

14. Bag-of-Words

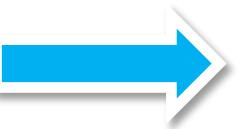


Image Classification



(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

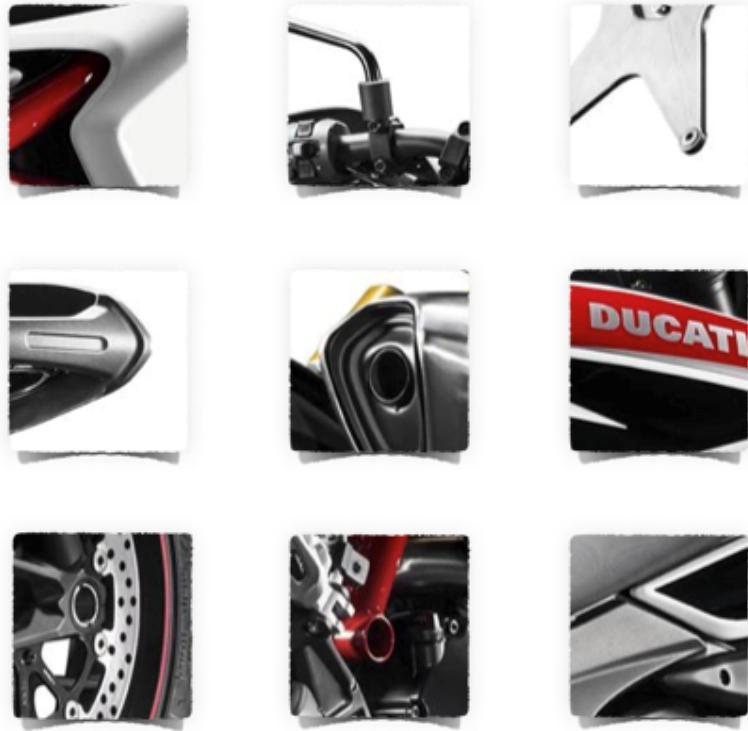
→ cat

More classification later
We focus on image search first

Some local feature
are very informative



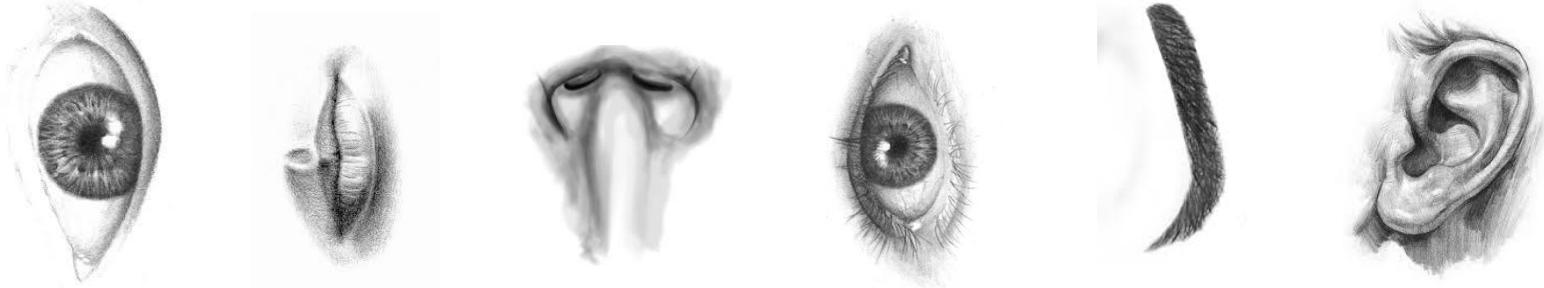
An object as



a collection of local features
(bag-of-features)

- deals well with occlusion
- scale invariant
- rotation invariant

(not so) crazy assumption



spatial information of local features can be ignored for object
recognition (i.e., verification)

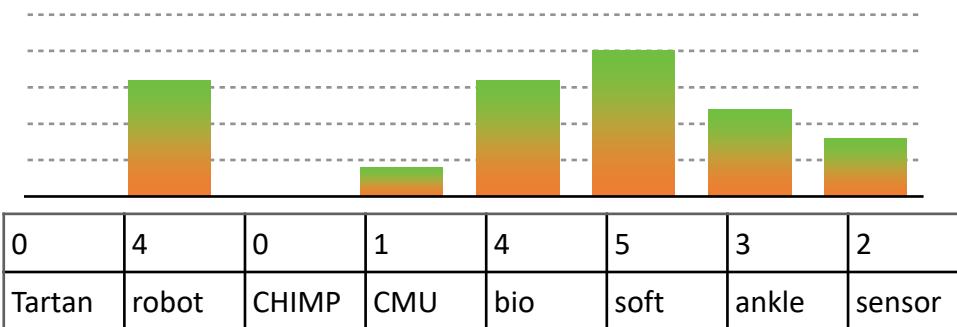
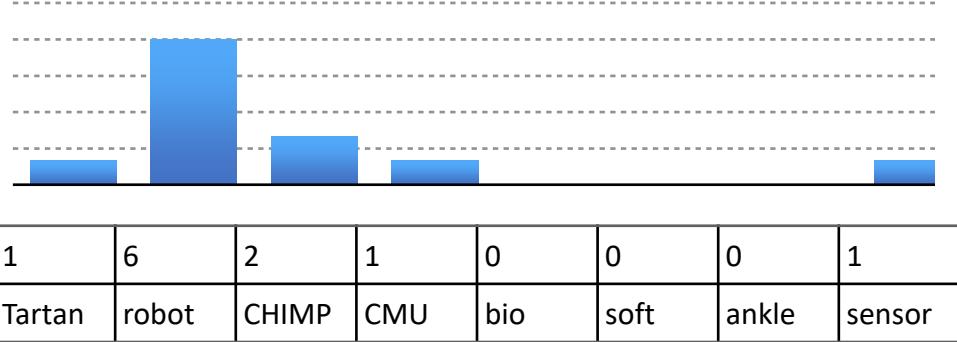


Bag-of-features

- Represent a data item (document, texture, image) as a **histogram** over features
- an old idea (e.g., texture recognition and information retrieval)



Vector Space Model





Vector Space Model

- A document (datapoint) is a vector of counts over each word (feature)

$$\mathbf{v}_d = [n(w_{1,d}) \ n(w_{2,d}) \ \cdots \ n(w_{T,d})]$$

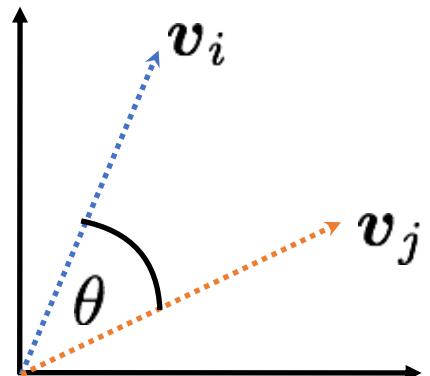
$n(\cdot)$ counts the number of occurrences



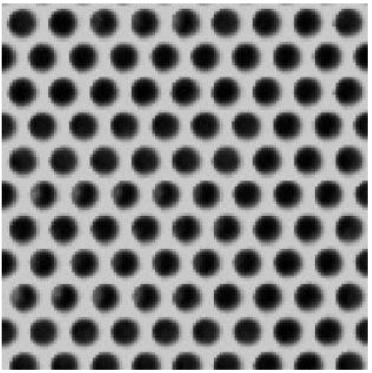
just a histogram over words

- What is the similarity between two documents?
 - Use any distance you want but the cosine distance is fast

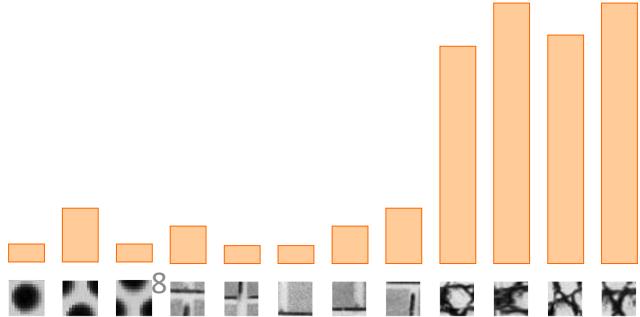
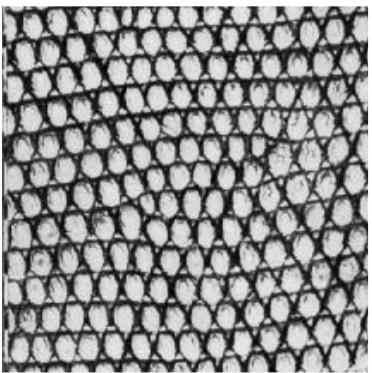
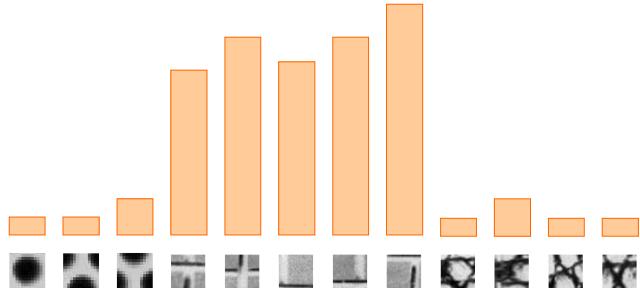
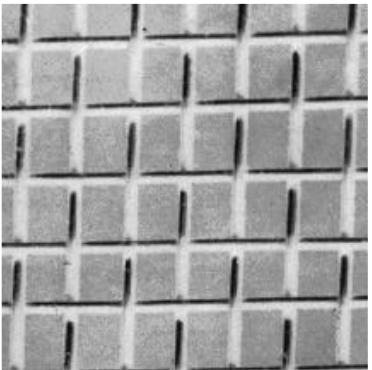
$$\begin{aligned} d(\mathbf{v}_i, \mathbf{v}_j) &= \cos \theta \\ &= \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|} \end{aligned}$$



