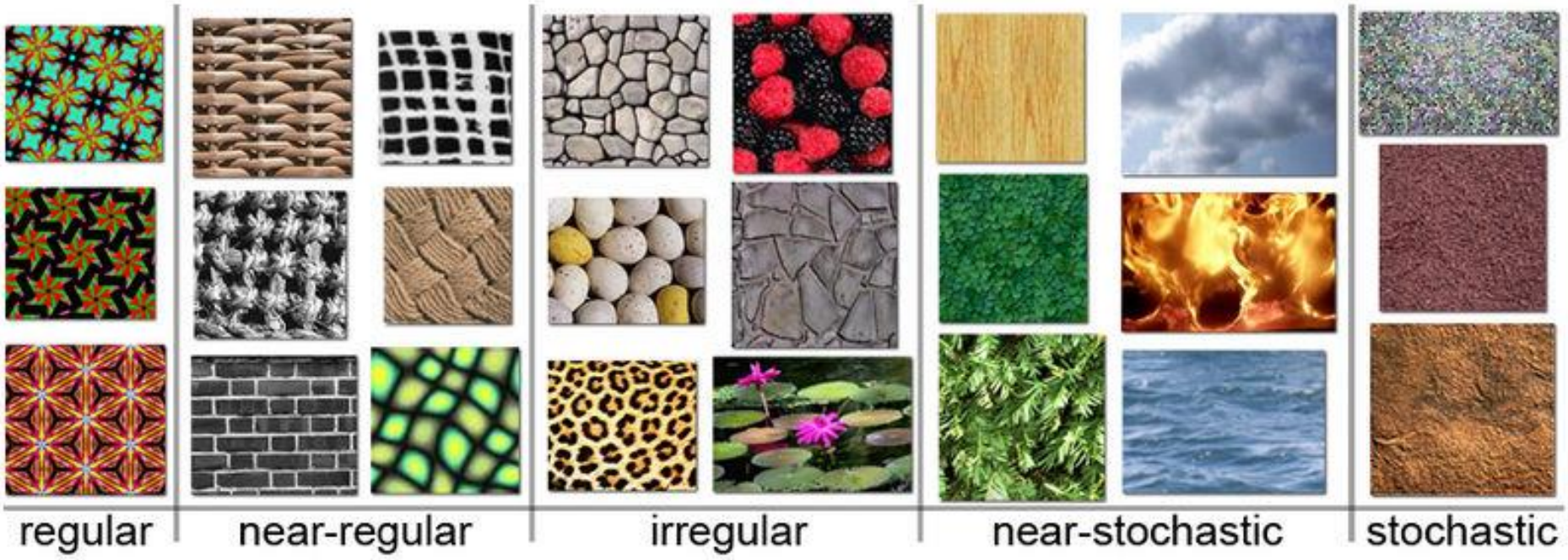
JACOBS UNIVERSITY

COMPUTER VISION
LECTURE 13 – VISUAL BAG OF WORDS

Prof. Dr. Francesco Maurelli
2018-10-16

Object → **Bag of 'words'**

Origin 1: Texture Recognition

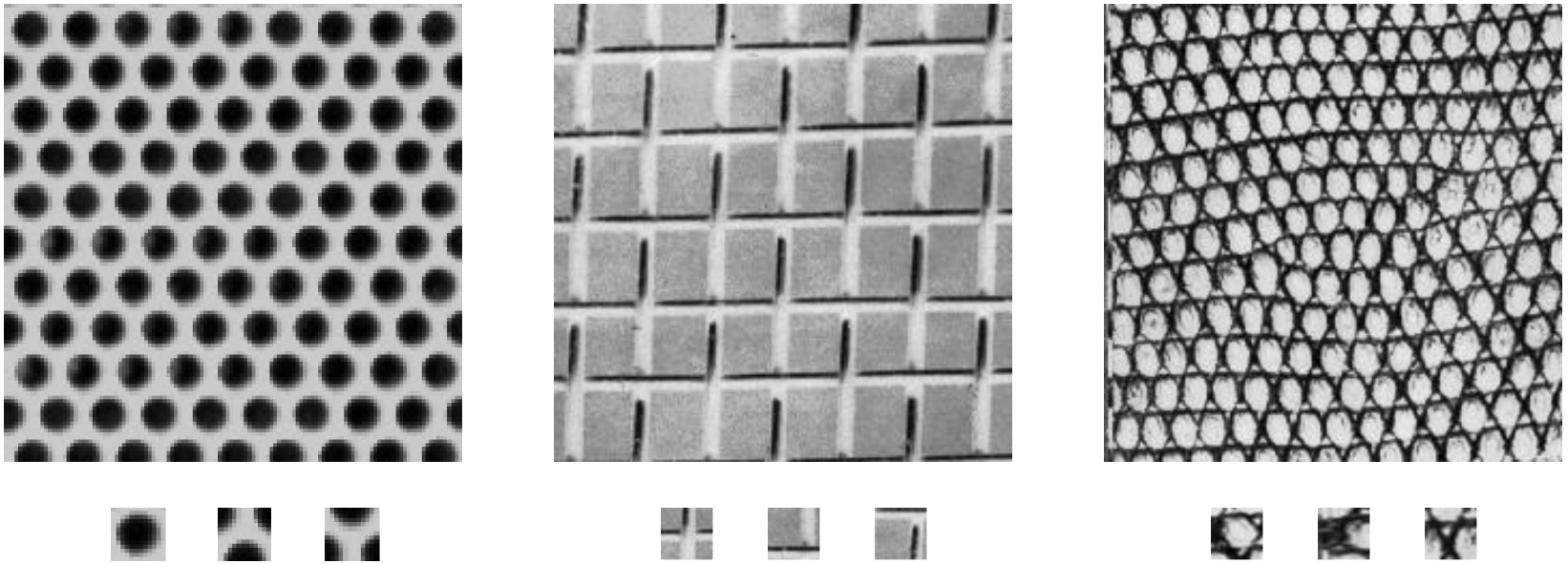


Example textures (from Wikipedia)

Texture properties (psychology of perception):
coarseness, contrast, directionality, line-likeness and *roughness*

