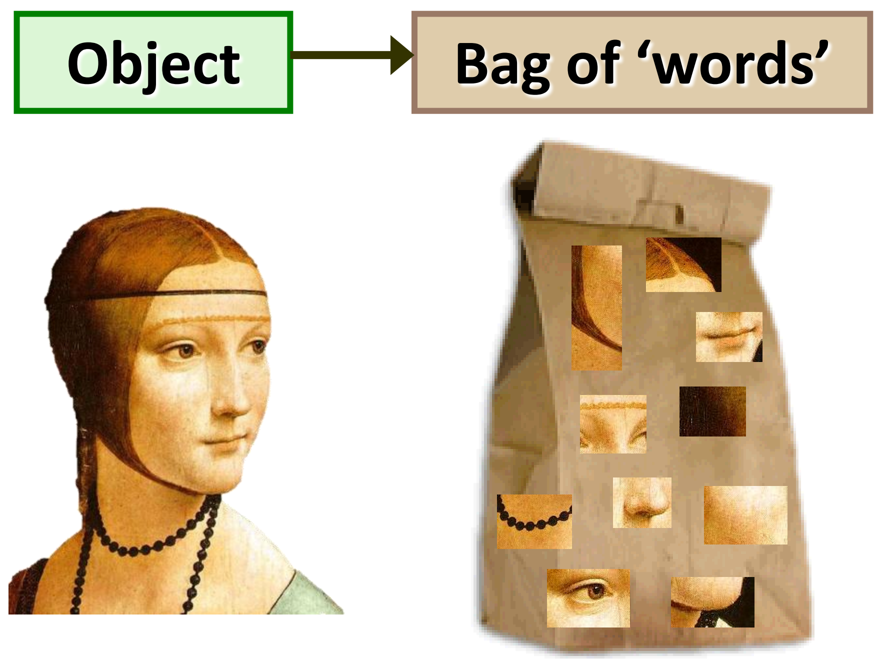
CS4670: Computer Vision

Noah Snavely

Lecture 28: Bag-of-words models

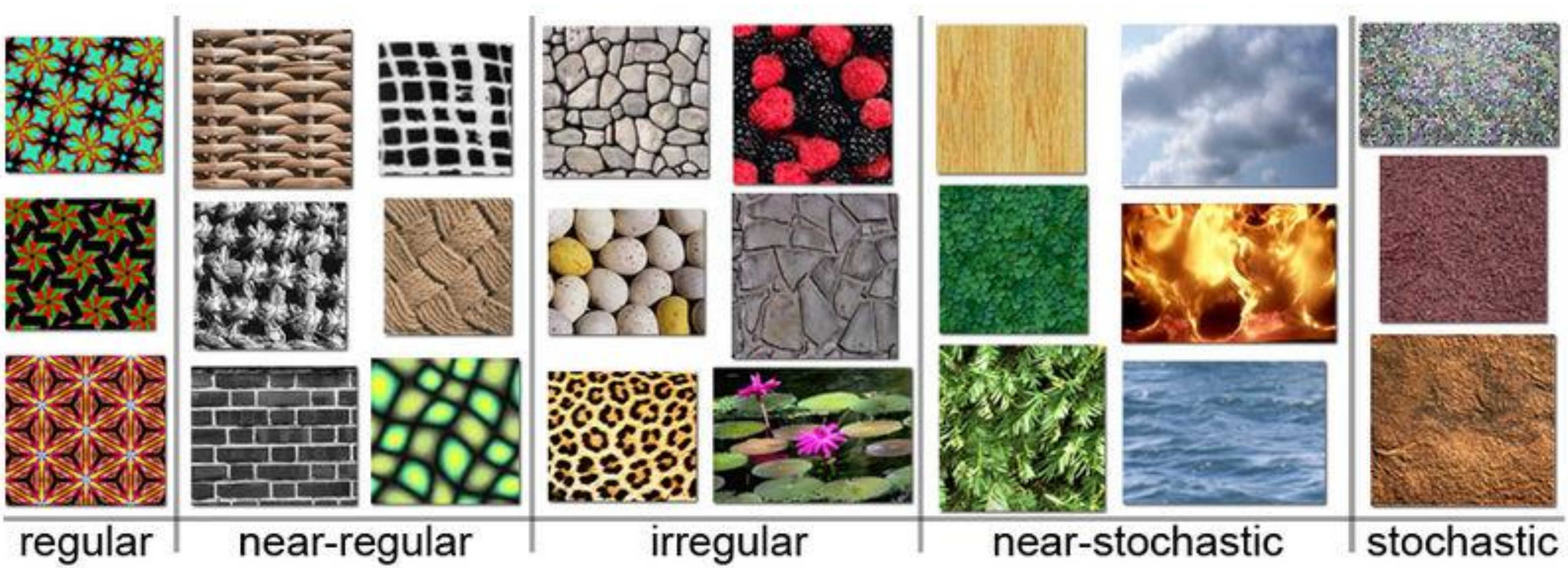


Bag of Words Models

Adapted from slides by Rob Fergus and Svetlana Lazebnik

Object → **Bag of ‘words’**

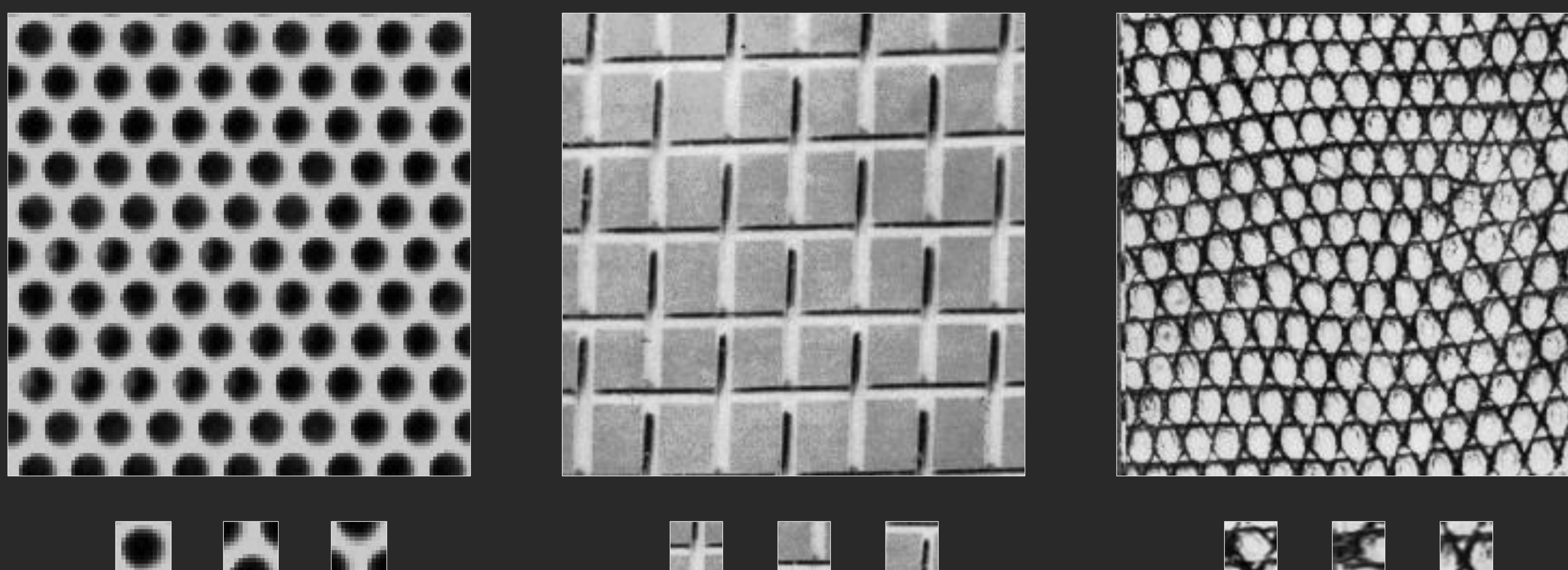
Origin 1: Texture Recognition



Example textures (from Wikipedia)

