabulary size – number of words
    - What’s a good number of features to be used for representation

Inverted File Indexing

Representing image with visual words

- If a local image region is a visual “word”, how can we summarize an entire image (the “document”) ?

Analogy to documents



Inverted file index

Database Images

Image #1

Image #2

Image #3

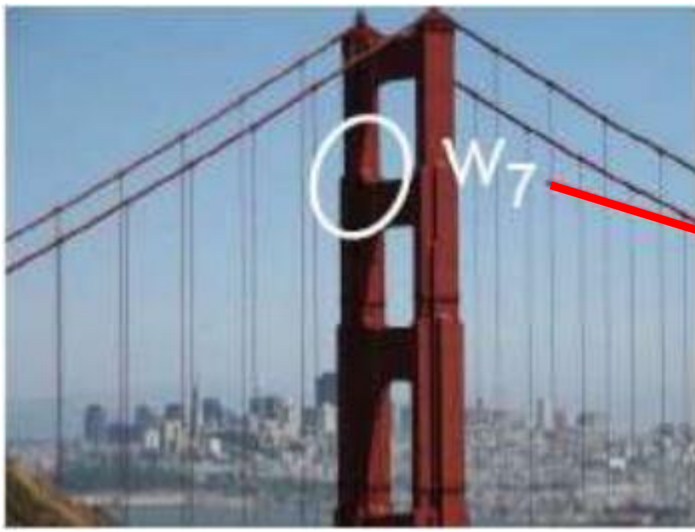
⋮

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

…

- Database images are loaded into the index mapping words to image numbers

Inverted file index



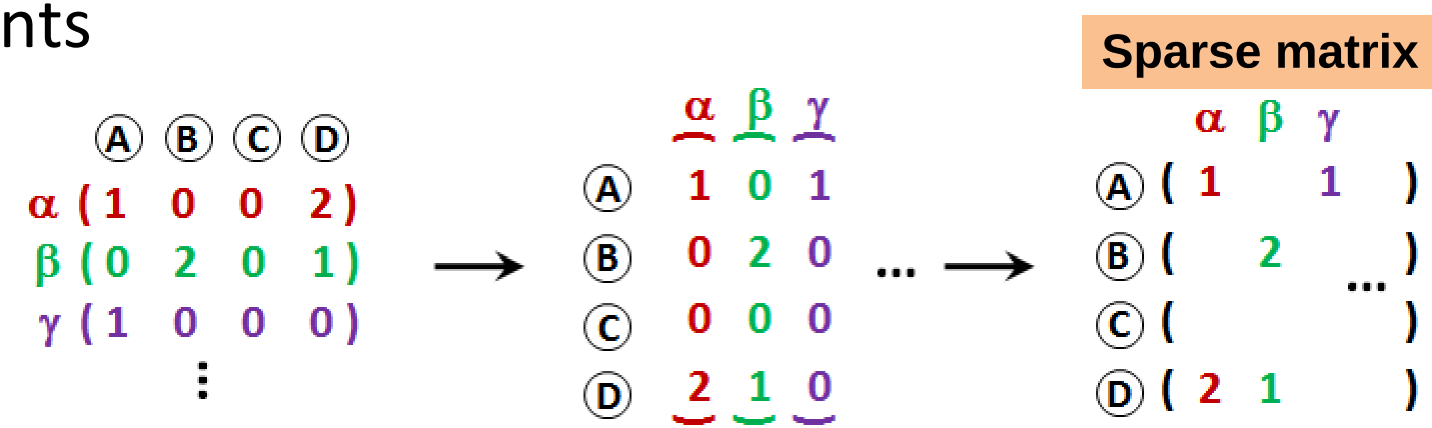
Word #	Image #
1	3
2	
⋮	
7	1, 2
8	3
9	
10	
⋮	
91	2
⋮	⋮

…

- A new query image is mapped to indices of database images that share a particular word

Inverted file index

- The visual vocabulary can be very big (thousands to millions of words)
- For quick searching, use an **inverted file index** in sparse form to reduce the size of a matrix that has many zero elements



- Logically, the **weight of each word** computed from its frequency divided by length of vector of words could be useful…

tf-idf weighting

- **T**erm **f**requency – **i**nverse **d**ocument **f**requency
- Describe by frequency of each word within it, **downweight words** that appear often in database
- Standard weighting for text retrieval – can be applied to image search too

Number of occurrences of word i in document d

Total number of documents in database

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

Number of words in document d

Number of documents word i occurs in, in whole database

Common to perform normalization later, using just the raw frequency

Denominator can be 0 if the word does not occur. So normally, we can use $\log \frac{1+N}{1+n_i} + 1$ to solve this problem

tf-idf example

- **Remember:** document ⇒ image

$$t_{id} = n_{id}\left(\log\frac{1+N}{1+n_i} + 1\right)$$

N = number of images in database
n_i = number of images with word i
n_{id} = number of occurrences of word i in image d