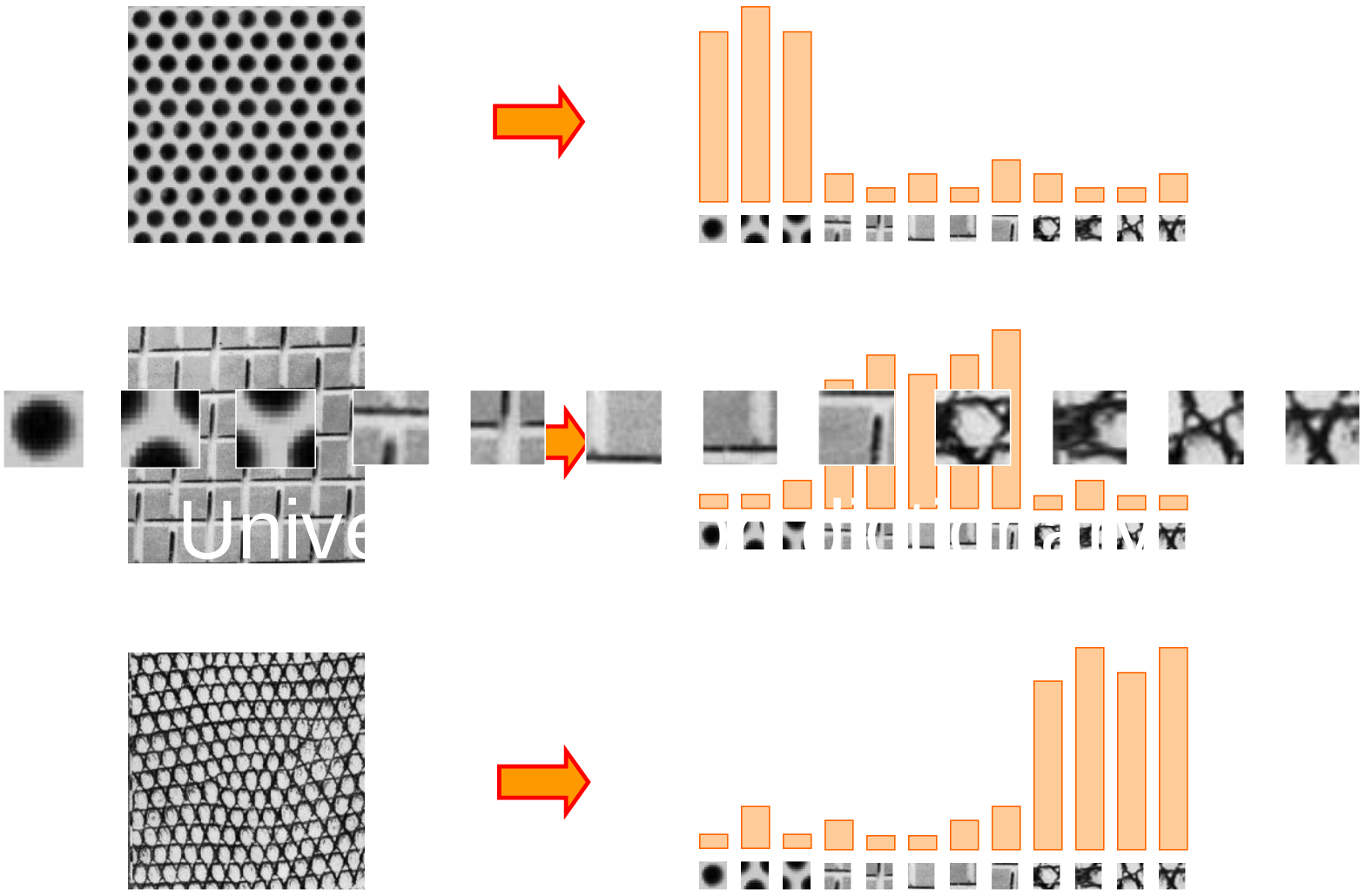
Origin 1: Texture recognition

- Texture is characterized by the repetition of basic elements or ***textons***



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



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






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

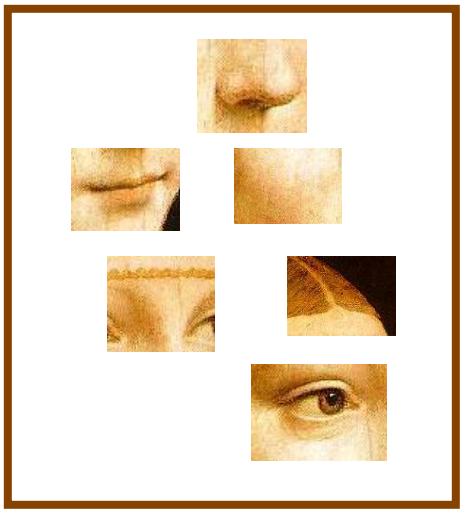
Bag of features

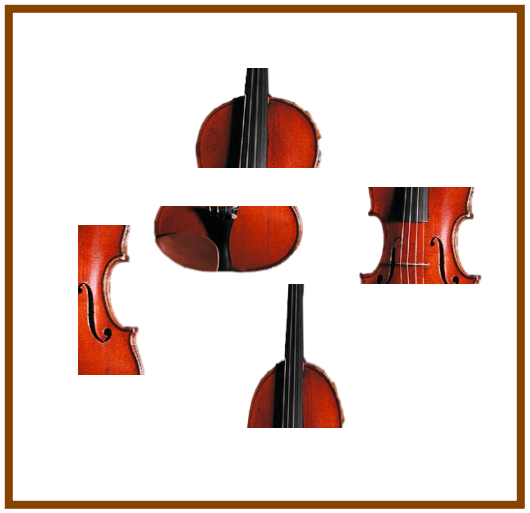
- First, take a bunch of images, extract features, and build up a “dictionary” or “visual vocabulary” – a list of common features

- 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

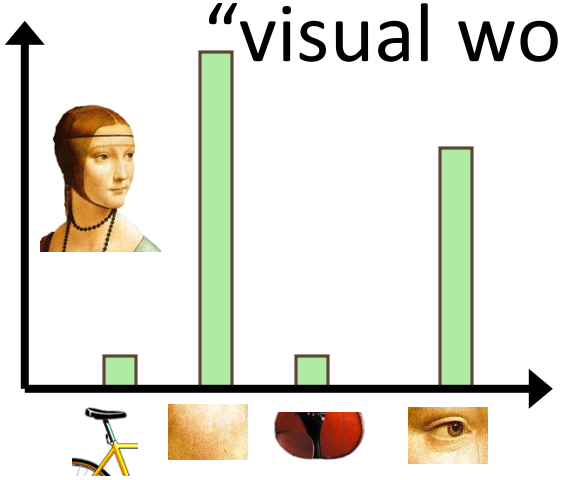
1. Extract features
2. Learn “visual vocabulary”

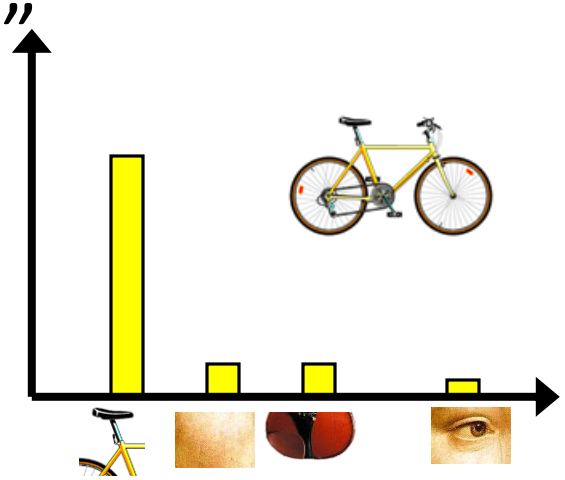
Bag of features: outline

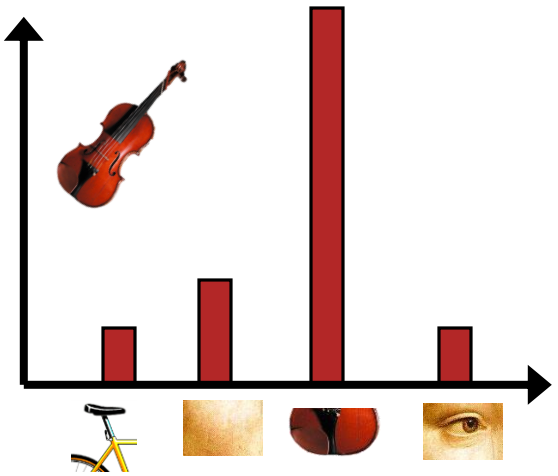
1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary

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”