Texture Recognition

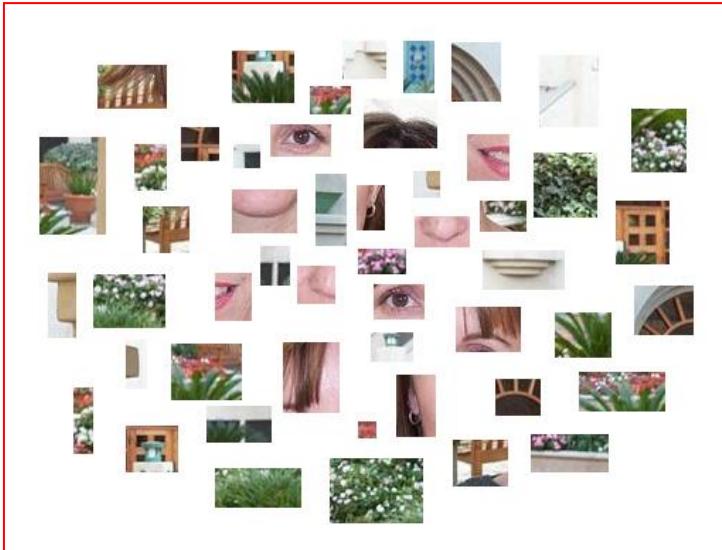
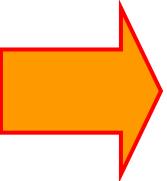


Universal texton dictionary



Bags of features for object recognition

- Works pretty well for image-level classification and for recognizing object *instances*



face, flowers, building

Bags of features for object recognition



class	bag of features		Parts-and-shape model Fergus et al. (2003)
	Zhang et al. (2005)	Willamowski et al. (2004)	
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

Questions?



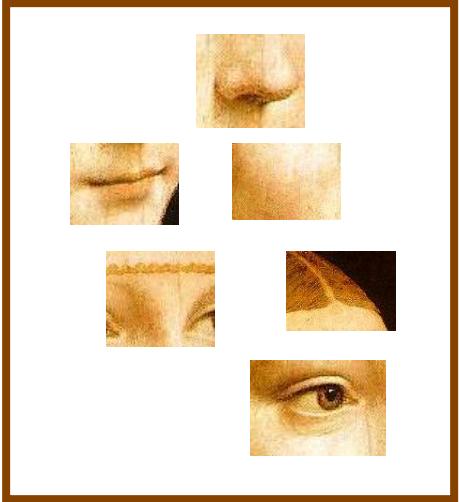


Bag of Words Pipeline

- Dictionary Learning: take a bunch of images, extract features, and build up a “dictionary” or “visual vocabulary” – a list of common features
- Encode: given a new image, extract features and build a histogram – for each feature, find the closest visual word in the dictionary
- Classify: train and test data using BOW features (later)

Bag of Words: outline

1. Extract features



Bag of Words: outline

1. Extract features
2. Learn “visual vocabulary”



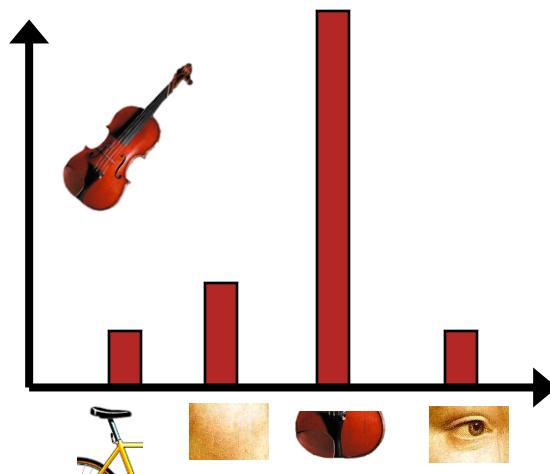
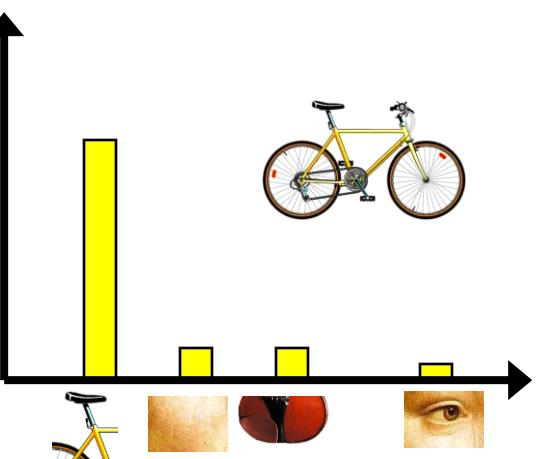
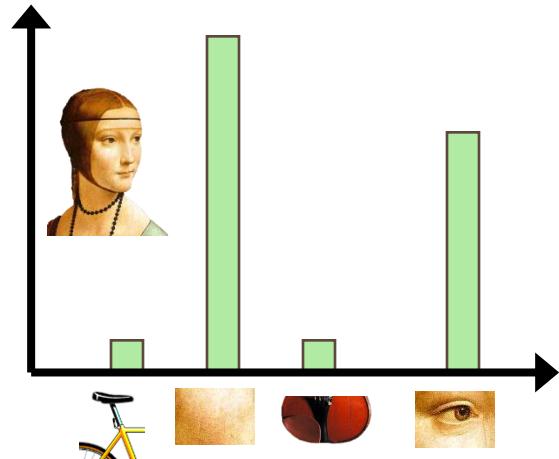


Bag of Words: outline

1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary

Bag of Words: outline

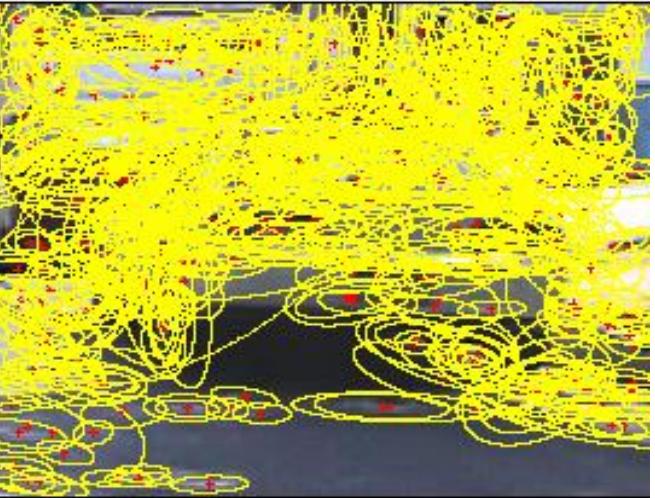
1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary
4. Represent images by frequencies of “visual words”