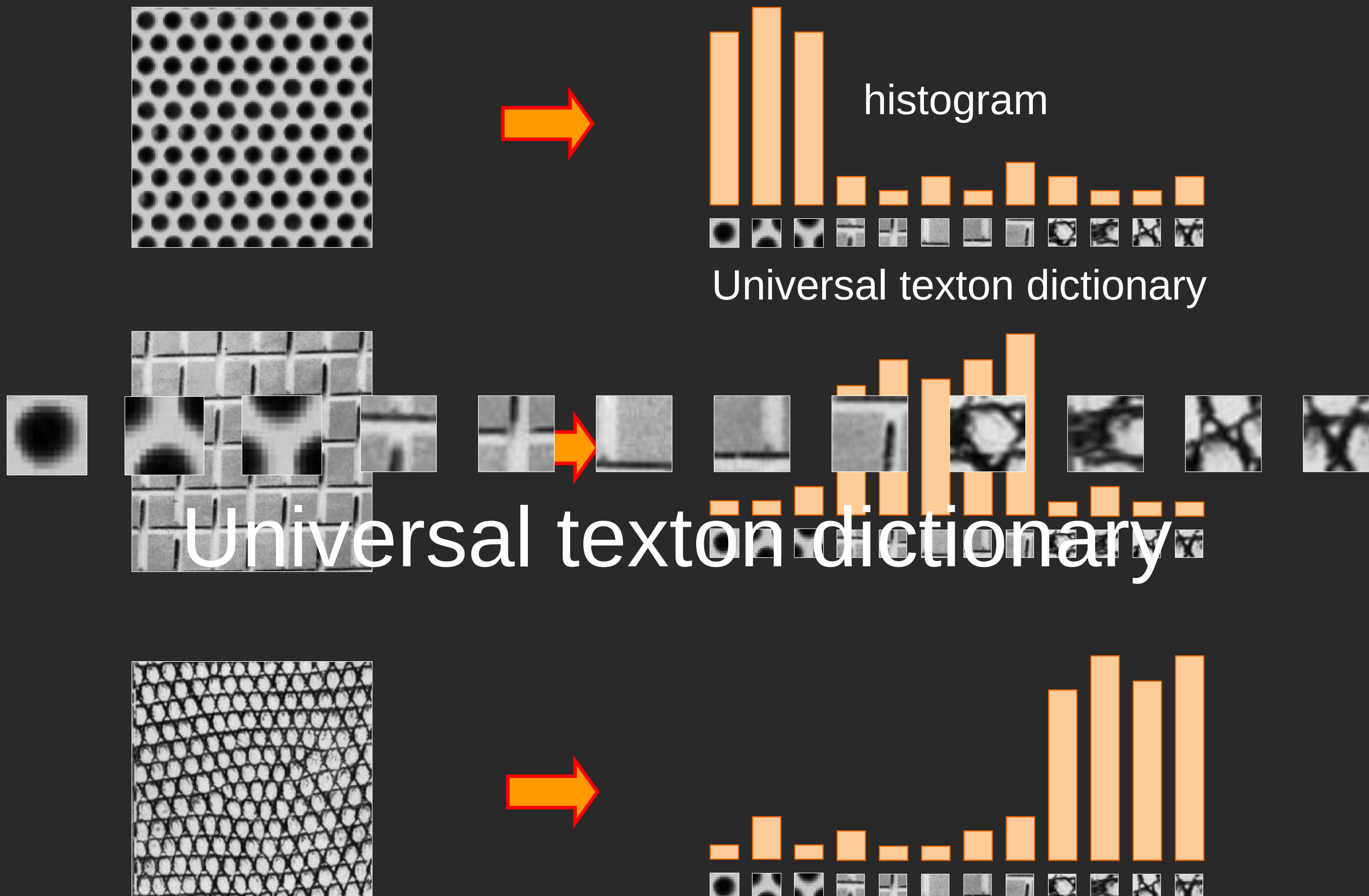
Origin 1: Texture recognition

- Texture is characterized by the repetition of basic elements or *textons*
- For stochastic textures, it is the identity of the textons, not their spatial arrangement, that matters



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

Origin 1: Texture recognition



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

Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

US Presidential Speeches Tag Cloud
`http://chir.ag/phernalia/preztags/`

Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

US Presidential Speeches Tag Cloud
`http://chir.ag/phernalia/preztags/`

Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

US Presidential Speeches Tag Cloud
`http://chir.ag/phernalia/preztags/`

Bags of features for object recognition

face, flowers, building

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

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

Bags of features for object recognition

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

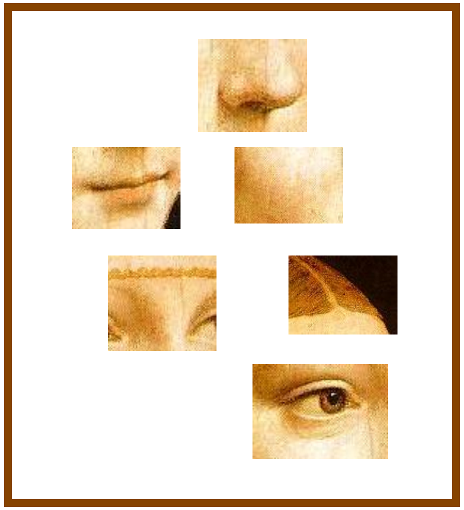
Bag of features

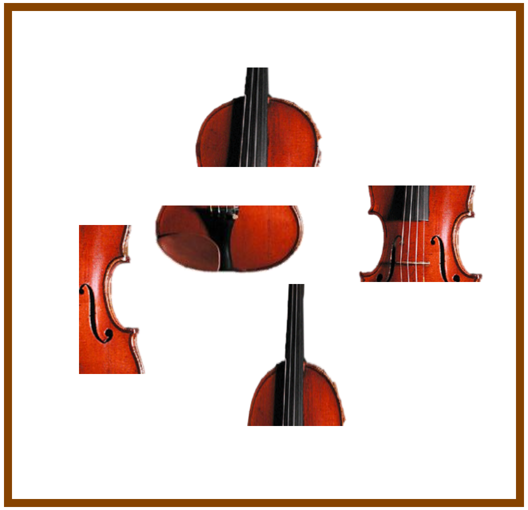
- First, take a bunch of images, extract features, and build up a “dictionary” or “visual vocabulary” – a list of common features

- Given a new image, extract features and build a histogram – for each feature, find the closest visual word in the dictionary

Bag of features: outline

1. Extract features





Bag of features: outline

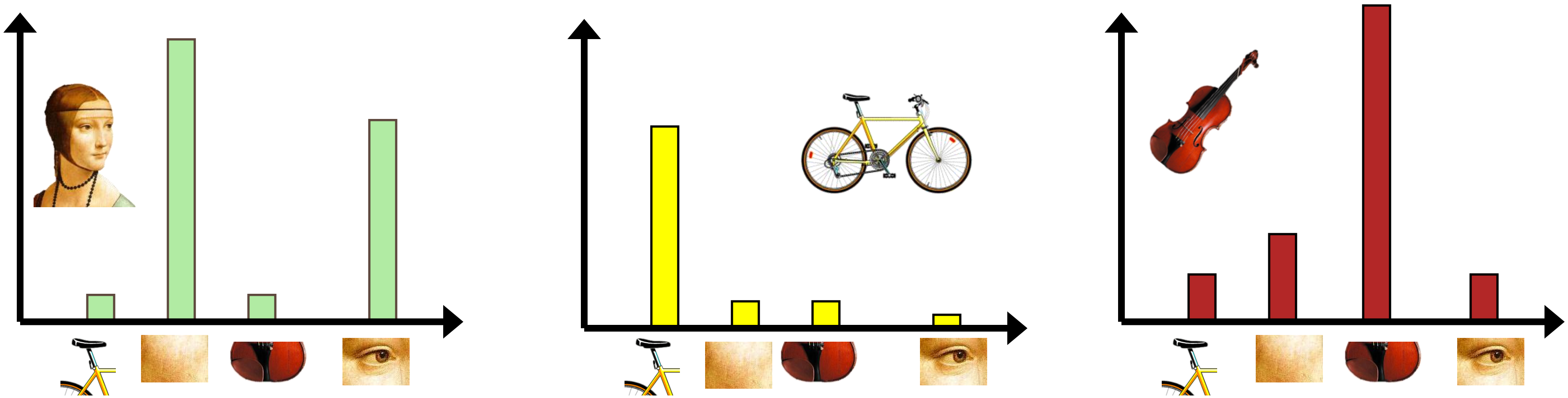
1. Extract features
2. Learn “visual vocabulary”