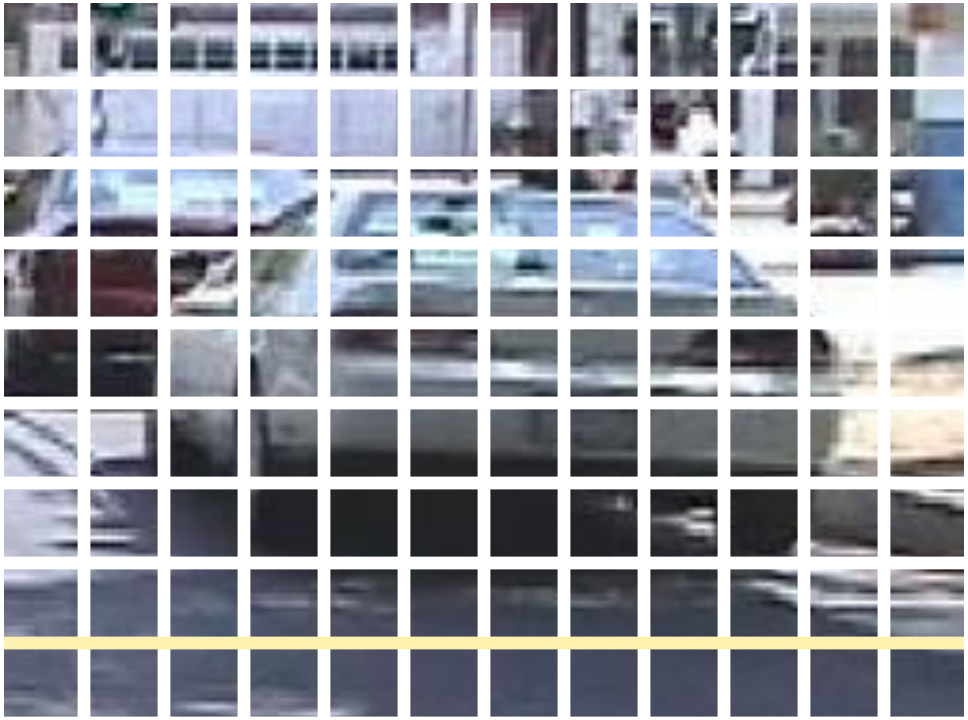






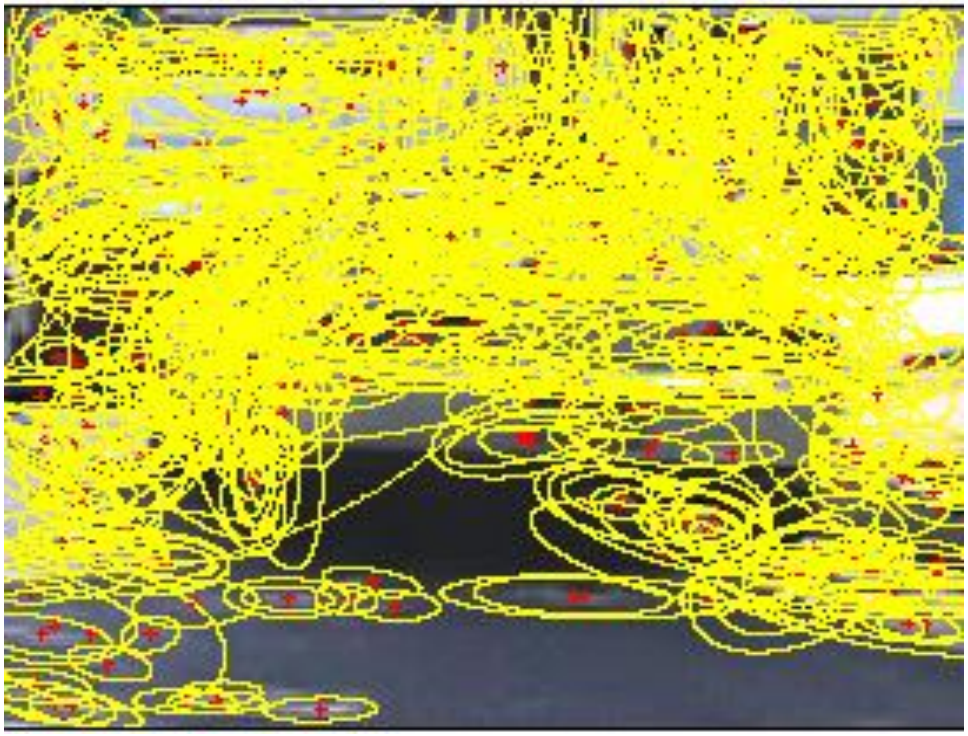
1. Feature extraction

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



1. Feature extraction

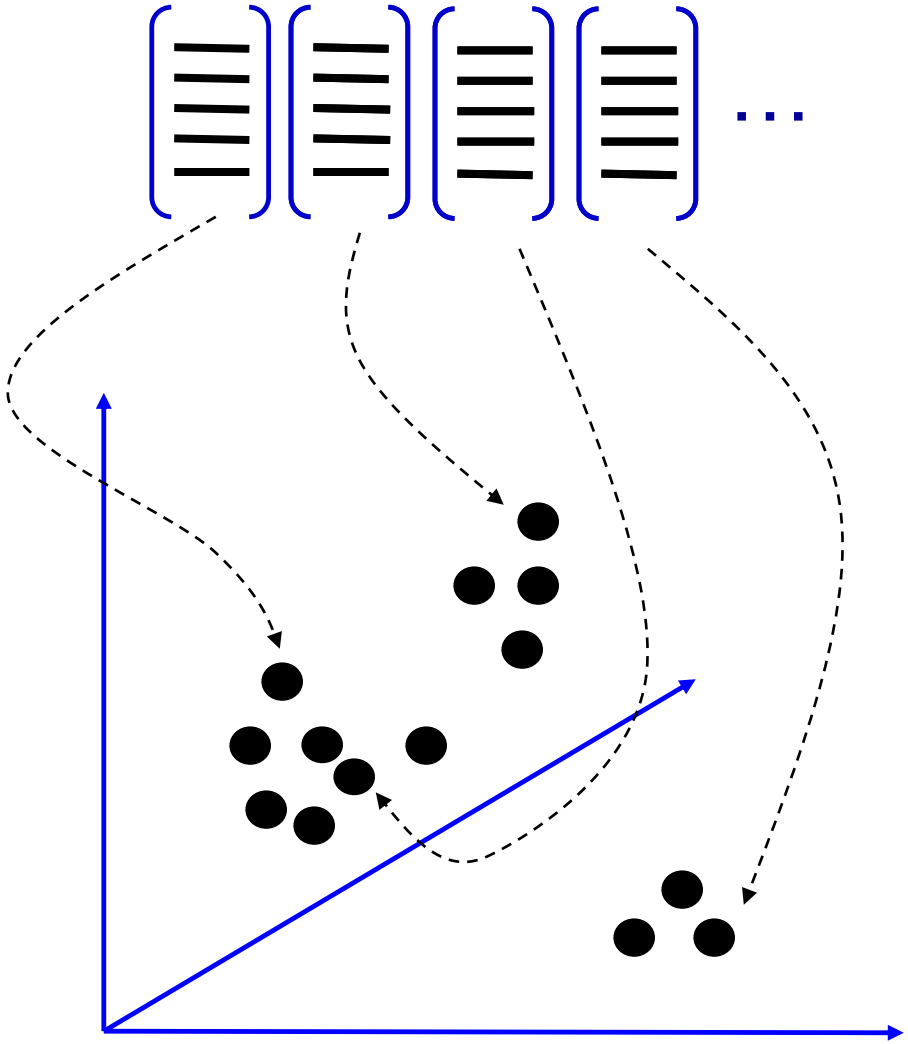
- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic et al. 2005