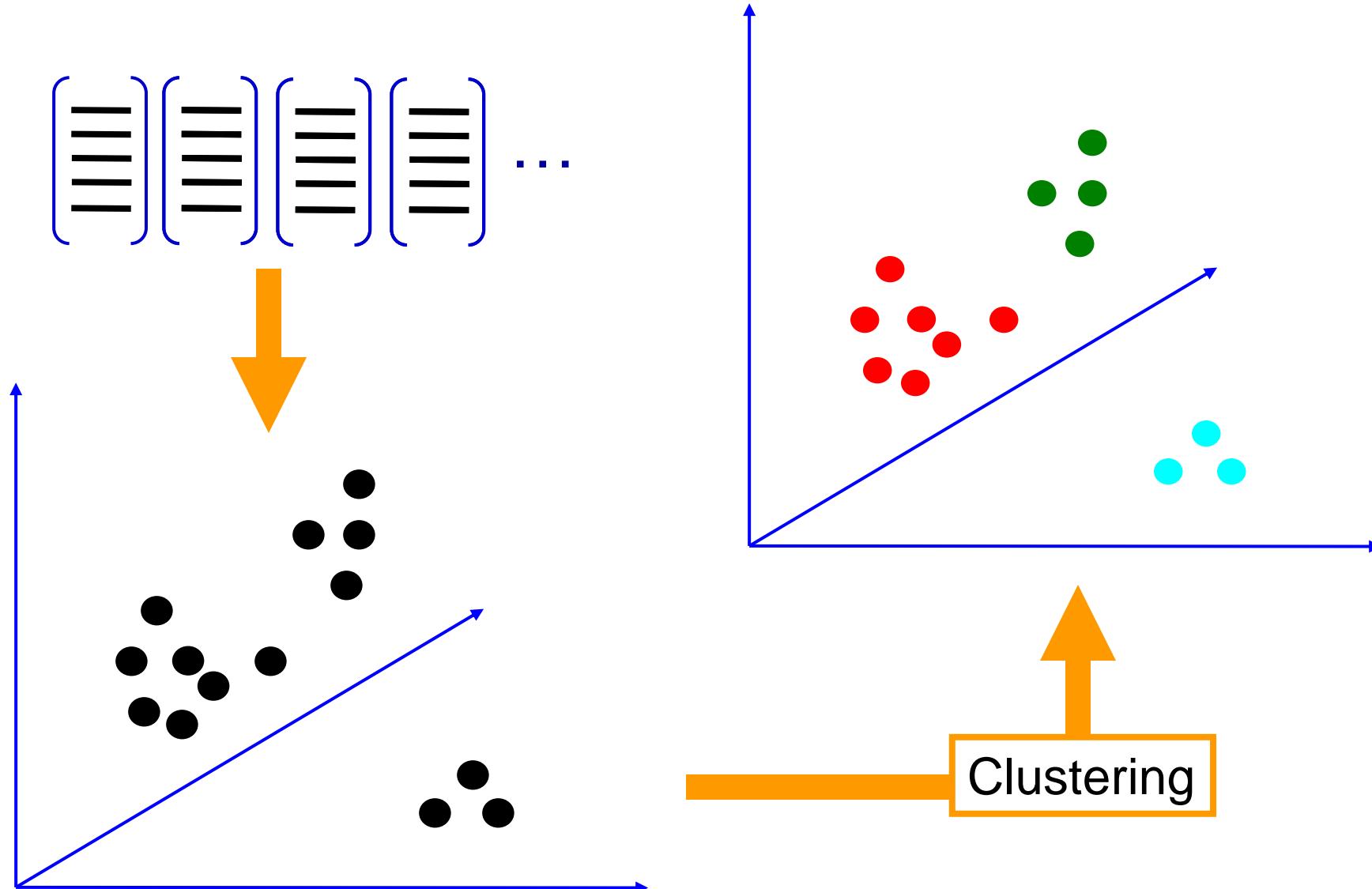
1. Feature extraction



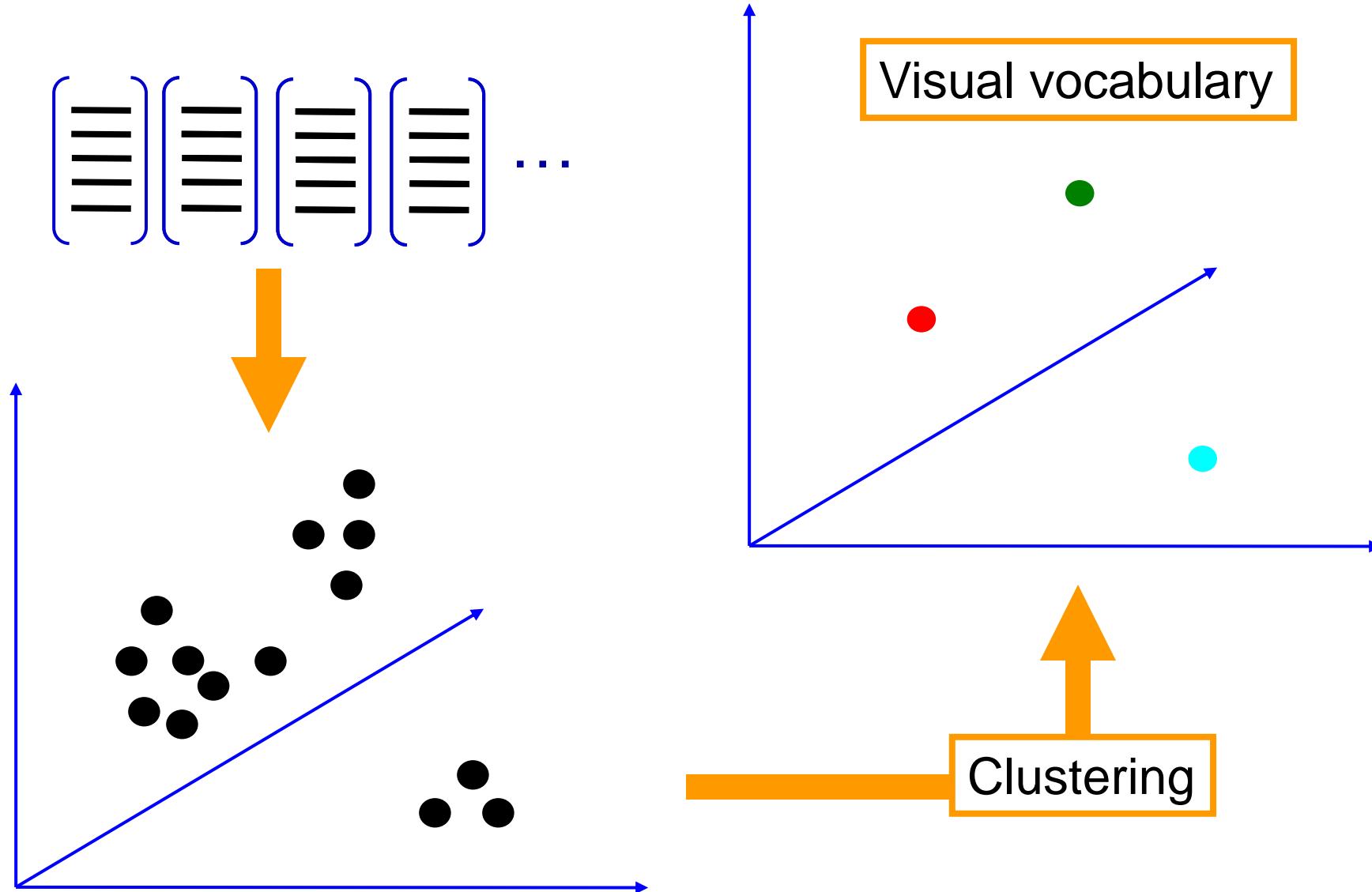
- Regular grid
 - Vogel & Schiele, 2003
 - Fei-Fei & Perona, 2005
- Interest point detector
 - Csurka et al. 2004
 - Fei-Fei & Perona, 2005
 - Sivic et al. 2005
- Other methods
 - Random sampling (Vidal-Naquet & Ullman, 2002)
 - Segmentation-based patches (Barnard et al. 2003)



2. Learning the visual vocabulary



2. Learning the visual vocabulary

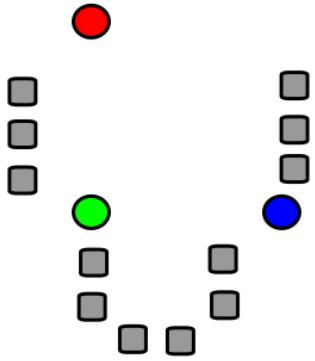


Slide credit: Josef Sivic

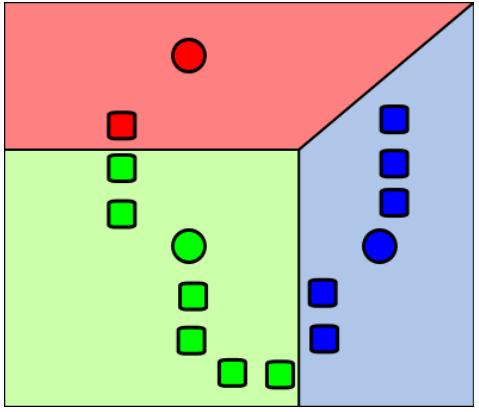
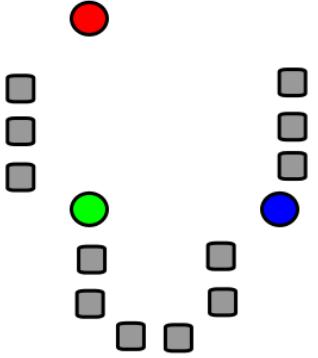


K-means Clustering

- Given k:
 1. Select initial centroids at random.
 2. Assign each object to the cluster with the nearest centroid.
 3. Compute each centroid as the mean of the objects assigned to it.
 4. Repeat previous 2 steps until no change.