For Image 0:

$n_0 = 6, n_{0,0} = 3$

$$t_{0,0} = 3\left(\log\frac{1+6}{1+6} + 1\right) = 3$$

$$t_{1,0} = 0\left(\log\frac{1+6}{1+1} + 1\right) = 0$$

$$t_{2,0} = 1\left(\log\frac{1+6}{1+2} + 1\right) = 1.368$$

3 words

$N = 6$

Word, i	0	1	2	Image, d
Counts =	[[3,	0,	1],	0
⋯	[2,	0,	0],	1
⋯	[3,	0,	0],	2
⋯	[4,	0,	0],	3
⋯	[3,	2,	0],	4
⋯	[3,	0,	2]]	5

[3, 0, 1.368]

[0.9099, 0, 0.4149]

We see an increase in this value.... Why?

Due to these changes, normalization MUST be performed after that, by normalizing the tf-idf value by its magnitude (Euclidean norm: $\|n_{id}\|_2$)

tf-idf example

Word, i	0	1	2	Image, d
Counts =	[[3,	0,	1],	0
…	[2,	0,	0],	1
…	[3,	0,	0],	2
…	[4,	0,	0],	3
…	[3,	2,	0],	4
…	[3,	0,	2]]	5

[3, 0, 1.368]

We see an increase in this value. Why?

A. The visual word has the least number of occurrence in the database, so it is most important.
B. The visual word has less number of occurrence but in various images, so it is useful as an index.
C. The visual word frequency needs to be normalized hence the change in value.
D. The frequency properly reflects the general count of words, not only in this database.