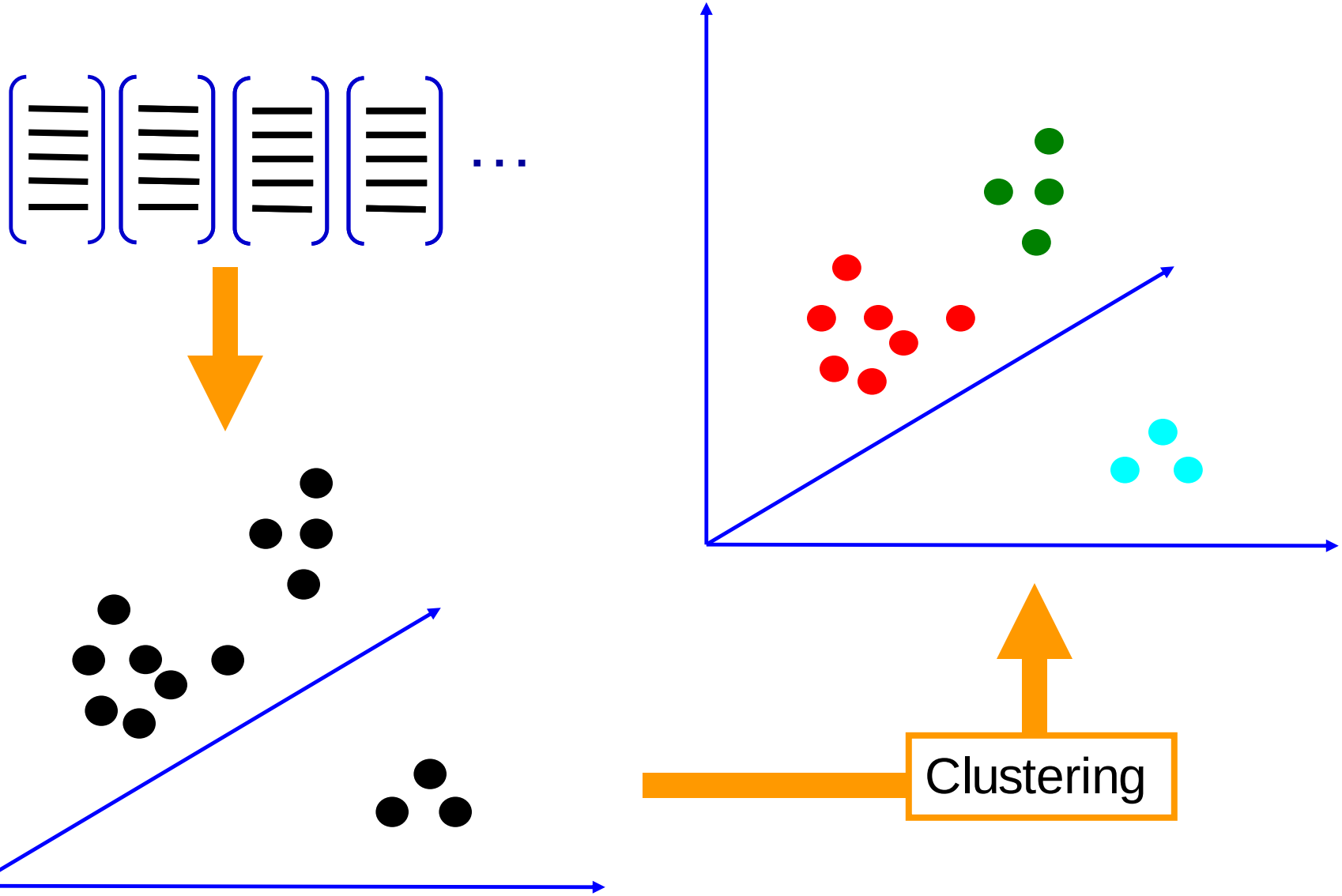
1. Feature extraction

- 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)