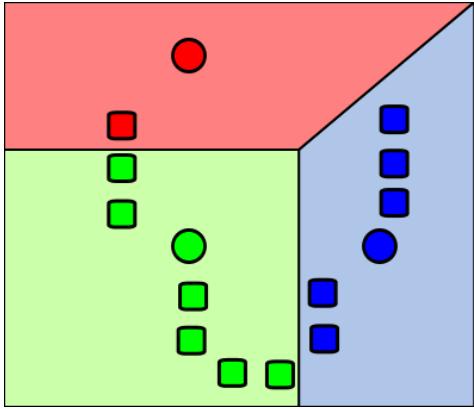
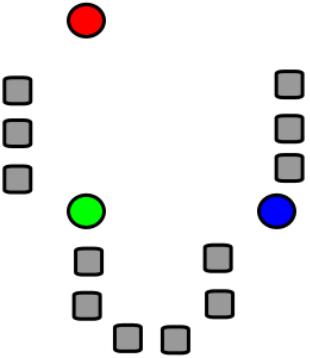


1. Select initial
centroids at random



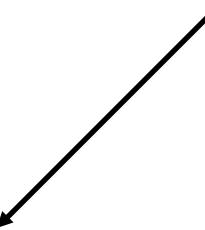
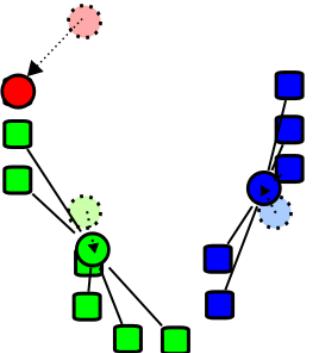
1. Select initial
centroids at random

2. Assign each object to
the cluster with the
nearest centroid.

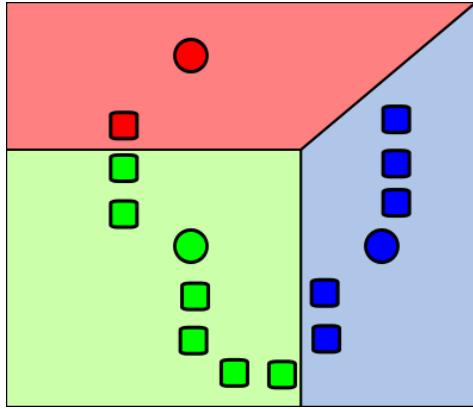
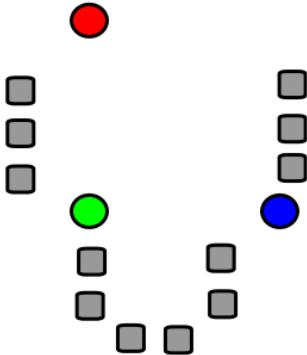


1. Select initial centroids at random

2. Assign each object to the cluster with the nearest centroid.

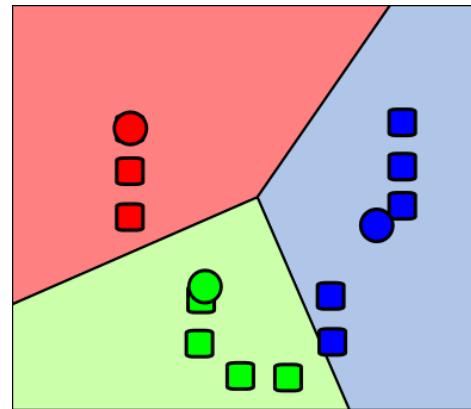
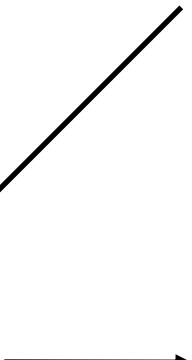
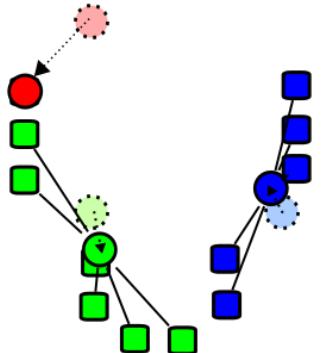