Bag of features: outline

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

Bag of features: outline

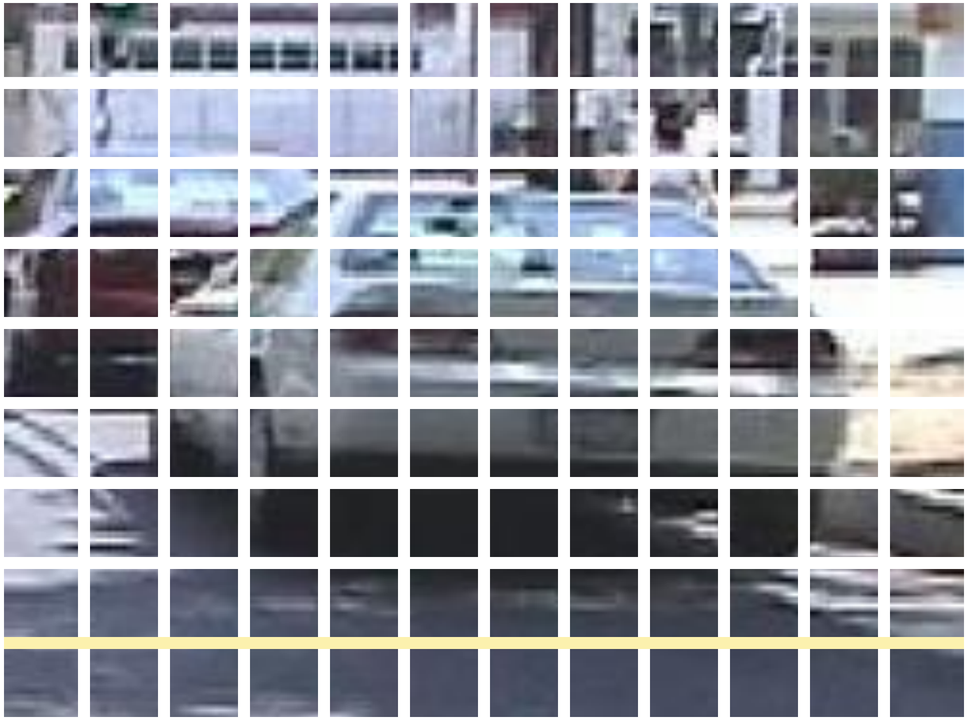
1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary
4. Represent images by frequencies of “visual words”



1. Feature extraction

Regular grid

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



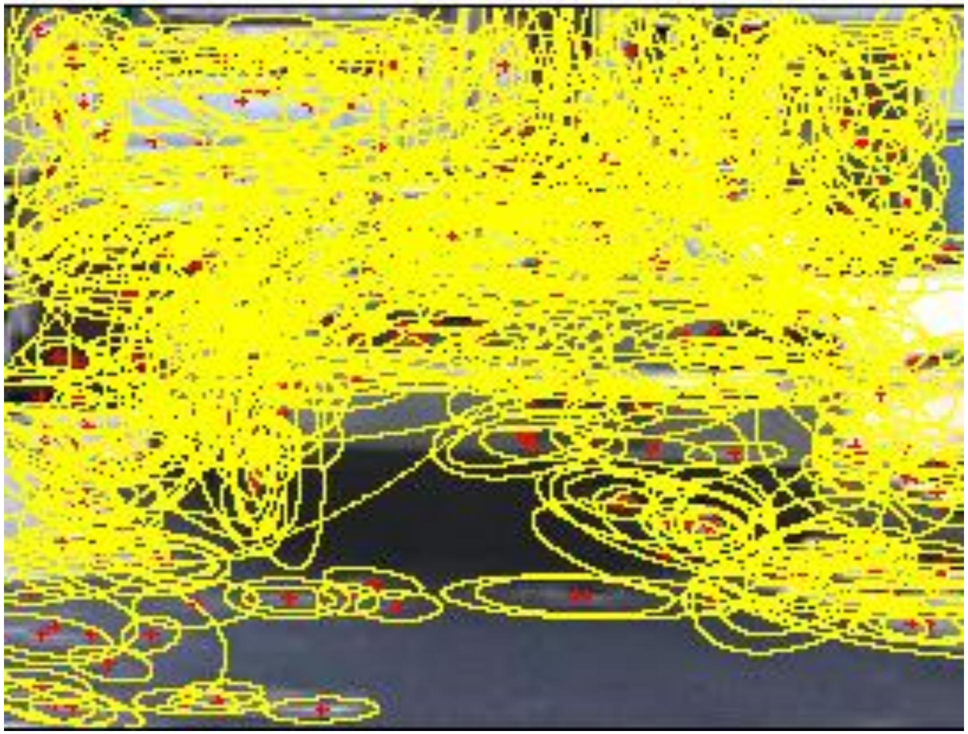
1. Feature extraction

Regular grid

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

Interest point detector

- Csurka et al. 2004
- Fei-Fei & Perona, 2005
- Sivic et al. 2005



1. Feature extraction

Regular grid

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

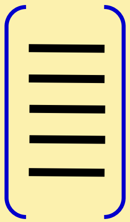
Interest point detector

- Csurka et al. 2004
- Fei-Fei & Perona, 2005
- Sivic et al. 2005

Other methods

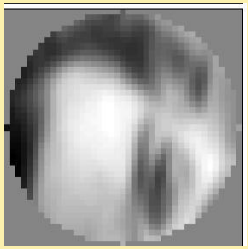
- Random sampling (Vidal-Naquet & Ullman, 2002)
- Segmentation-based patches (Barnard et al. 2003)

1. Feature extraction

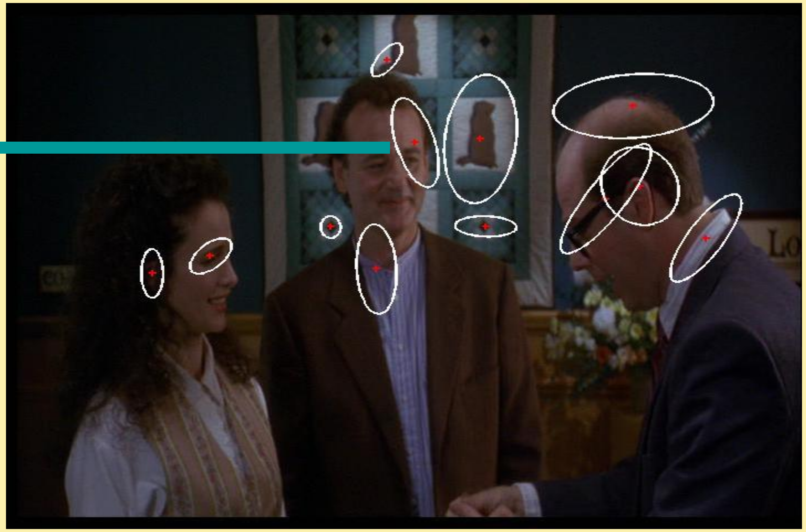


Compute SIFT descriptor

[Lowe’99]