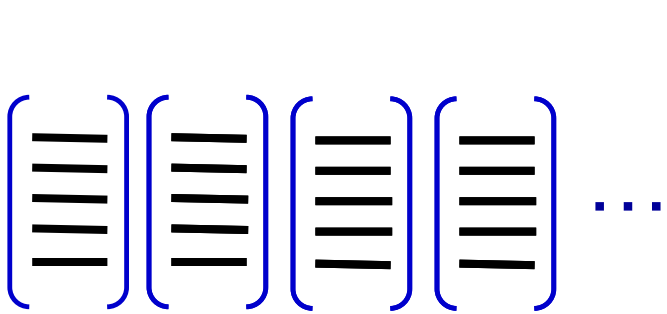
2. Learning the visual vocabulary

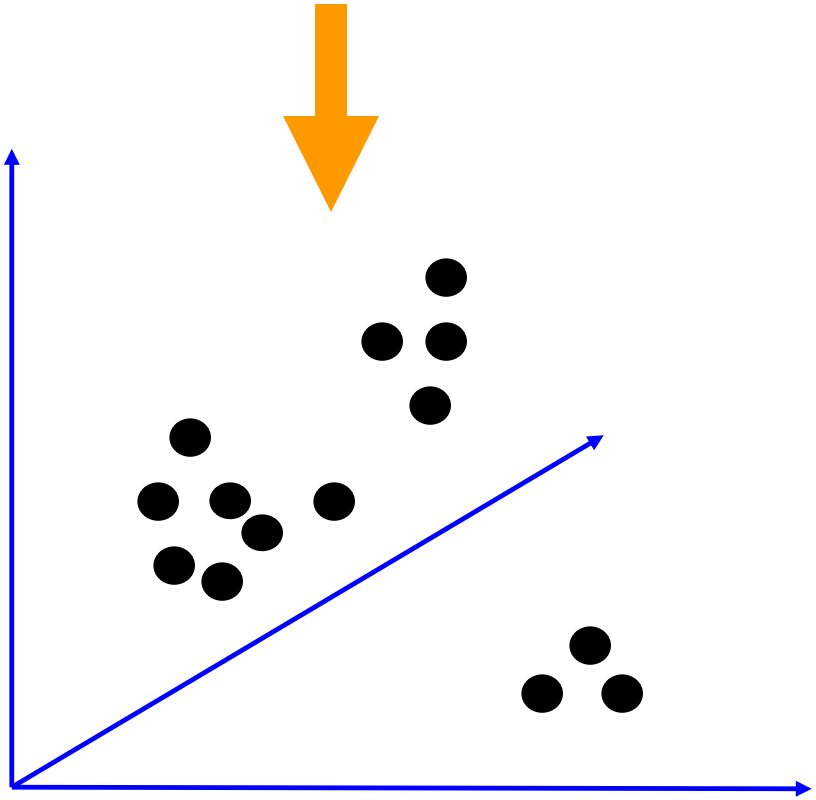


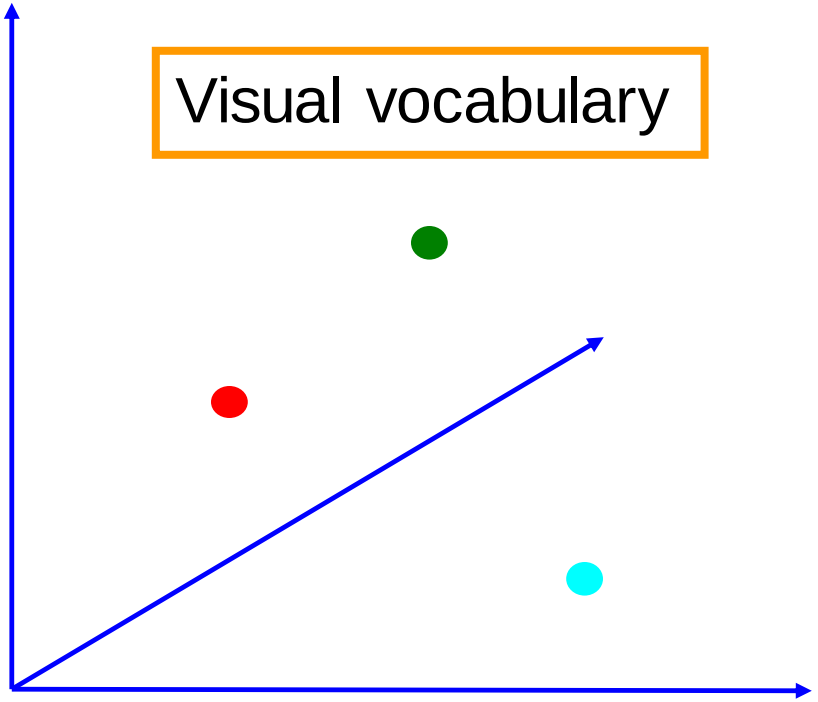
2. Learning the visual vocabulary



2. Learning the visual vocabulary







K-means clustering recap

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

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

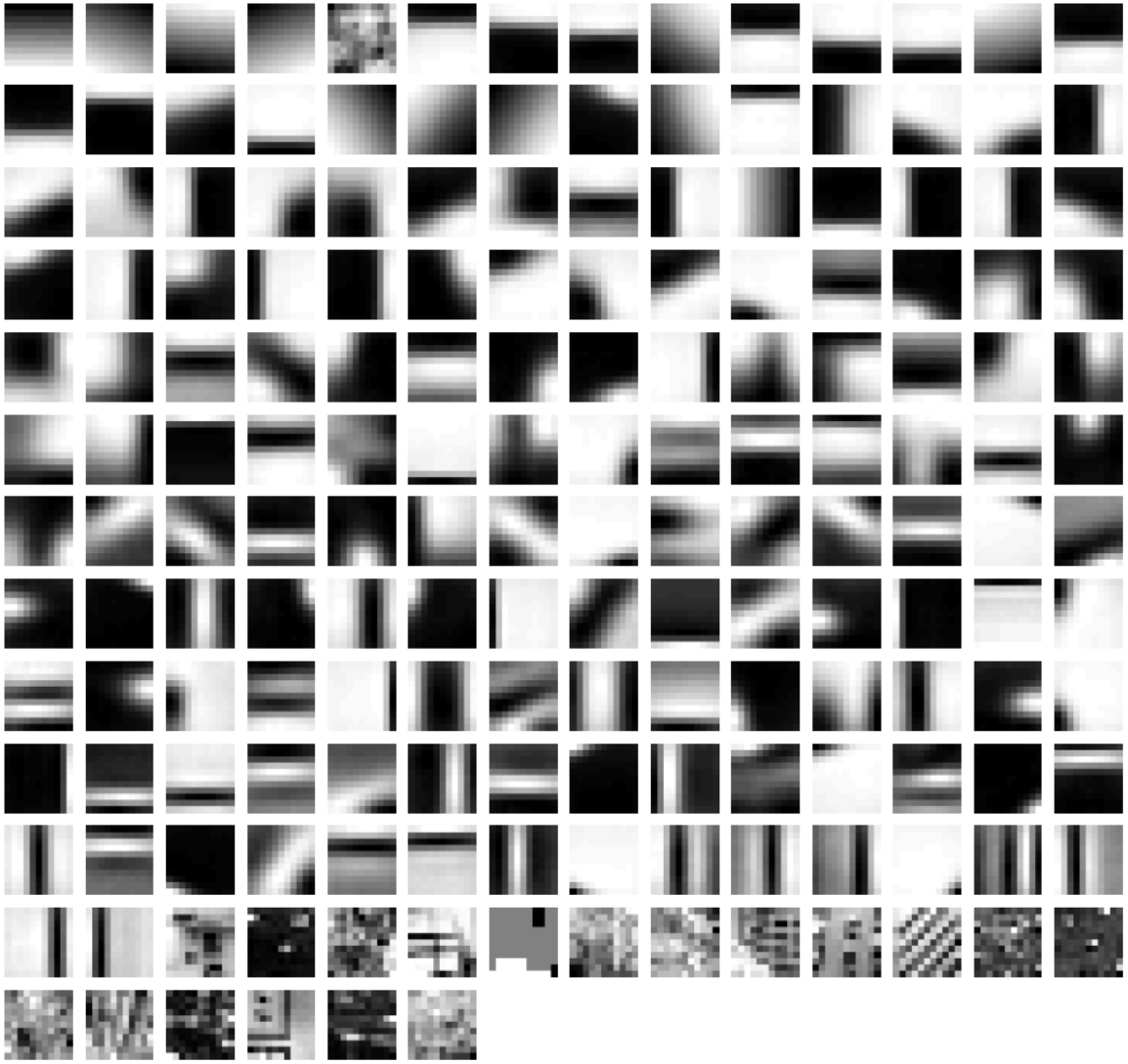
- Algorithm:
- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each data point to the nearest center
  - Recompute each cluster center as the mean of all points assigned to it

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”

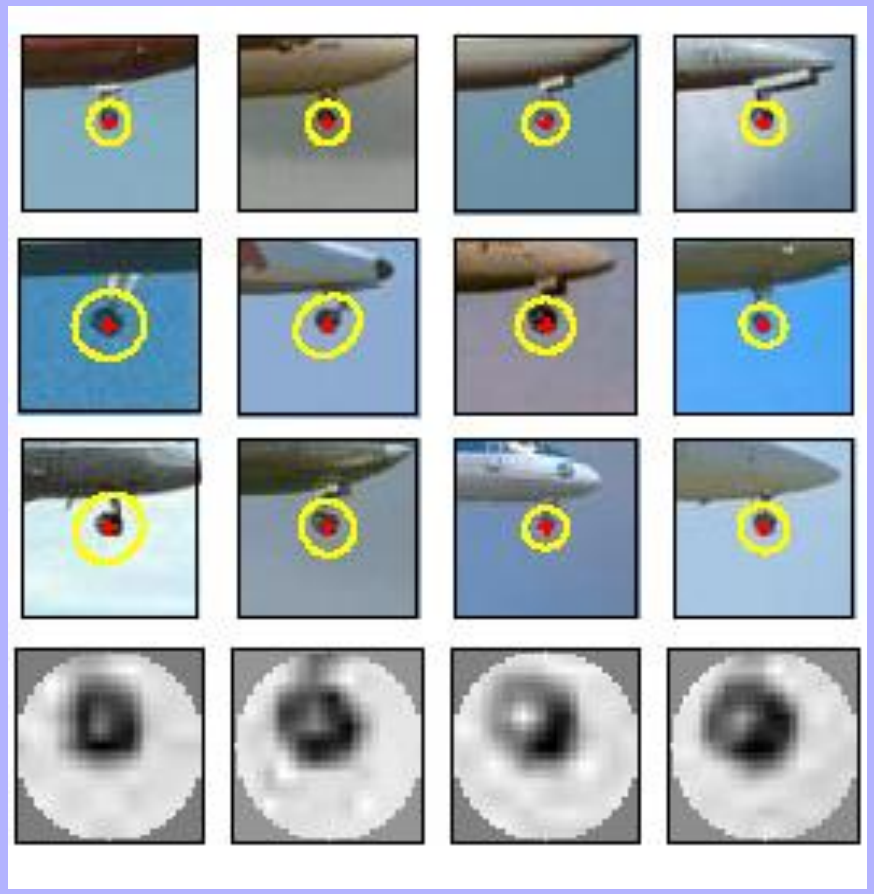
- The codebook is used for quantizing features
  - A *vector quantizer* takes a feature vector and maps it to the index of the nearest codevector in a codebook
  - Codebook = visual vocabulary
  - Codevector = visual word