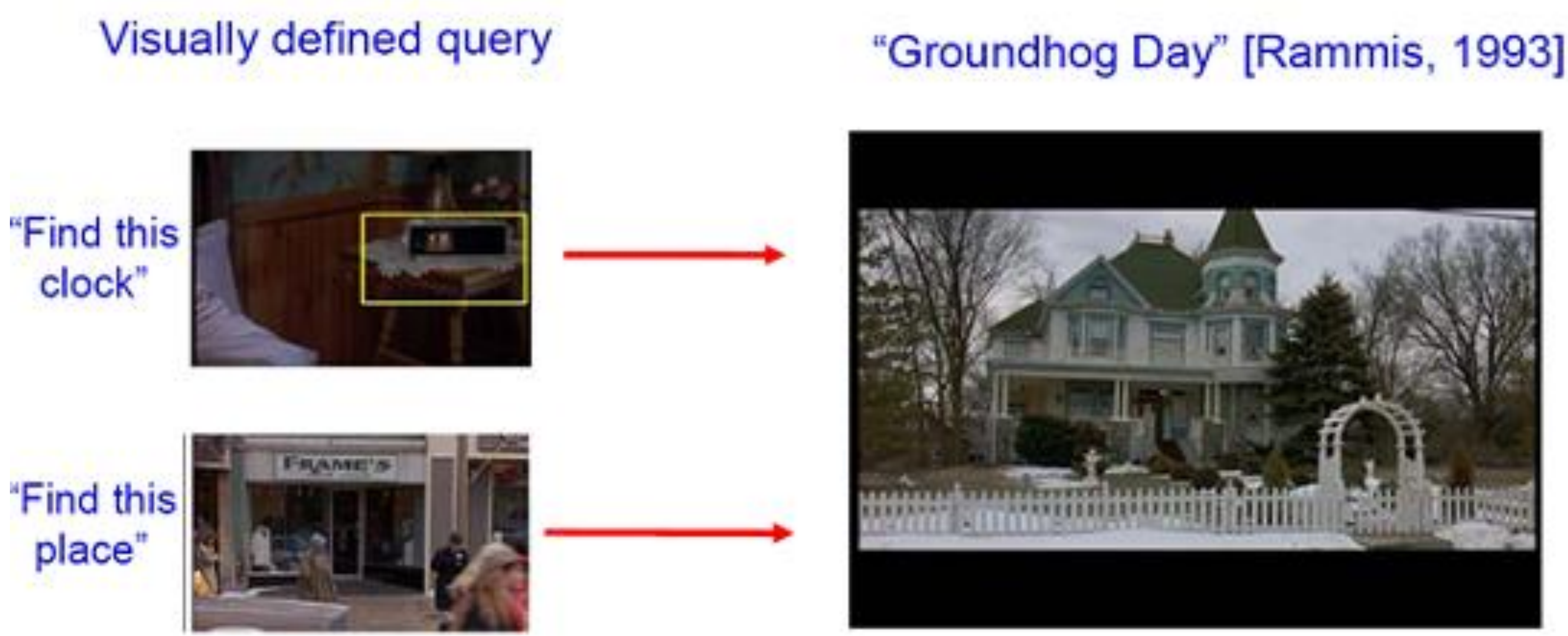
BOW: Order-less representation

- Bag-of-Words ⇒ **orderless** representation (spatial relationships between features are gone)
- But we can use the following ideas to help…
  - **Visual "phrases"** – frequently co-occurring words
    Descriptive visual words and visual phrases for image applications
  - Let **position** be part of each feature
  - **Localize it further**: Perform BOW only within sub-grids or blocks of an image
  - After matching, **verify spatial consistency** (look at neighbours, are they same too?)