Normalize patch



Detect patches

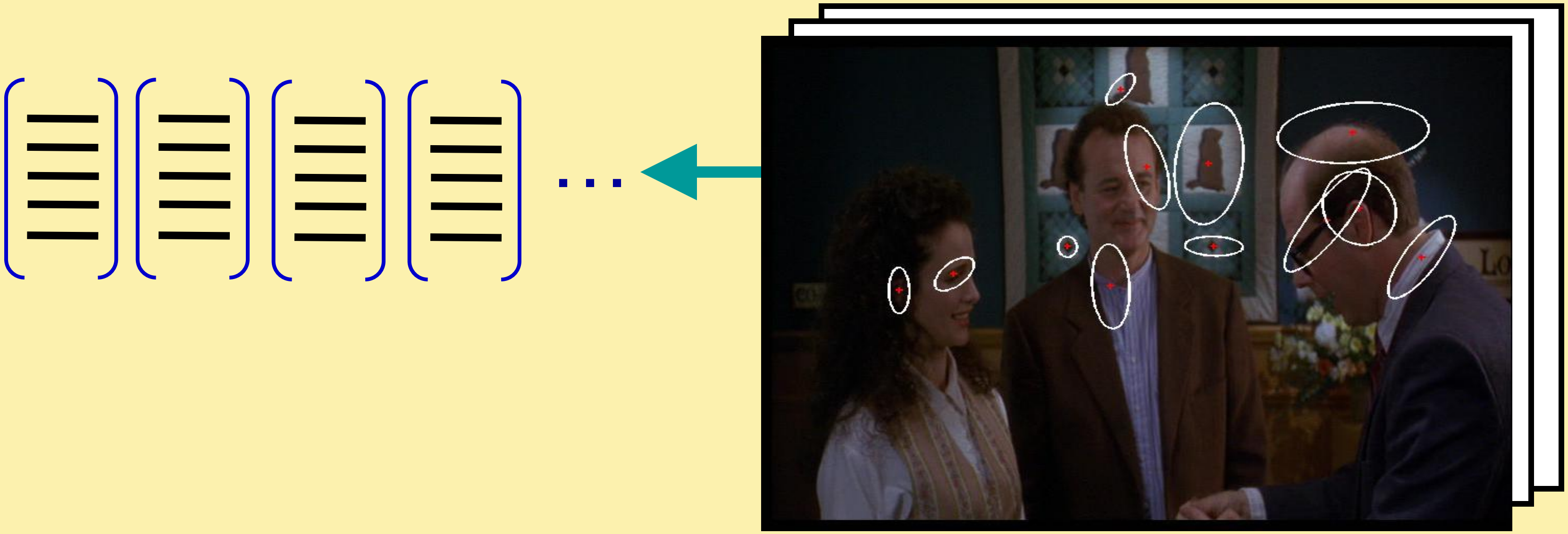
[Mikojaczyk and Schmid ’02]

[Mata, Chum, Urban & Pajdla, ’02]

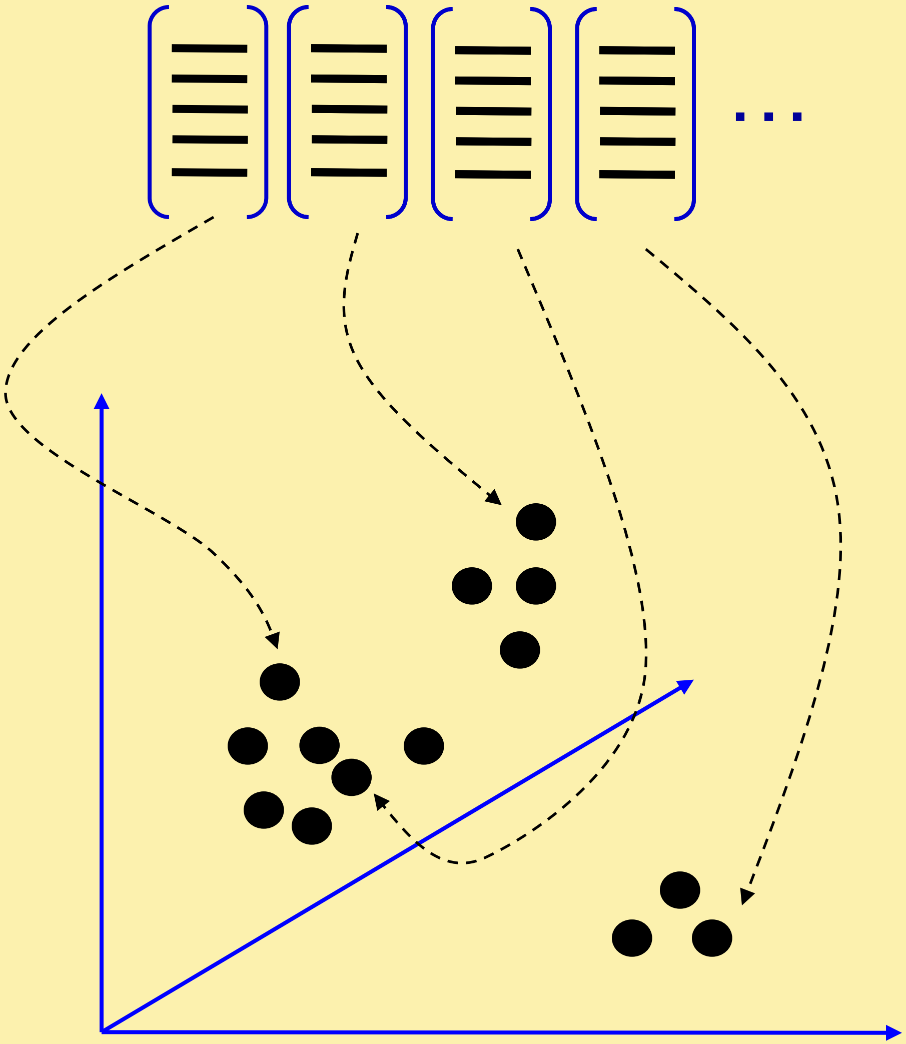
[Sivic & Zisserman, ’03]