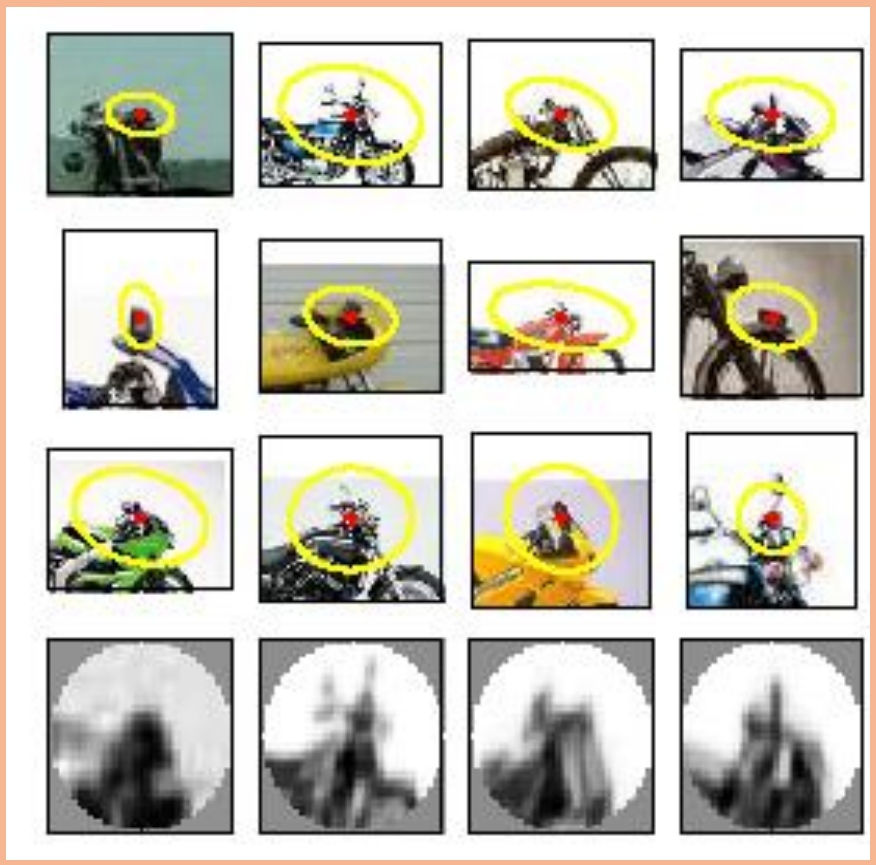
Example visual vocabulary



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
- Computational efficiency
  - Vocabulary trees (Nister & Stewenius, 2006)

3. Image representation

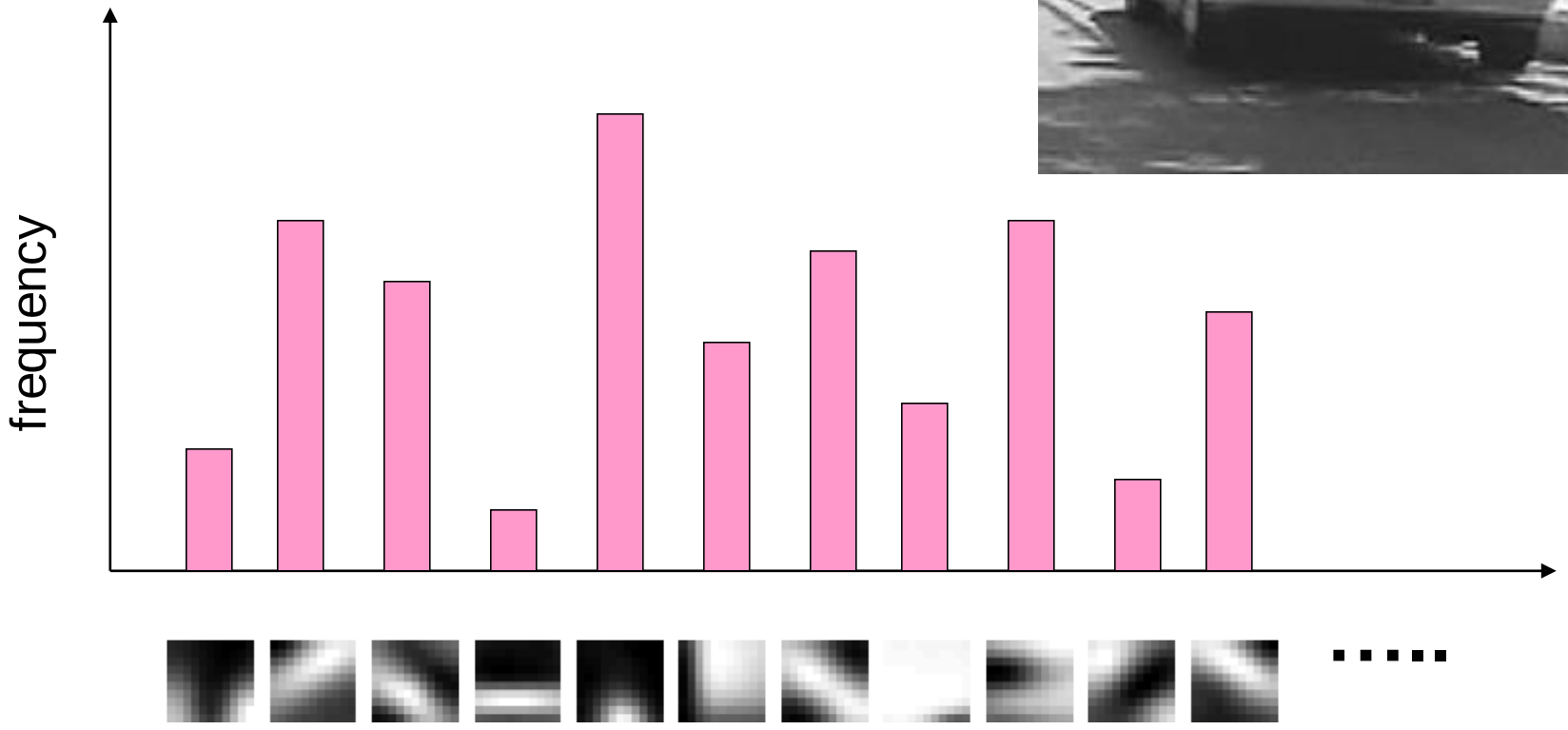
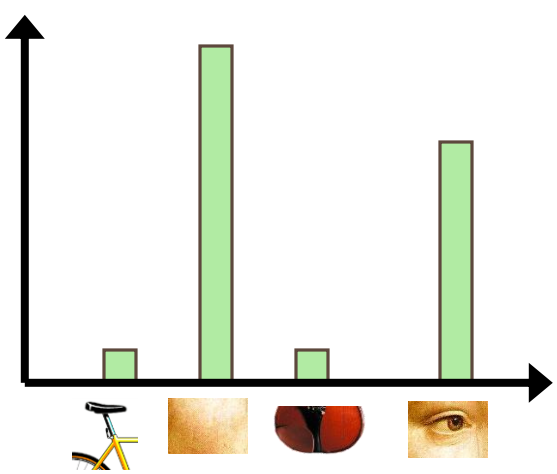
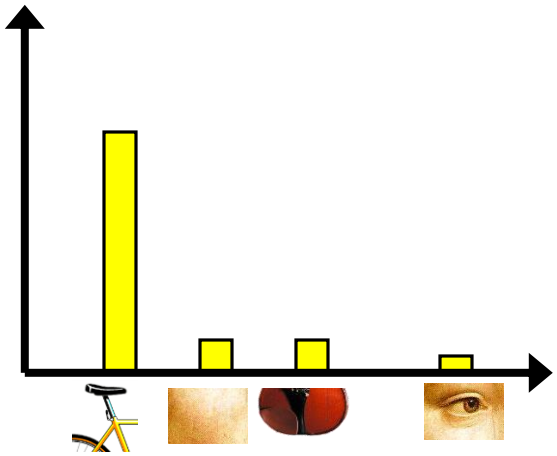
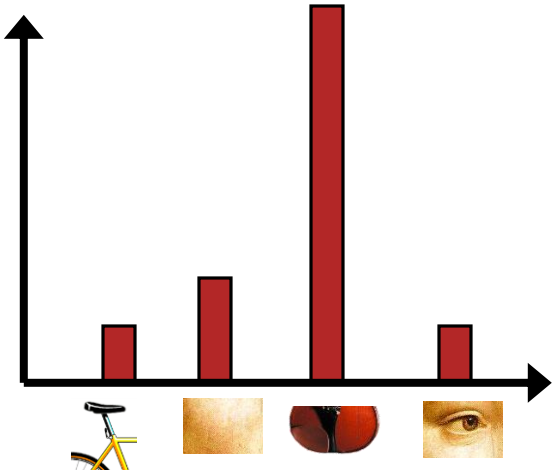