Application and Scoring

Application of BOW for image retrieval

- Retrieve an object from video that matches the query region



Slide from Andrew Zisserman
Sivic & Zisserman, ICCV 2003

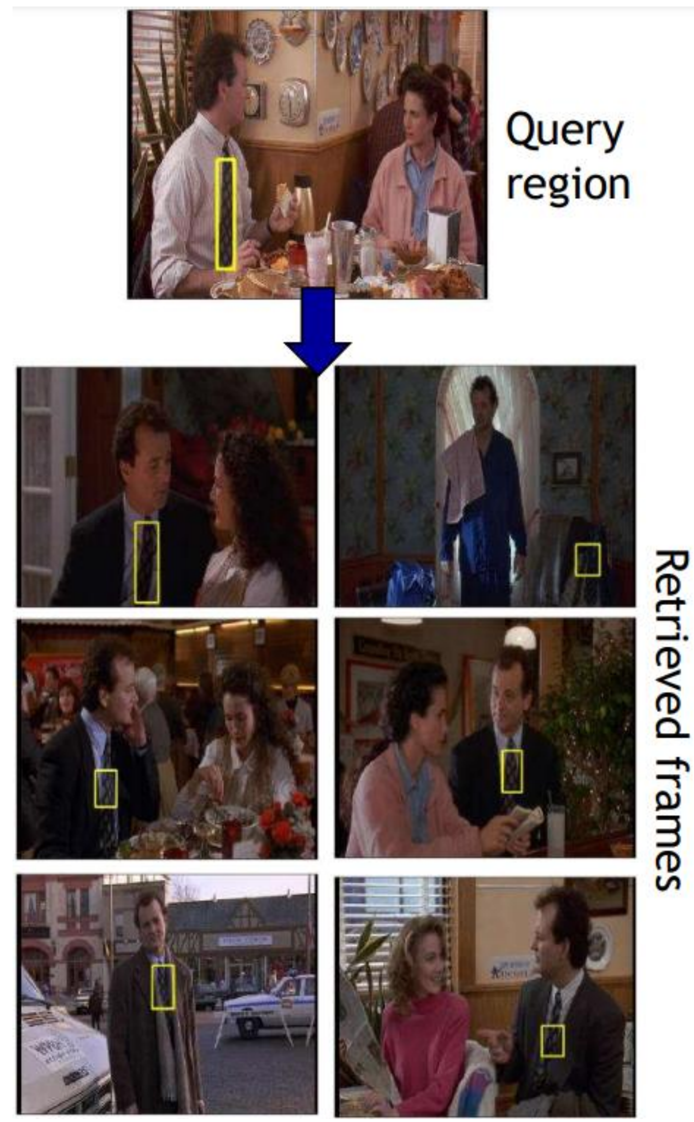
Video Google System

- "Object matching" in videos

Slide from Andrew Zisserman
Sivic & Zisserman, ICCV 2003

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

http://www.robots.ox.ac.uk/~vgg/research/vgoogle/index.html

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

http://www.robots.ox.ac.uk/~vgg/research/vgoogle/index.html

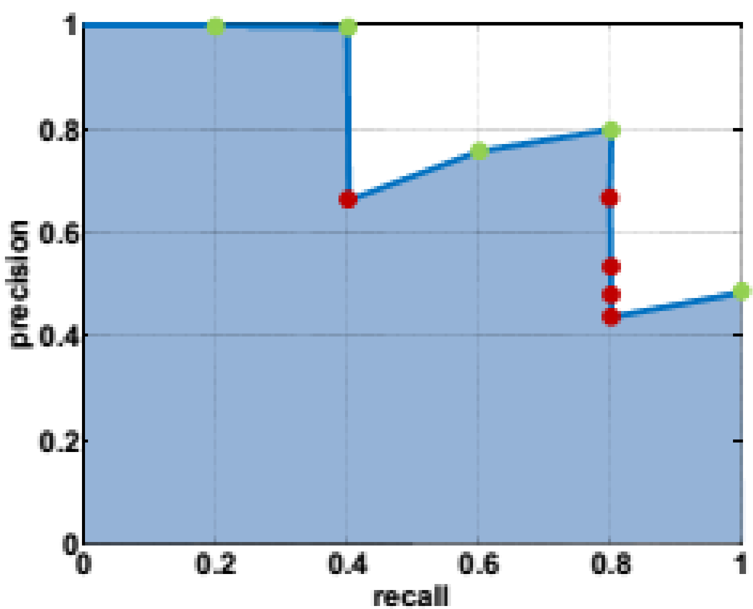
Scoring retrieval quality

Example:
Database size: 10 images
Relevant (total): 5 images

Query

Precision = # relevant / # returned
Recall = # relevant / # total relevant

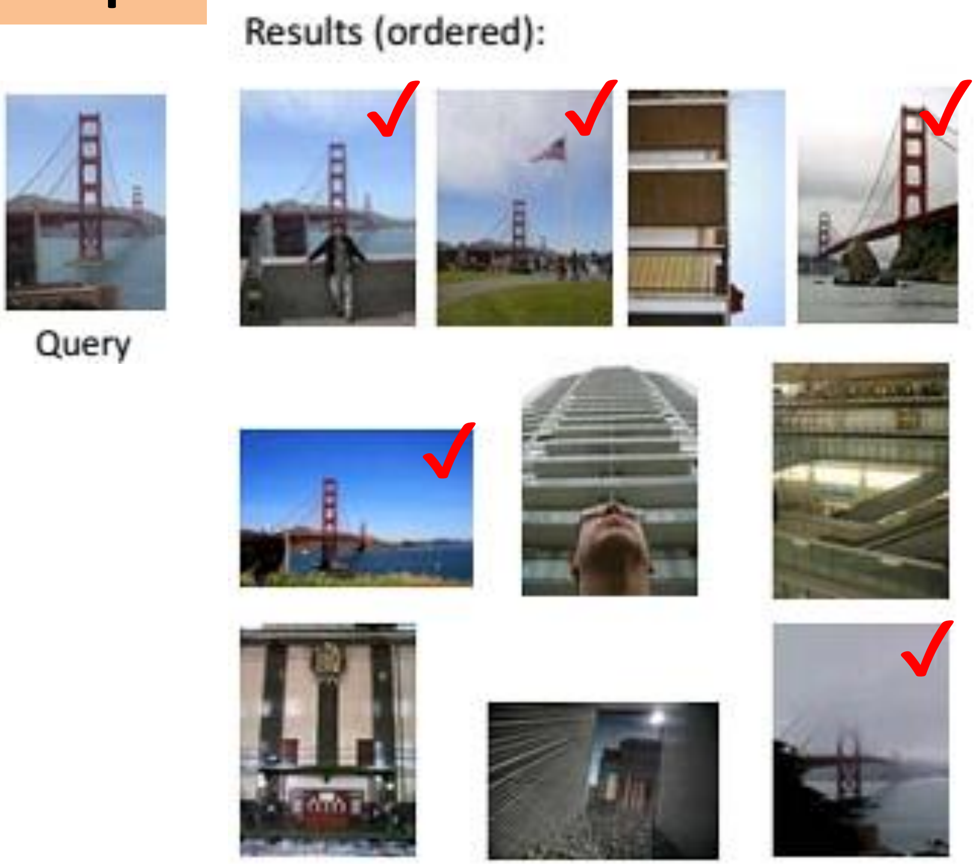
Results (ordered):



Precision-Recall curve

Scoring retrieval quality

Example



Total returned images: 10

- This can be tuned by setting a threshold on the distance/similarity score

Relevant images retrieved: 5

- Based on ground truth, the number of bridge images returned.

Total relevant images: 8

- Based on ground truth, the number of the bridge images that should have been returned

Recall = 5/8

Precision = 5/10

Bag of words: Pros and Cons

- **Pros**
  - Flexible to geometry / deformations / viewpoint
  - Compact summary of image content
  - Provides vector representation for sets
  - Very good results in practice
- **Cons**
  - Basic model ignores geometry – must verify or encode via features
  - Background and foreground mixed when bag covers whole image
  - Optimal vocabulary formation remains unclear (how many words?)

Summary

- Matching local invariant features
  - Useful not only to provide matches for multi-view geometry, but also to find objects and scenes in an image retrieval task
- Bag of words (BOW) representation
  - Quantize feature space to make discrete set of visual words – summarize image by distribution (or histogram) of words – then index these words
- Inverted index
  - Pre-compute index to enable faster search at query time