3. Compute each centroid as the mean of the objects assigned to it (go to 2)



1. Select initial centroids at random

2. Assign each object to the cluster with the nearest centroid.



3. Compute each centroid as the mean of the objects assigned to it (go to 2)

2. Assign each object to the cluster with the nearest centroid.

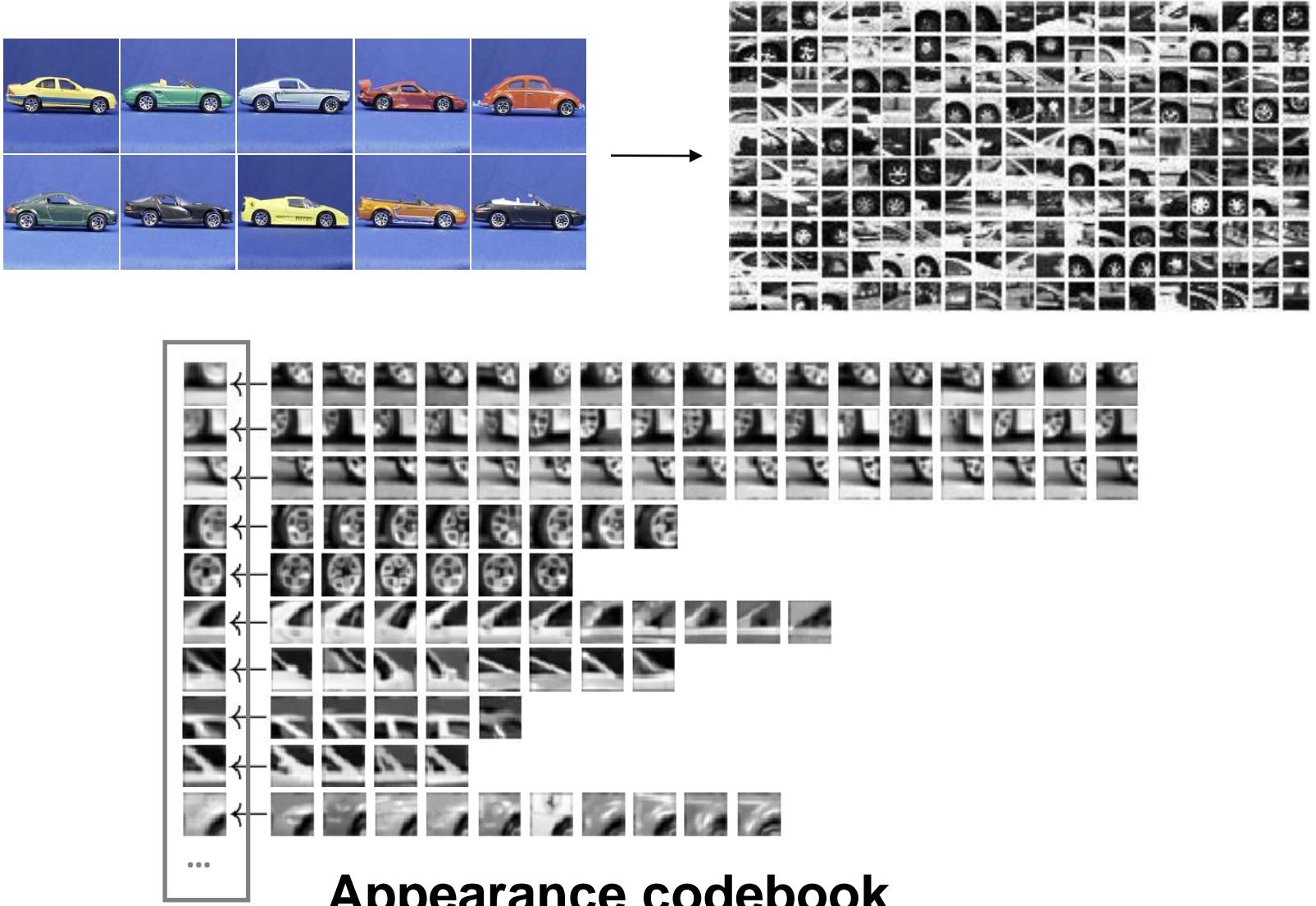
Repeat previous 2 steps until no change



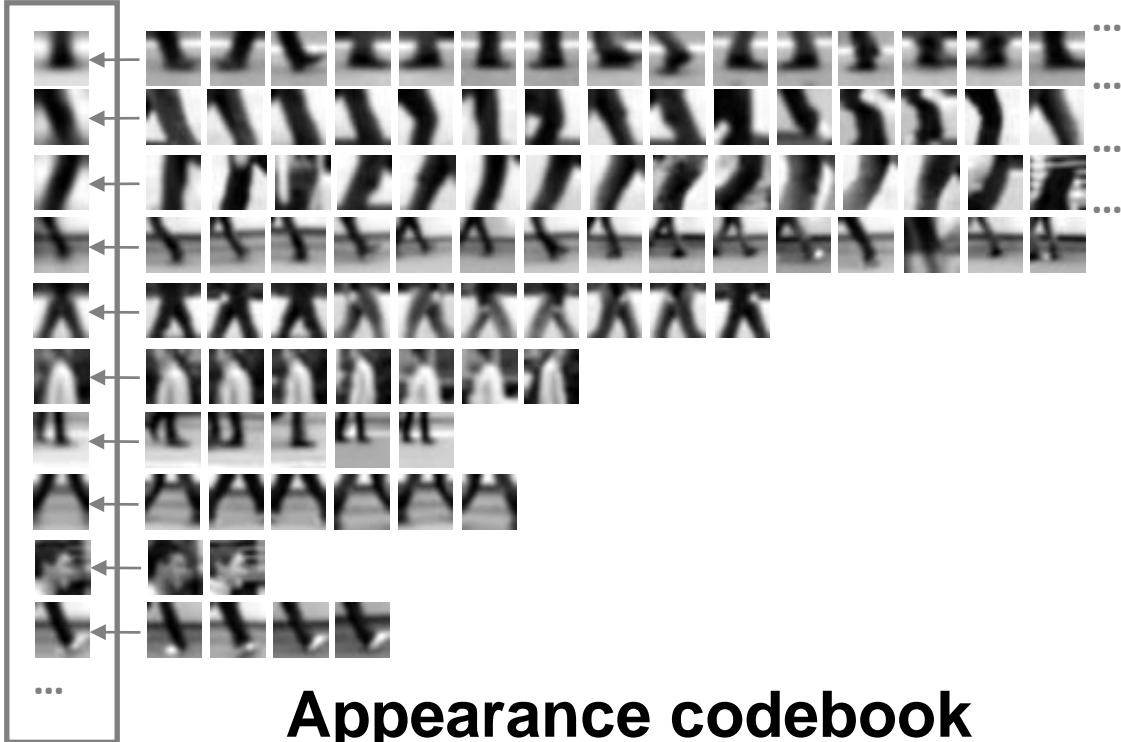
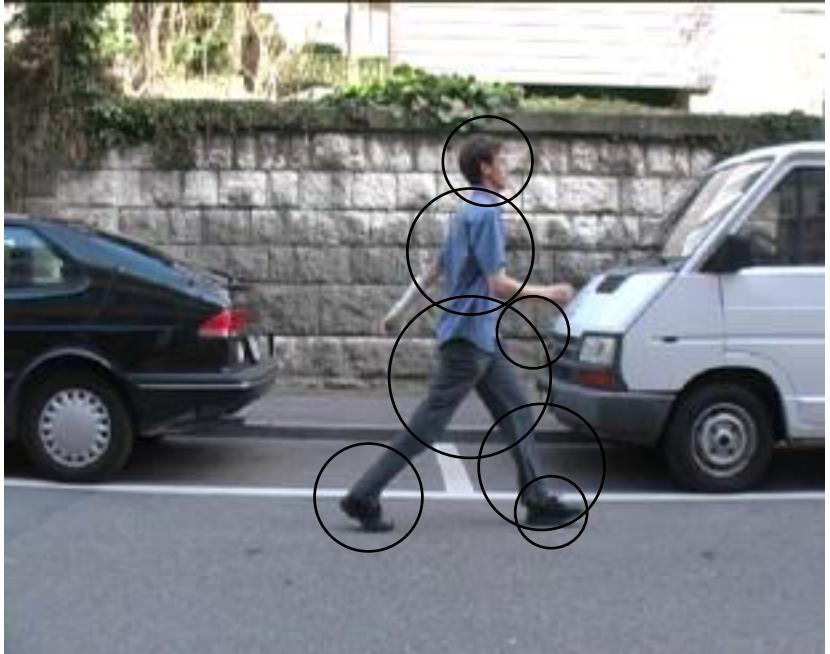
2. Learning the visual vocabulary

- From what data should I learn the dictionary?
 - Dictionary can be learned on separate training set
 - Provided the training set is sufficiently representative, the dictionary will be “universal”

Example Dictionary



Another Dictionary



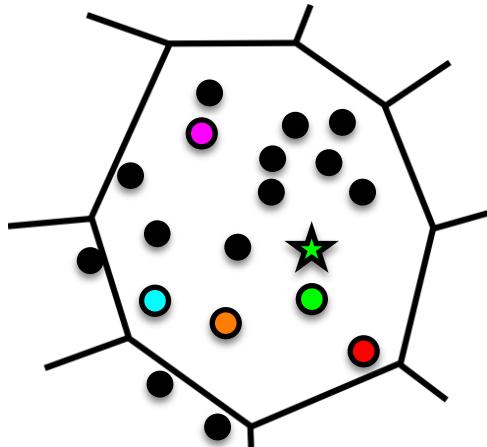
Appearance codebook

Source: B. Leibe

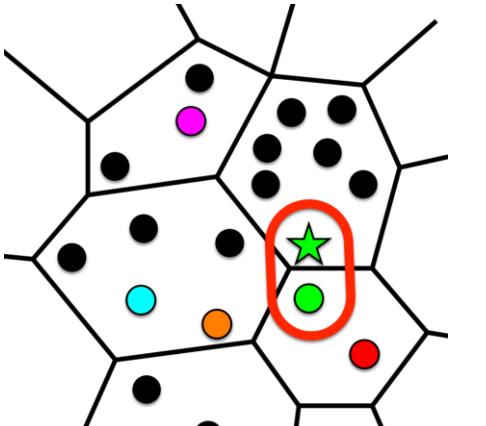
Visual Vocabularies: Issues



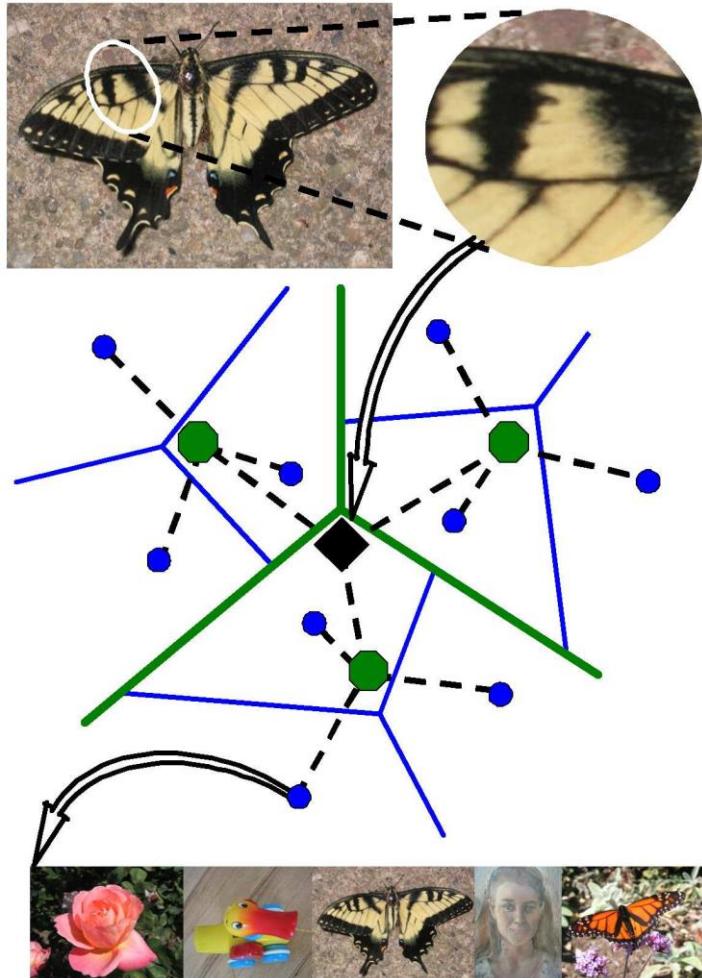
- How to choose vocabulary size?
 - Too small: visual words not representative of all patches
 - Too large: quantization artifacts, overfitting
- Computational efficiency
 - Vocabulary trees
(Nister & Stewenius, 2006)