Image classification

- 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

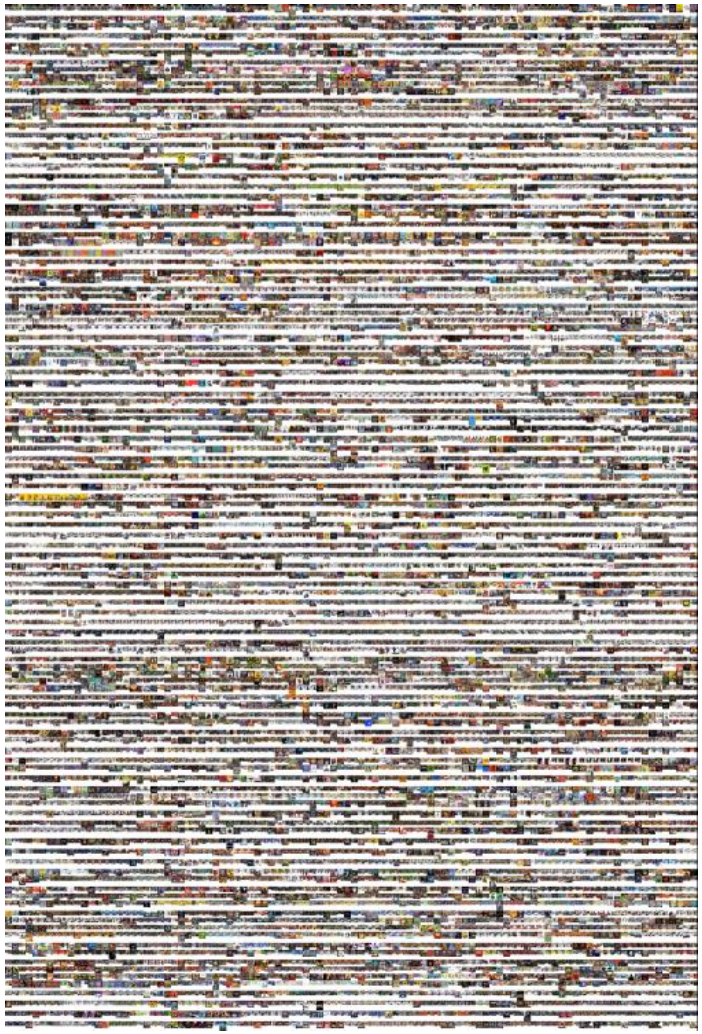
Large-scale image matching

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* **x** *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