Slide credit: Josef Sivic

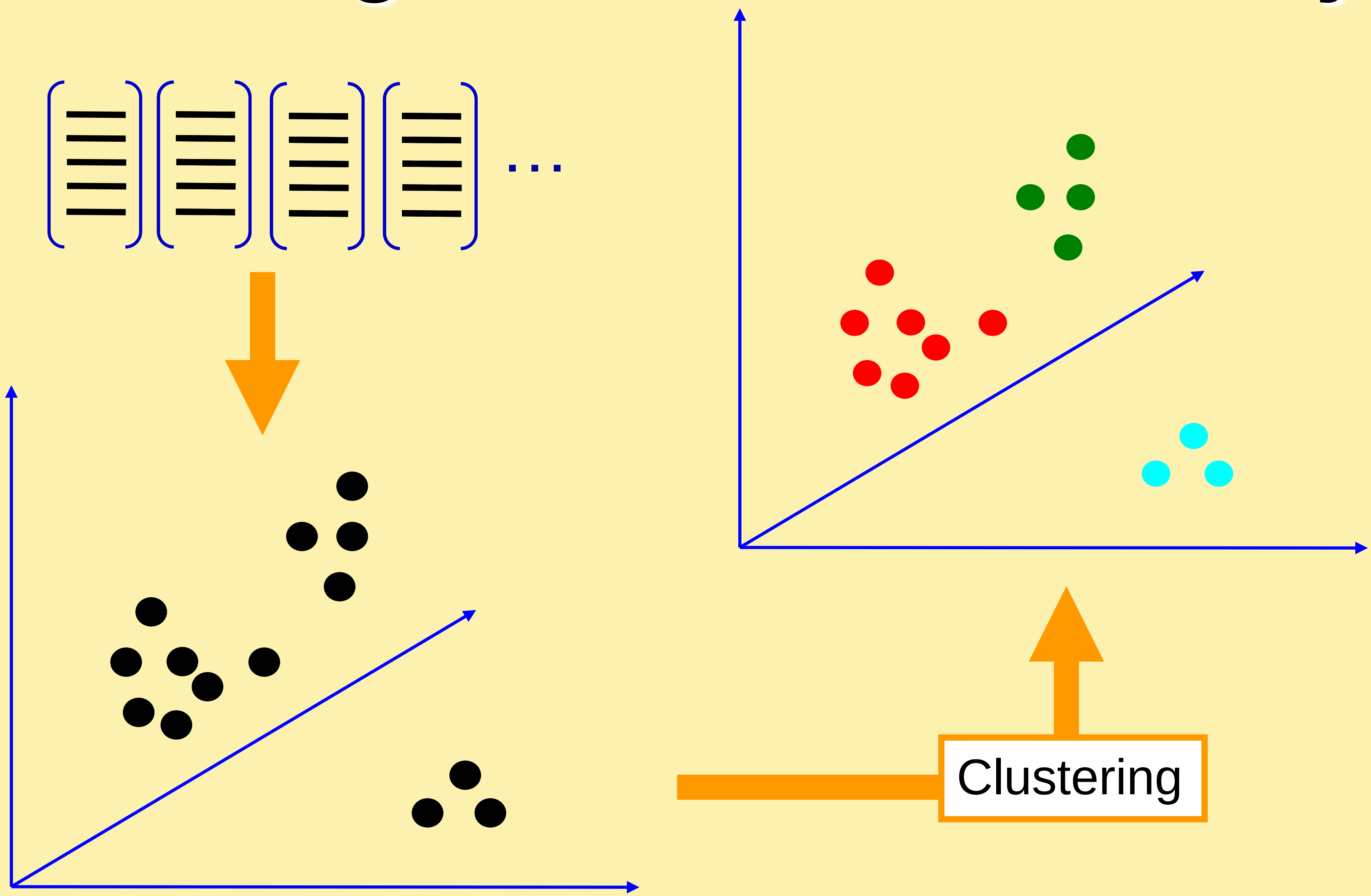
1. Feature extraction



2. Learning the visual vocabulary

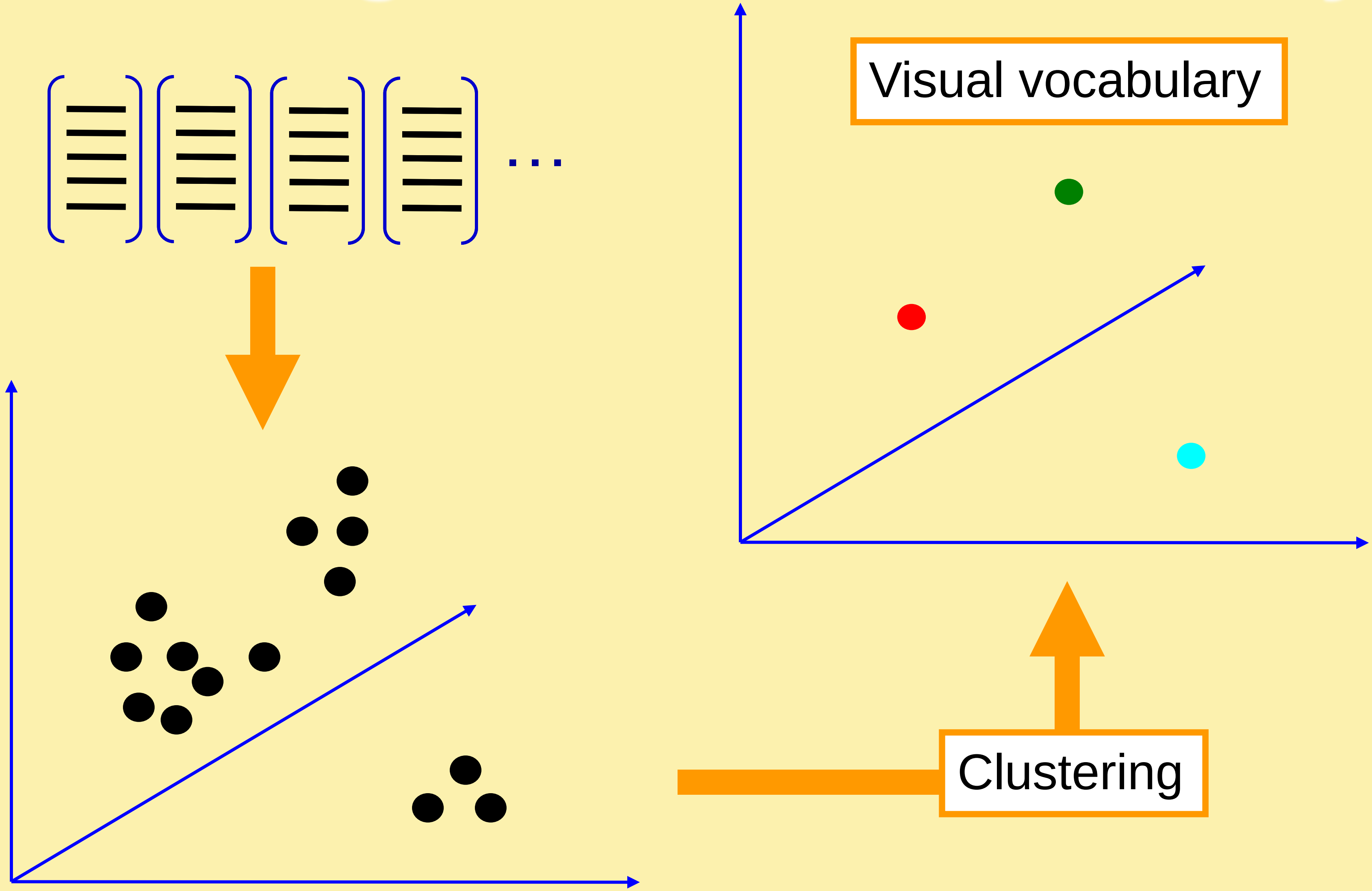


2. Learning the visual vocabulary



Slide credit: Josef Sivic

2. Learning the visual vocabulary



Slide credit: Josef Sivic

K-means clustering

- Want to minimize sum of squared Euclidean distances between points x_i and their nearest cluster centers m_k

$$D(X, M) = \sum_{\text{cluster } k} \sum_{\substack{\text{point } i \text{ in} \\ \text{cluster } k}} (x_i - m_k)^2$$

Algorithm:

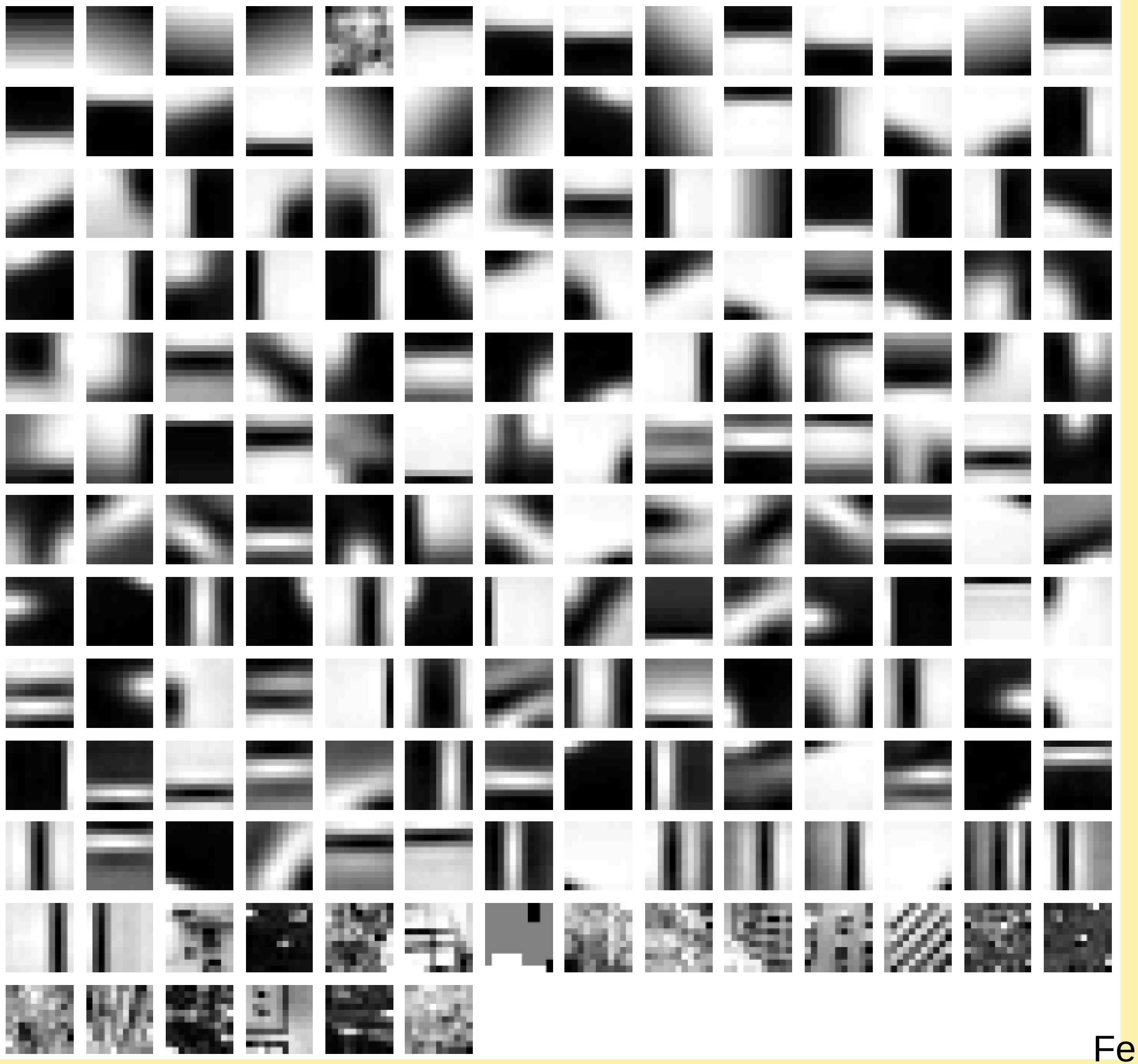
- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each data point to the nearest center
  - Recompute each cluster center as the mean of all points assigned to it