Smaller Vocabulary

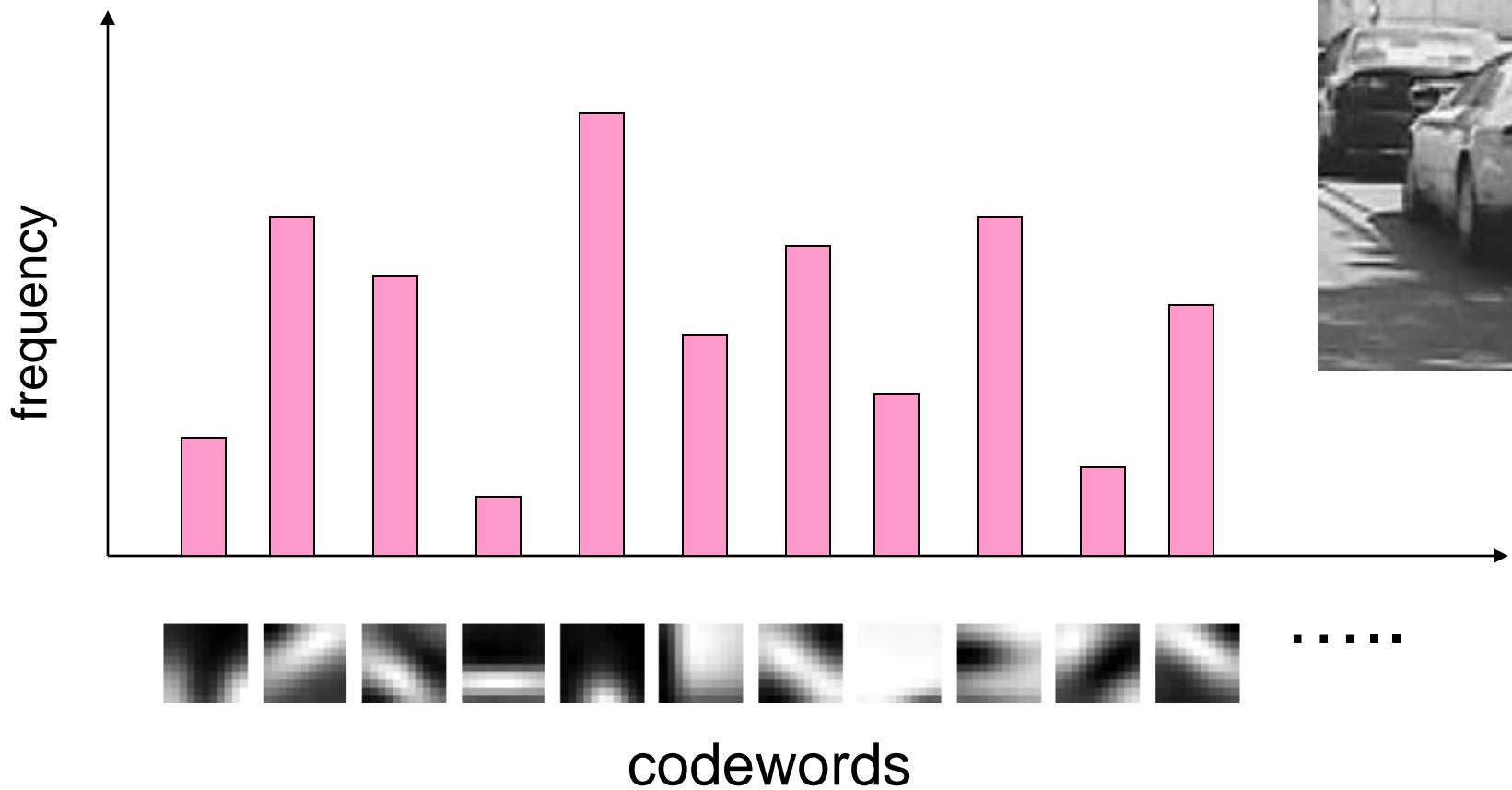


Match missed
due to quantization!

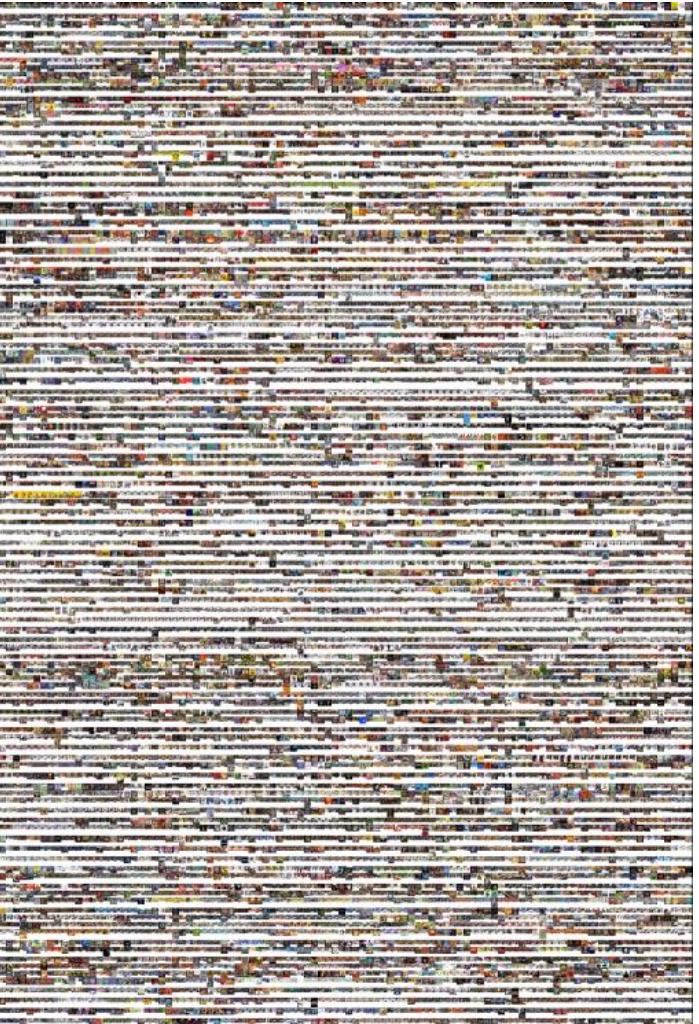


3. Image Representation

- Histogram: count the number of visual word occurrences



Large-scale Image Search



11,400 images of game covers
(Caltech games dataset)

- Bag-of-words models have been useful in matching an image to a large database of object *instances*



how do I find this image in the database?

Large-scale Image Search



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



Weighting the Words

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

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

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



TF-IDF Weighting

- Instead of computing a regular histogram distance, we'll weight each word by it's *inverse document frequency*
- inverse document frequency (IDF) of word j =

$$\log \frac{\text{number of documents}}{\text{number of documents in which } j \text{ appears}}$$



TF-IDF Weighting

- To compute the value of bin j in image I :

term frequency of j in I × inverse document frequency of j



Inverted File

- Each image has ~1,000 features
- We have ~100,000 visual words
 - each histogram is extremely sparse (mostly zeros)
- Inverted file
 - mapping from words to documents

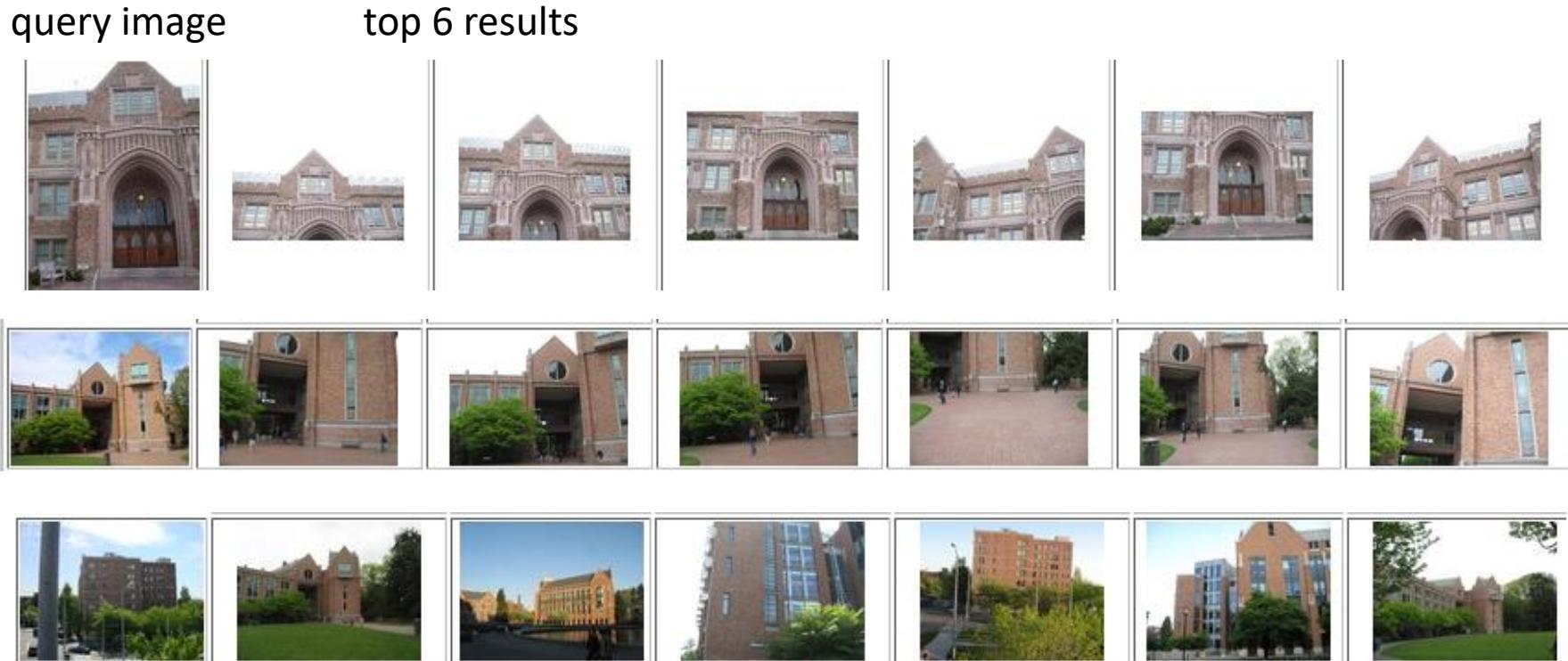
```
"a":      {2}  
"banana": {2}  
"is":     {0, 1, 2}  
"it":     {0, 1, 2}  
"what":   {0, 1}
```