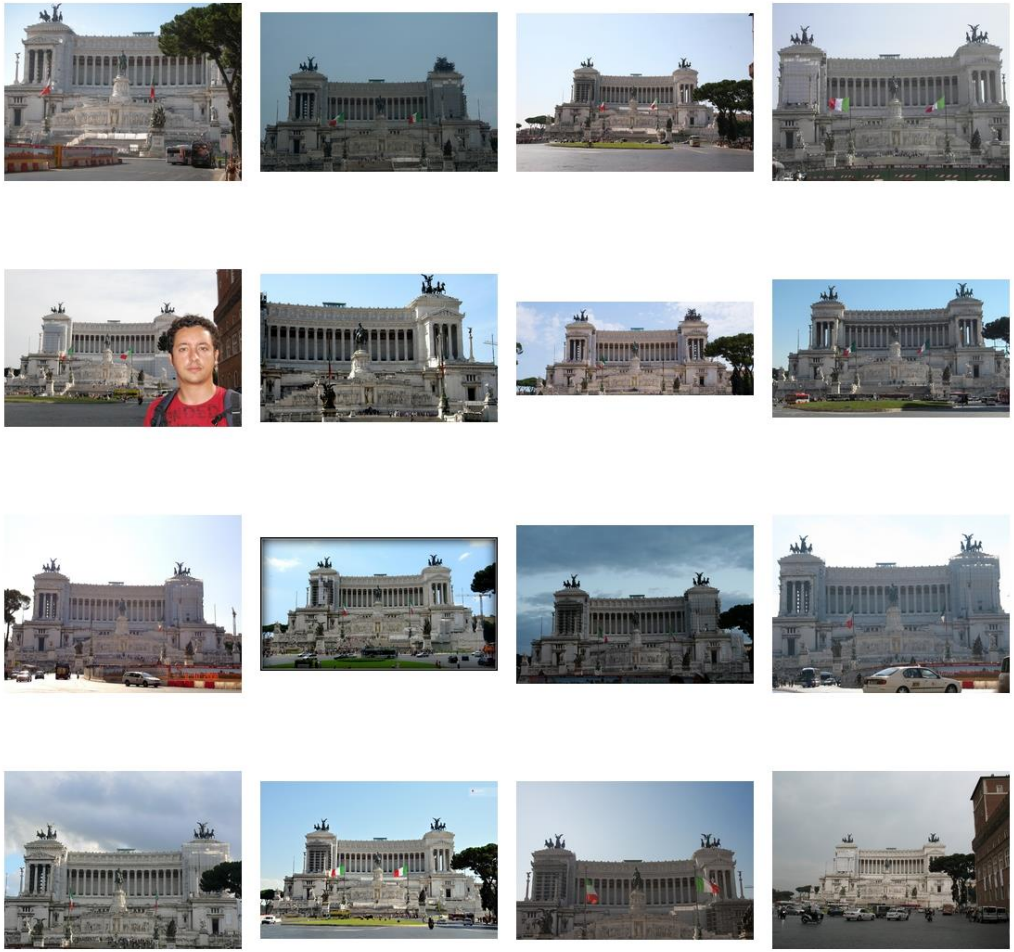
Large-scale image search

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

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

Nister & Stewenius, **Scalable Recognition with a Vocabulary Tree**

Example bag-of-words matches



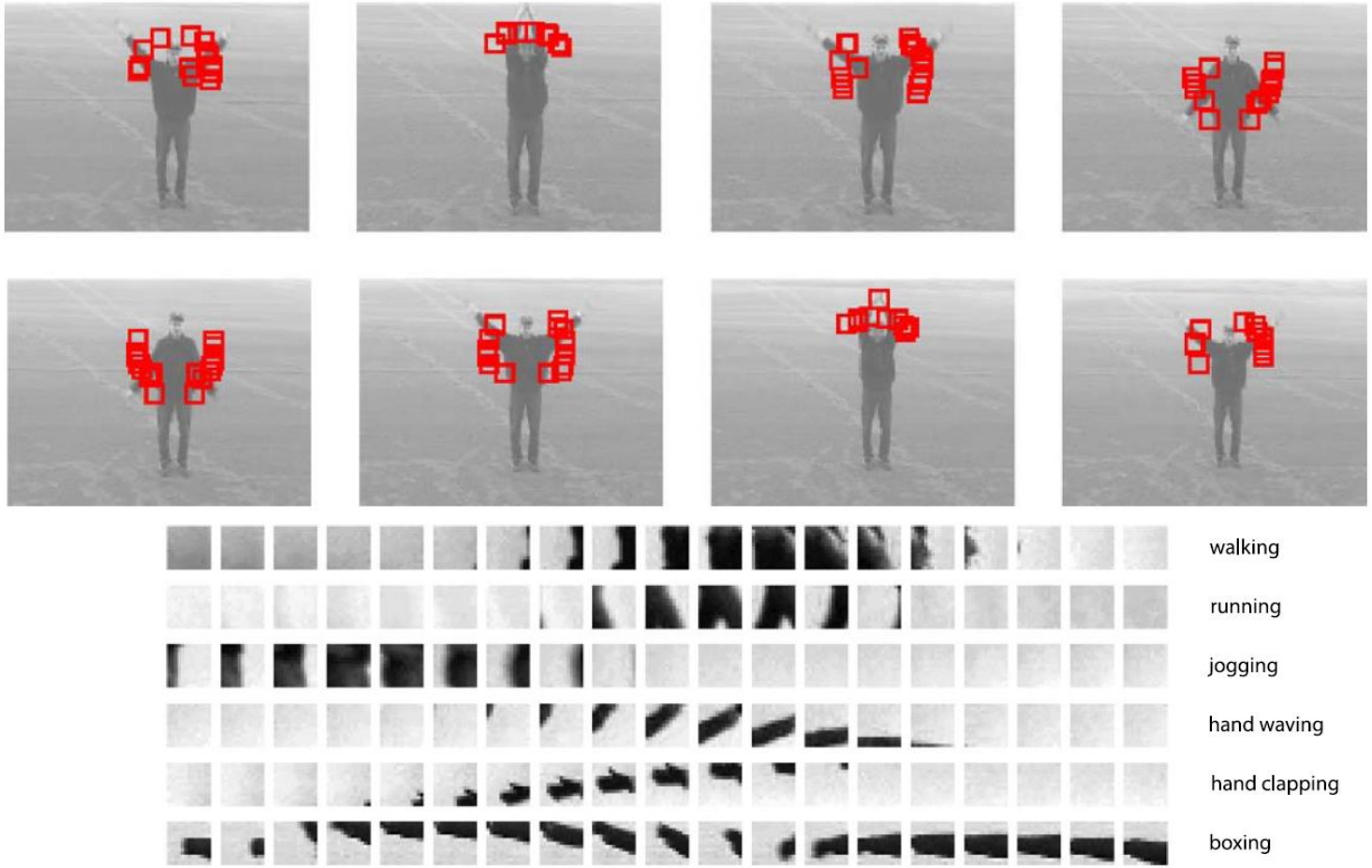
Example bag-of-words matches

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

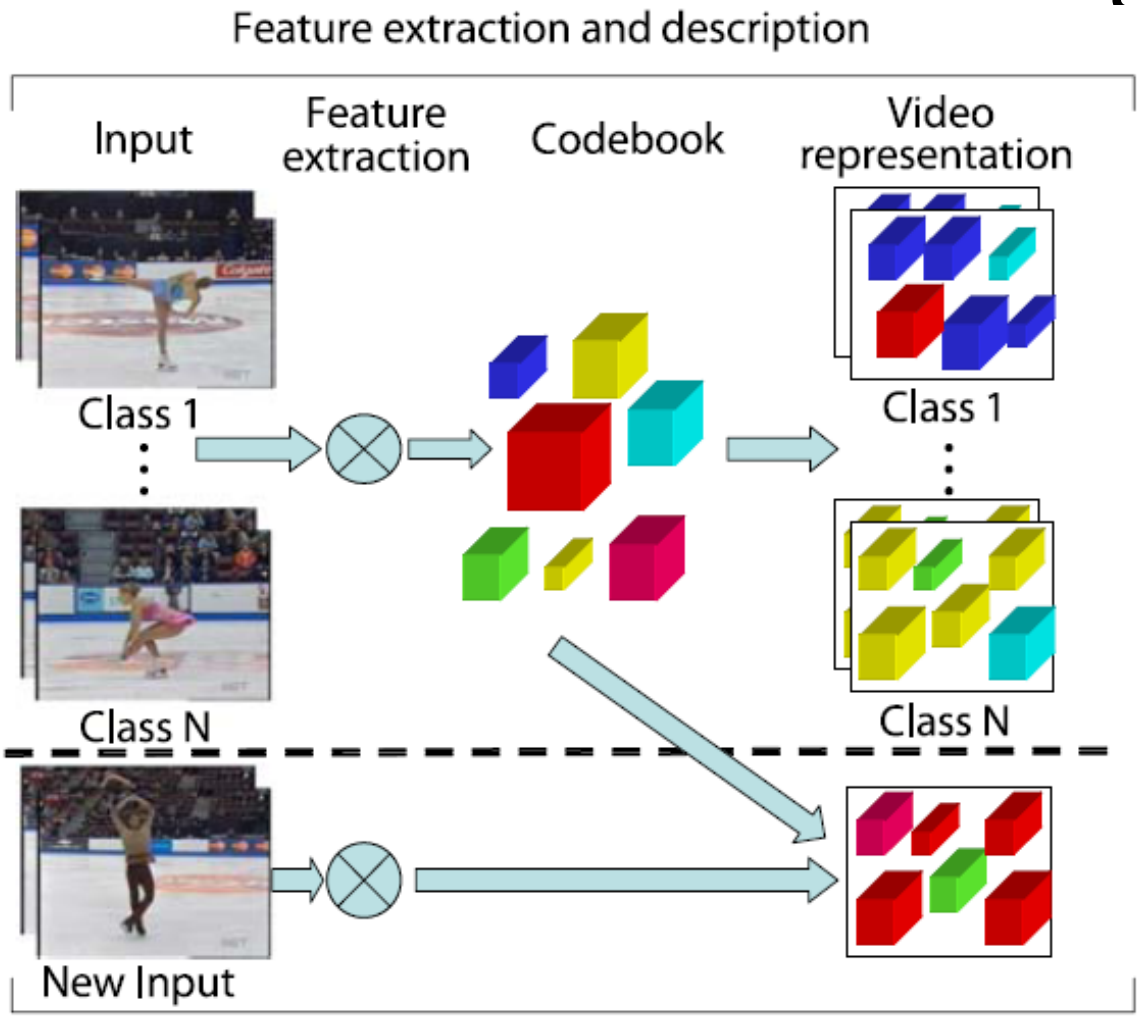
Bags of features for action recognition

Space-time interest points



Juan Carlos Niebles, Hongcheng Wang and Li Fei-Fei, **Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words**, IJCV 2008.

Bags of features for action recognition



Juan Carlos Niebles, Hongcheng Wang and Li Fei-Fei, **Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words**, IJCV 2008.