From clustering to vector quantization

- Clustering is a common method for learning a visual vocabulary or codebook
  - Unsupervised learning process
  - Each cluster center produced by k-means becomes a codevector
  - Codebook can be learned on separate training set
  - Provided the training set is sufficiently representative, the codebook will be “universal”

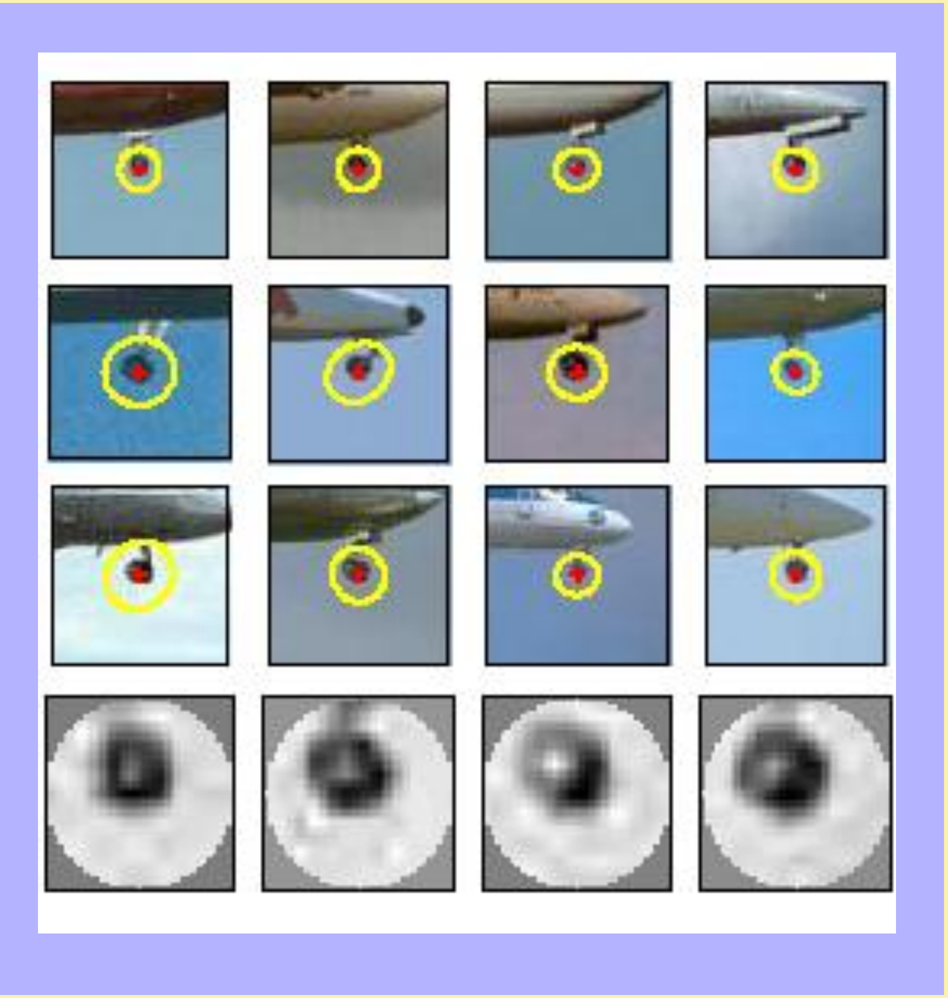
- The codebook is used for quantizing features
  - A *vector quantizer* takes a feature vector and maps it to the index of the nearest codevector in a codebook
  - Codebook = visual vocabulary
  - Codevector = visual word