Inverted File

- Can quickly use the inverted file to compute similarity between a new image and all the images in the database
 - Only consider database images whose bins overlap the query image

Large-scale image search



- Cons:
 - not as accurate as per-image-pair feature matching
 - performance degrades as the database grows

Large-scale Image Search

- Pros:
 - Works well for CD covers, movie posters
 - Real-time performance possible



real-time retrieval from a database of 40,000 CD covers

Scalable Recognition with a Vocabulary Tree

David Nistér and Henrik Stewénius

Center for Visualization and Virtual Environments
Department of Computer Science, University of Kentucky

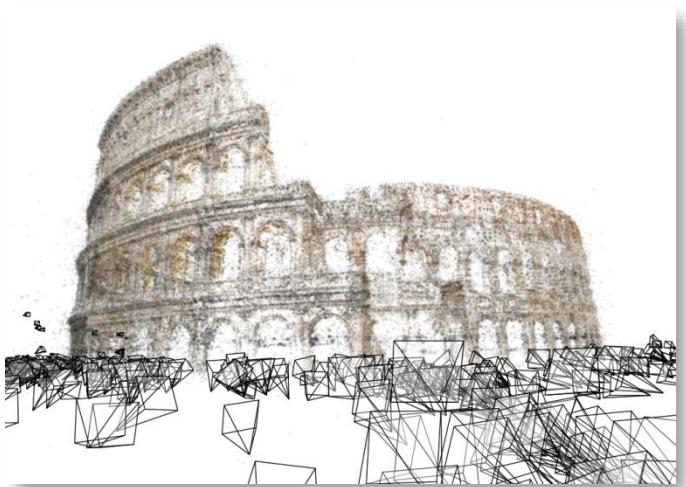
<http://www.vis.uky.edu/~dnister/> <http://www.vis.uky.edu/~stewe/>

[CVPR 2006]

Large-scale image matching

Turn 1,000,000 images of Rome...