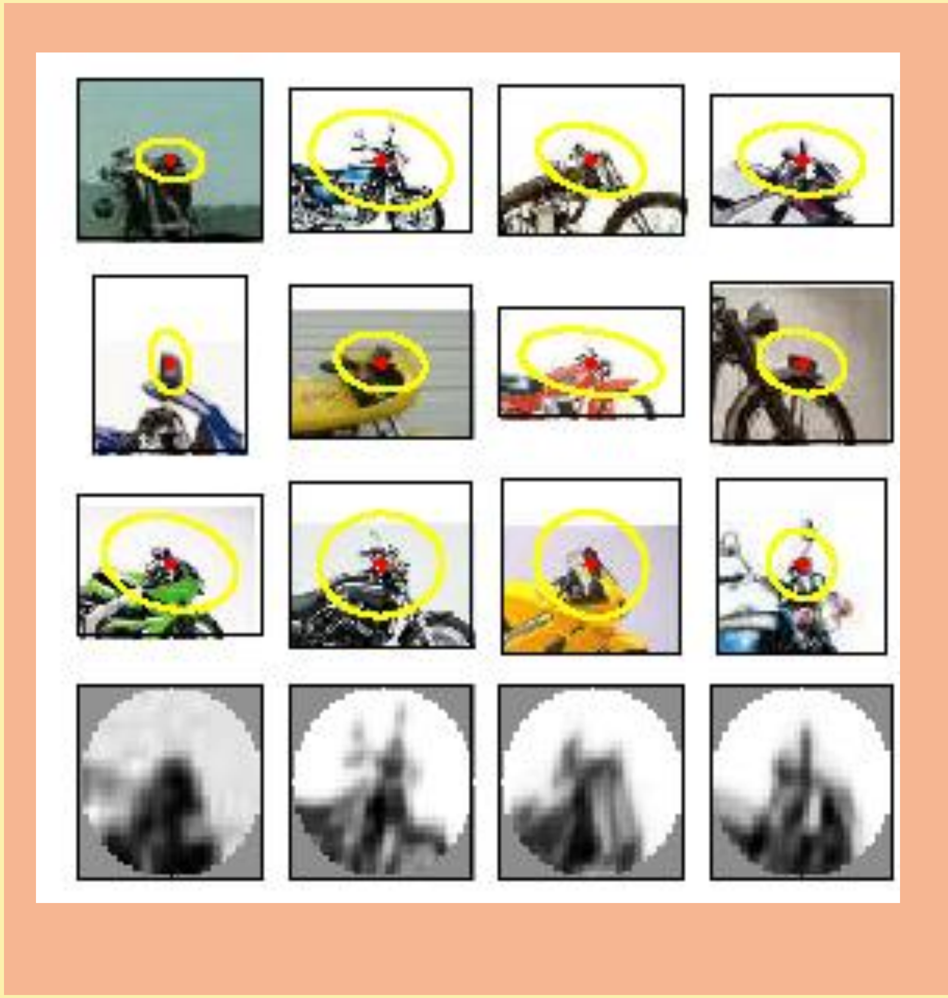
Example visual vocabulary



Fei-Fei et al. 2005

Image patch examples of visual words

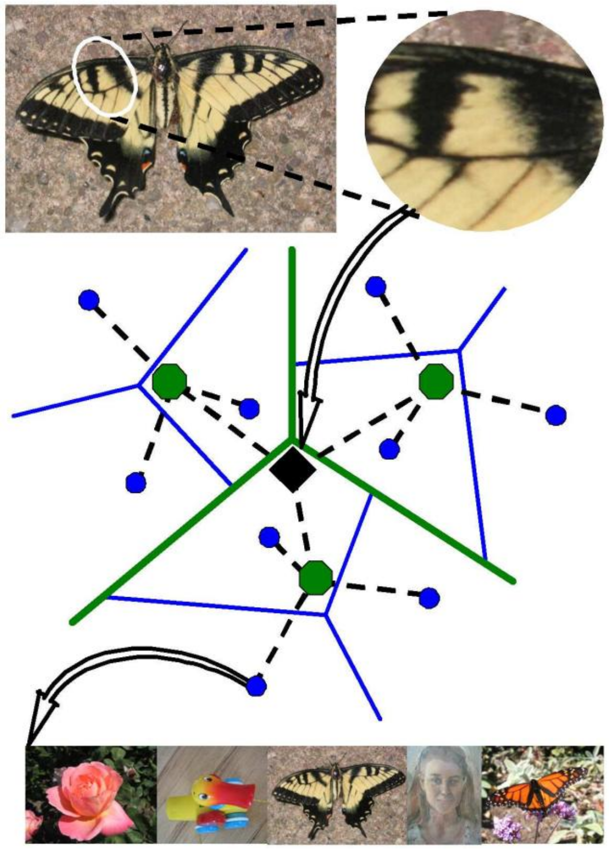




Sivic et al. 2005

Visual vocabularies: Issues

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



3. Image representation

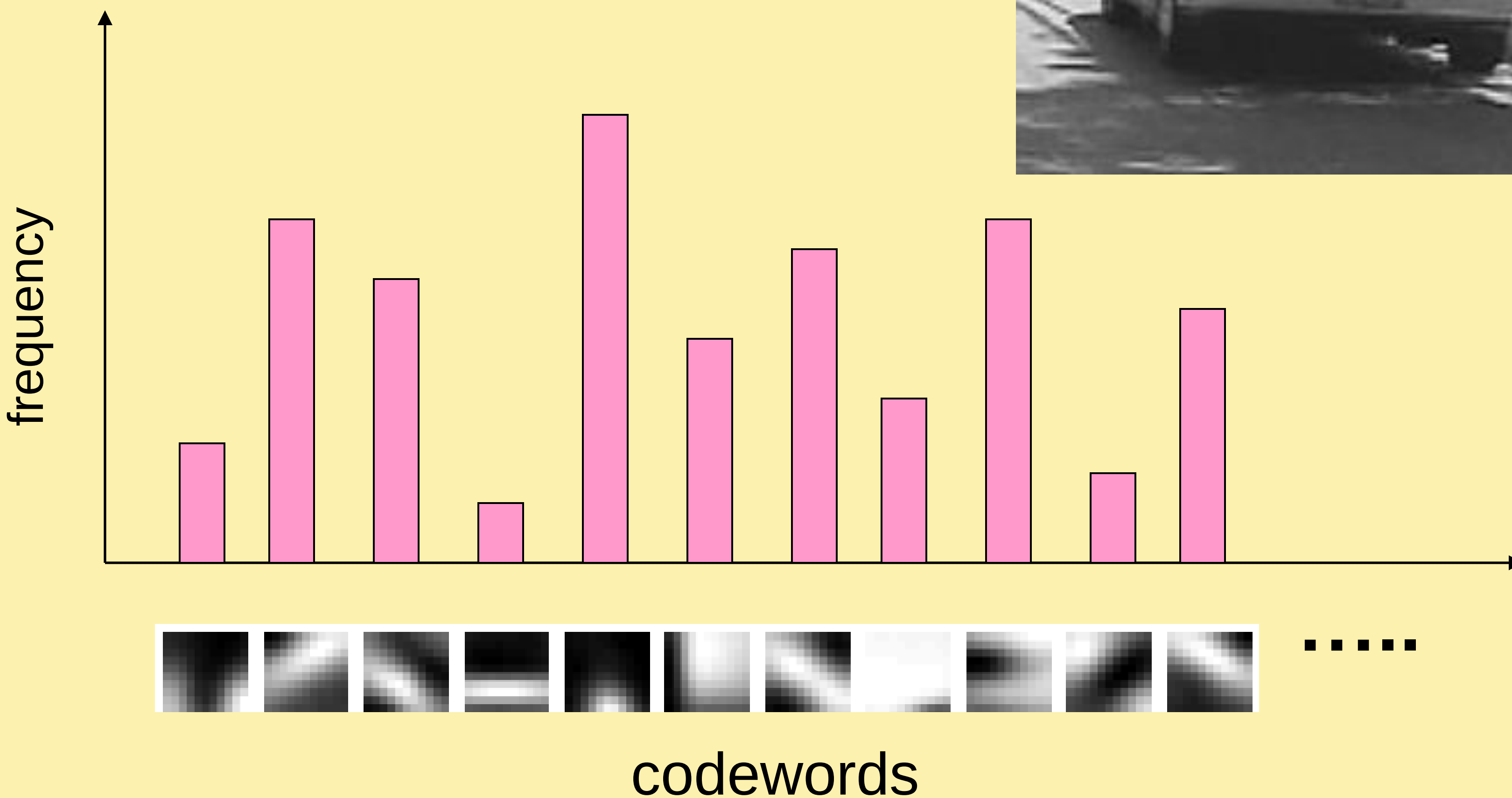
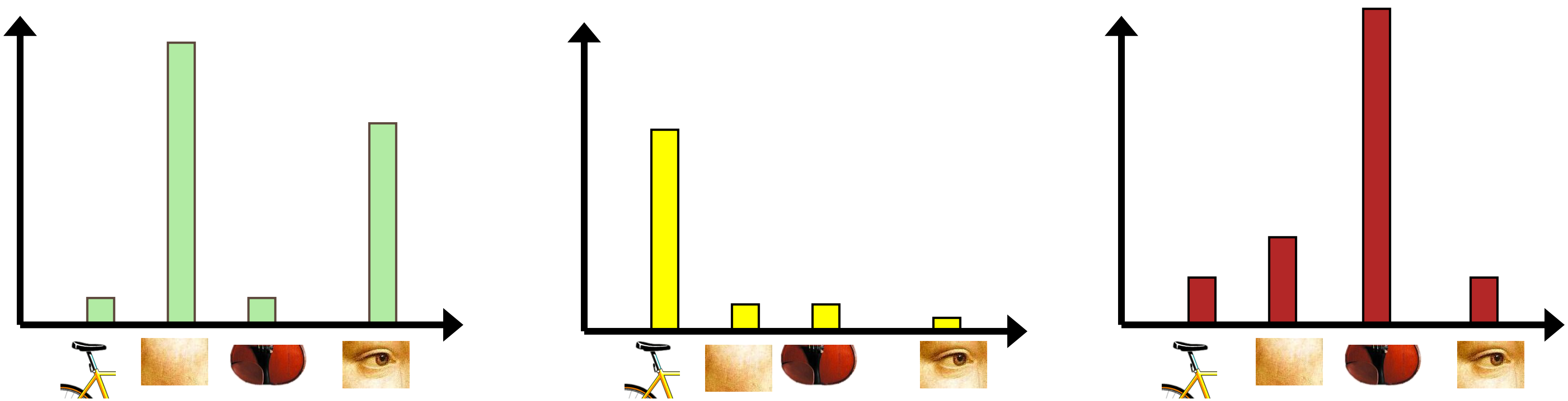


Image classification

- Given the bag-of-features representations of images from different classes, how do we learn a model for distinguishing them?