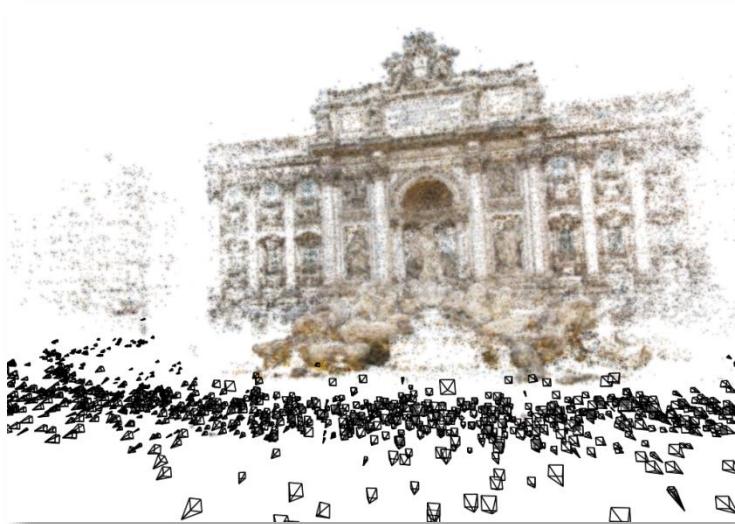
...into 3D models



Colosseum



St. Peter's Basilica



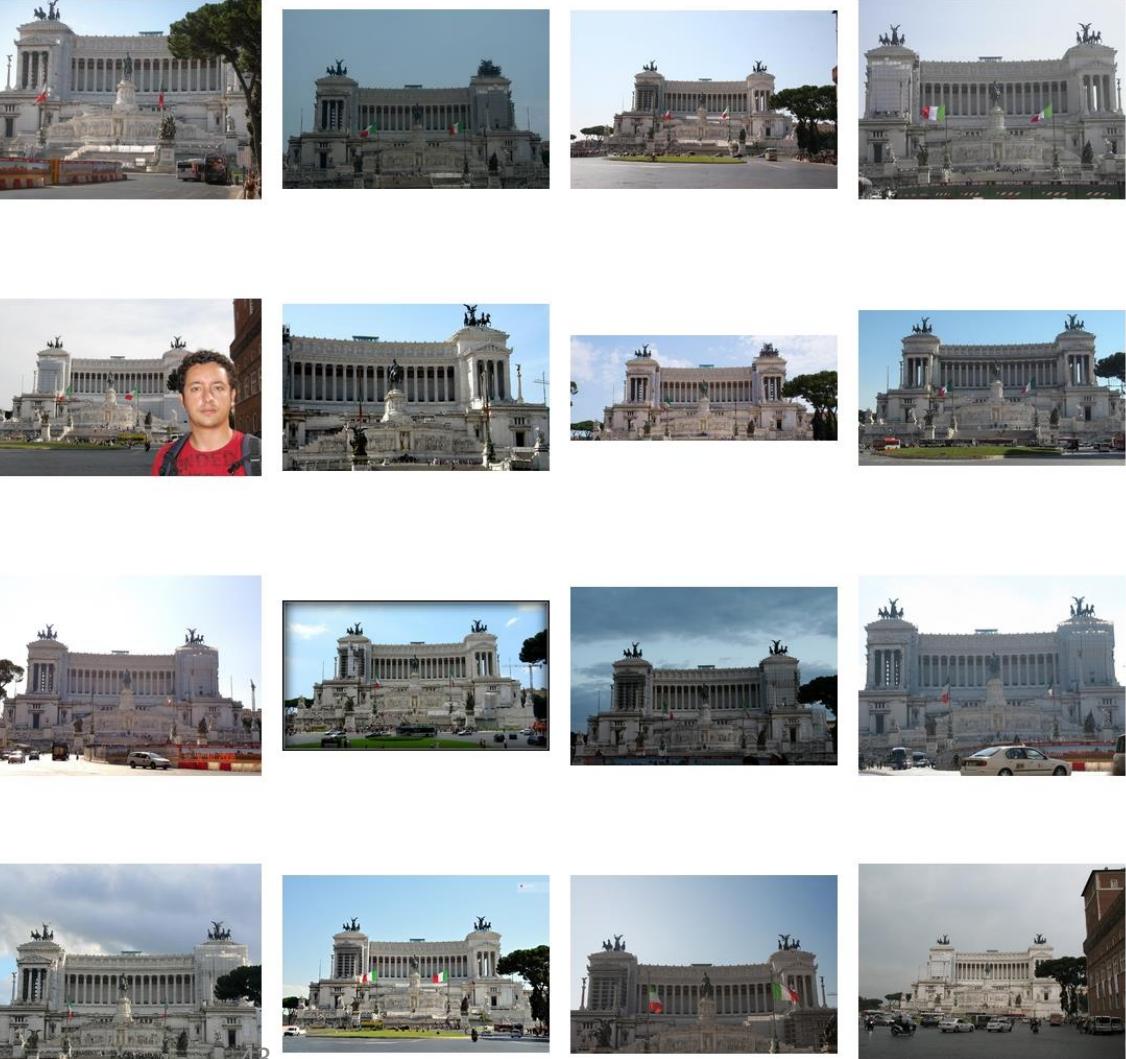
Trevi Fountain



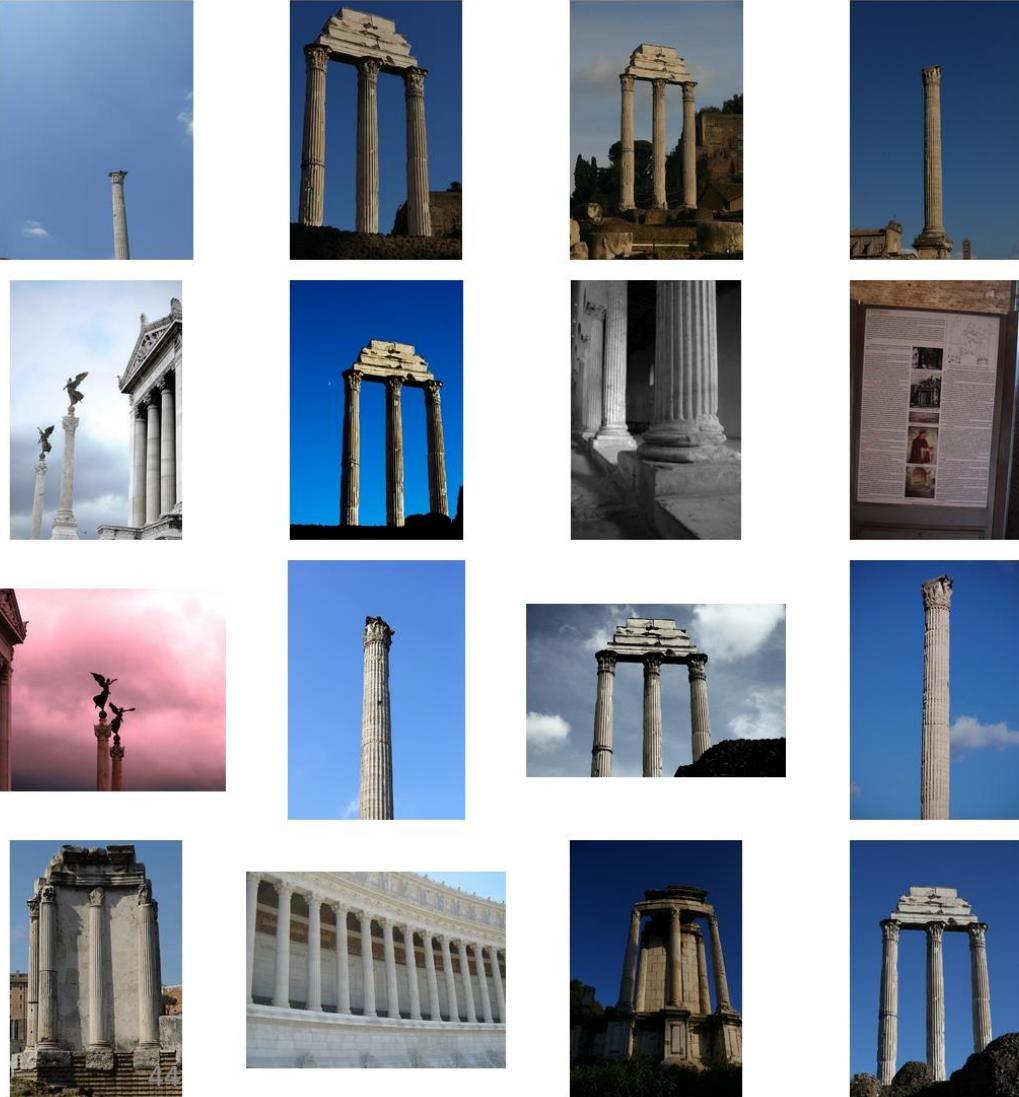
Large-scale image matching

- How can we match 1,000,000 images to each other?
- Brute force approach: 500,000,000,000 pairs
 - won't scale
- Better approach: use bag-of-words technique to find *likely* matches
- For each image, find the top M scoring other images, do detailed SIFT matching with those

Example bag-of-words matches



Example bag-of-words matches





Matching Statistics

Dataset	Size	Matches possible	Matches Tried	Matches Found	Time
Dubrovnik	58K	1.6 Billion	2.6M	0.5M	5 hrs
Rome	150K	11.2 Billion	8.8M	2.7M	13 hrs
Venice	250K	31.2 Billion	35.5M	6.2M	27 hrs



NetVLAD: CNN architecture for weakly supervised place recognition

Relja Arandjelović
INRIA *

Petr Gronat
INRIA *

Akihiko Torii
Tokyo Tech †

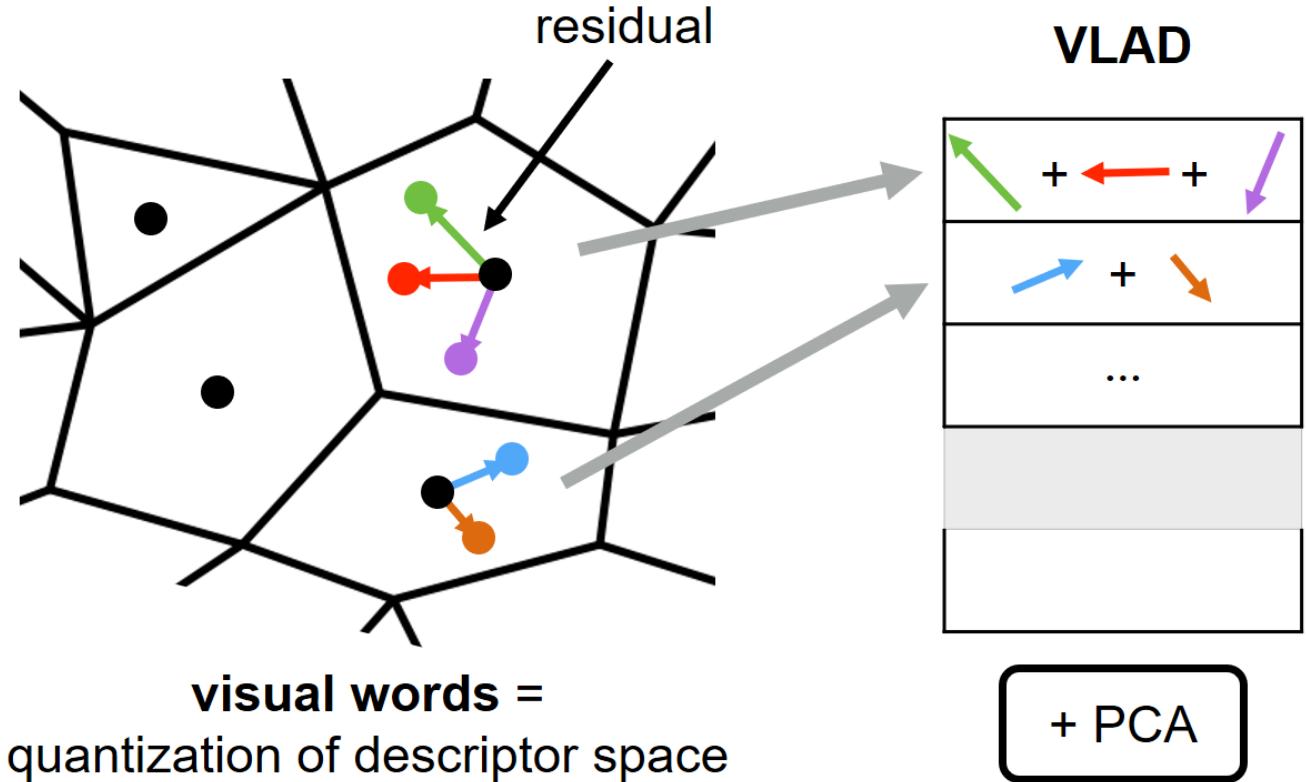
Tomas Pajdla
CTU in Prague ‡

Josef Sivic
INRIA *

Vector of Locally Aggregated Descriptors (VLAD)

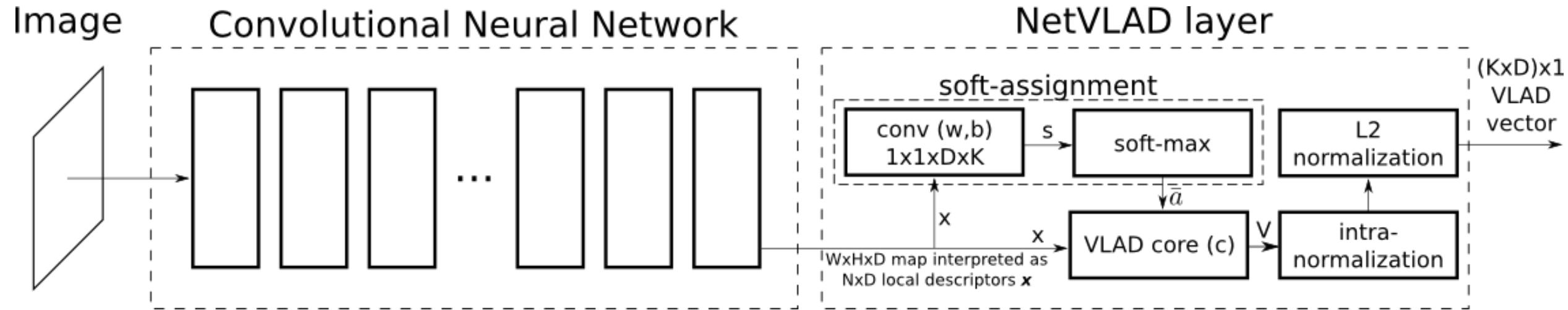


image with **local features**



$$V(j, k) = \sum_{i=1}^N \alpha_k(x_i)(x_i(j) - c_k(j))$$

NetVLAD



$$V(j, k) = \sum_{i=1}^N \frac{e^{w_k^T x_i + b_k}}{\sum_{k'} e^{w_{k'}^T x_i + b_{k'}}} (x_i(j) - c_k(j))$$

[Arandjelović, Gronat, Torii, Pajdla, Sivic, NetVLAD: CNN architecture for weakly supervised place recognition. CVPR 2016]