Uses of BoW representation

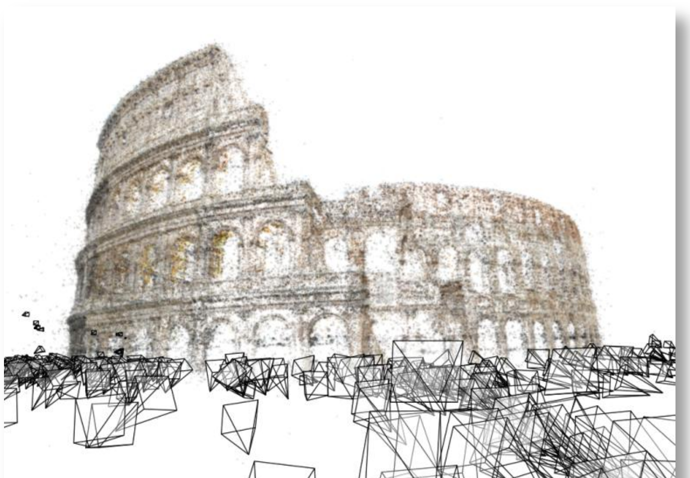
- Treat as feature vector for standard classifier
  - e.g k-nearest neighbors, support vector machine
- Cluster BoW vectors over image collection
  - Discover visual themes

Large-scale image matching

Turn 1,000,000 images of Rome…

…into 3D models

St. Peter's Basilica



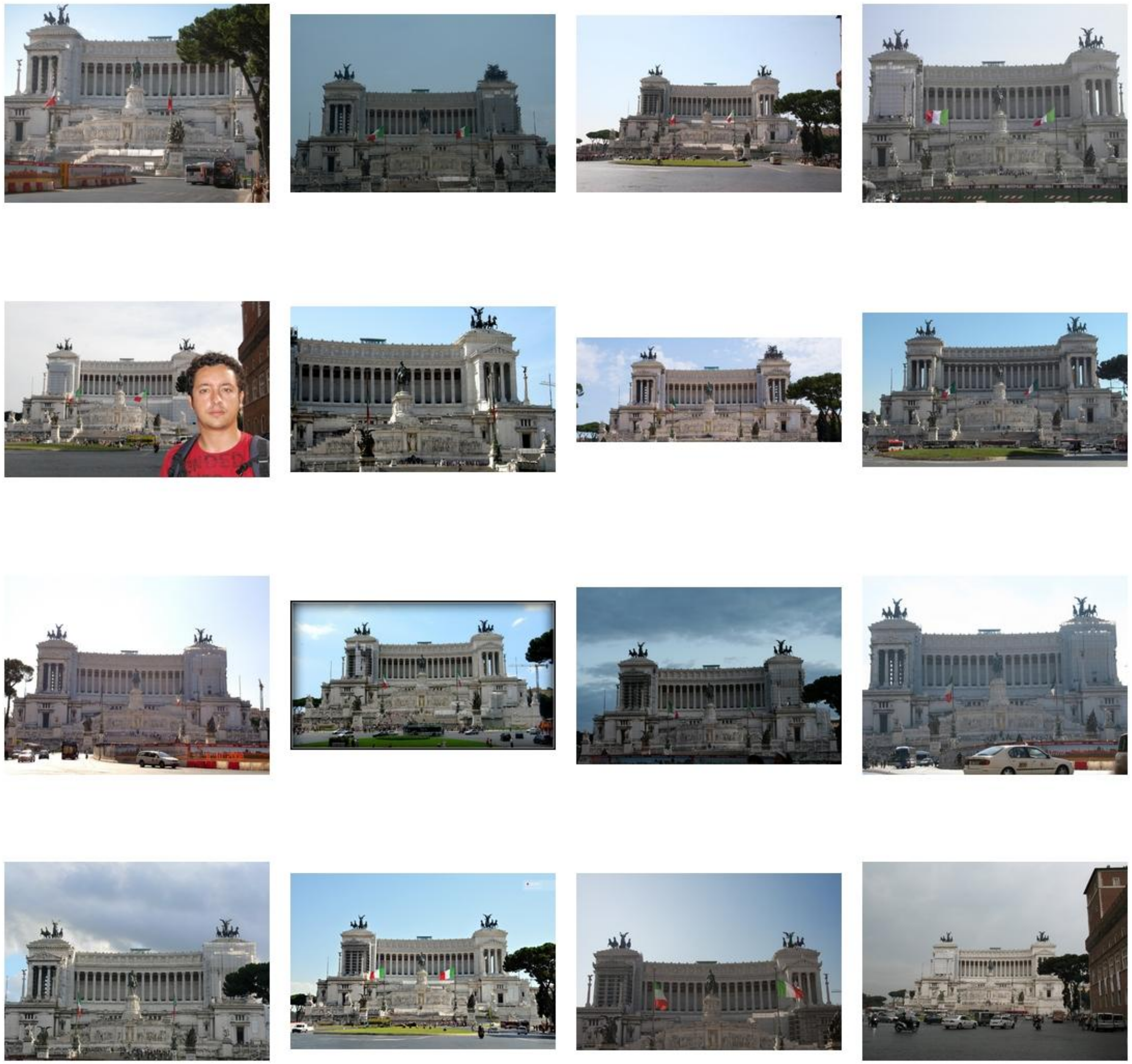
Colosseum

Trevi Fountain

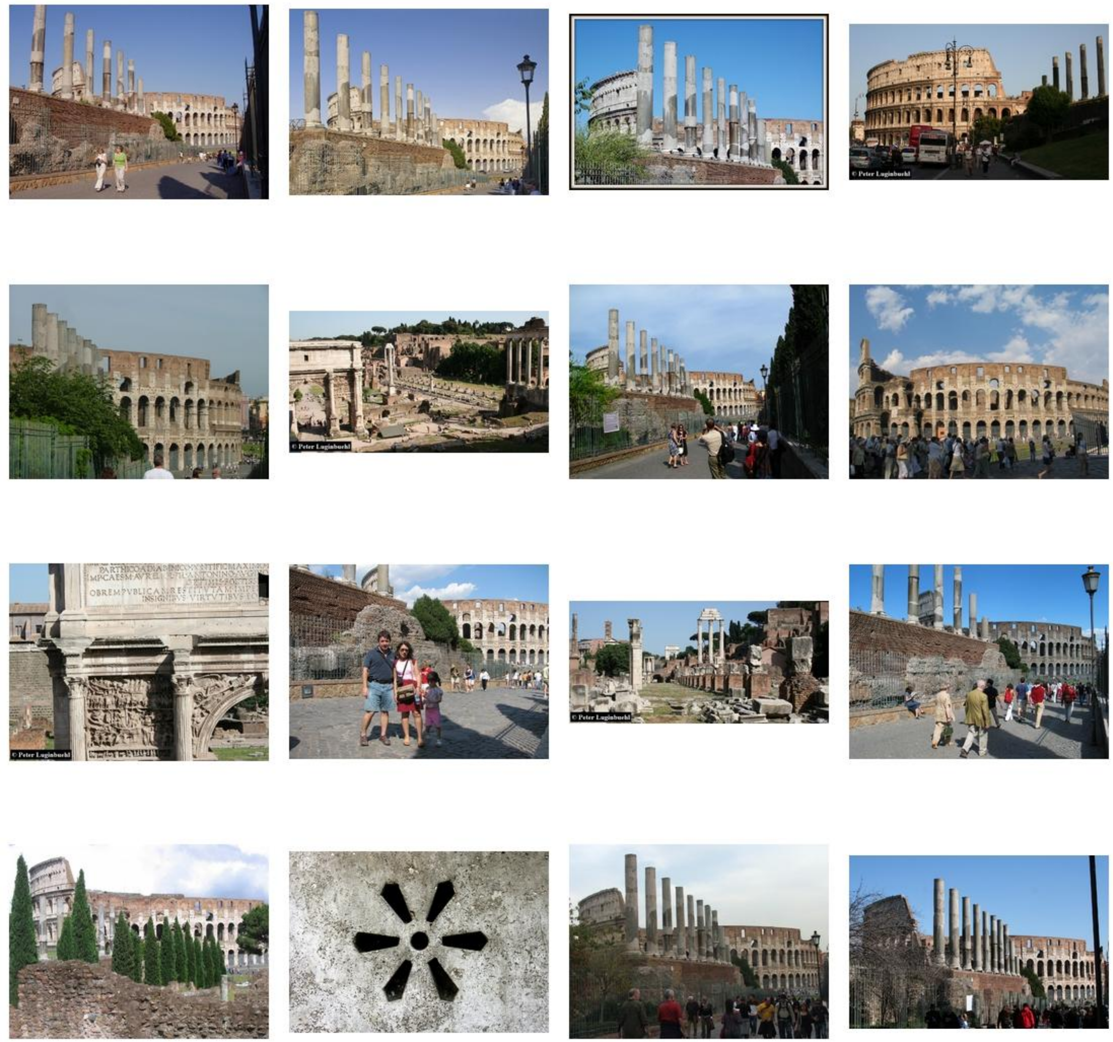
Large-scale image matching

- 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



Example bag-of-words matches