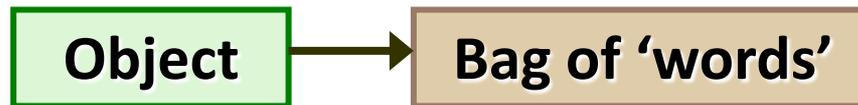


# CS6670: Computer Vision

Noah Snavely

## Lecture 16: Bag-of-words models

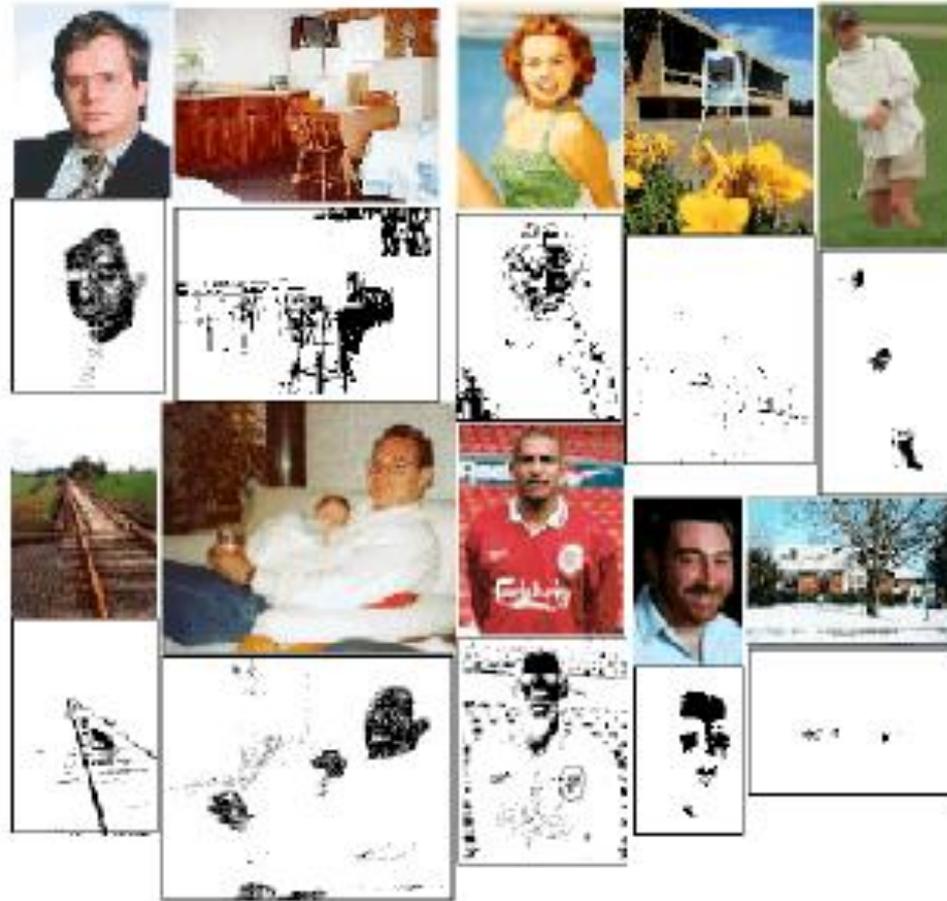


# Announcements

- Project 3: Eigenfaces
  - due Wednesday, November 11 at 11:59pm
  - solo project
  
- Final project presentations:
  - During the final exams period

# Skin detection results

---



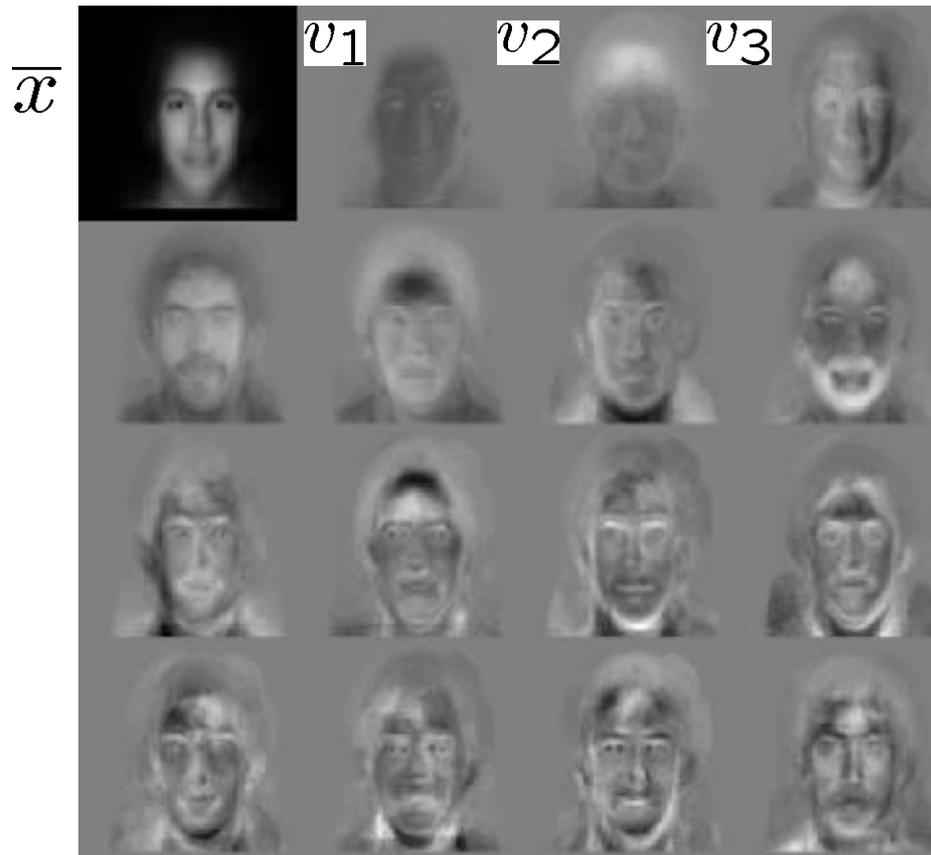
**Figure 25.3.** The figure shows a variety of images together with the output of the skin detector of Jones and Rehg applied to the image. Pixels marked black are skin pixels, and white are background. Notice that this process is relatively effective, and could certainly be used to focus attention on, say, faces and hands. *Figure from "Statistical color models with application to skin detection," M.J. Jones and J. Rehg, Proc. Computer Vision and Pattern Recognition, 1999 © 1999, IEEE*

# Eigenfaces

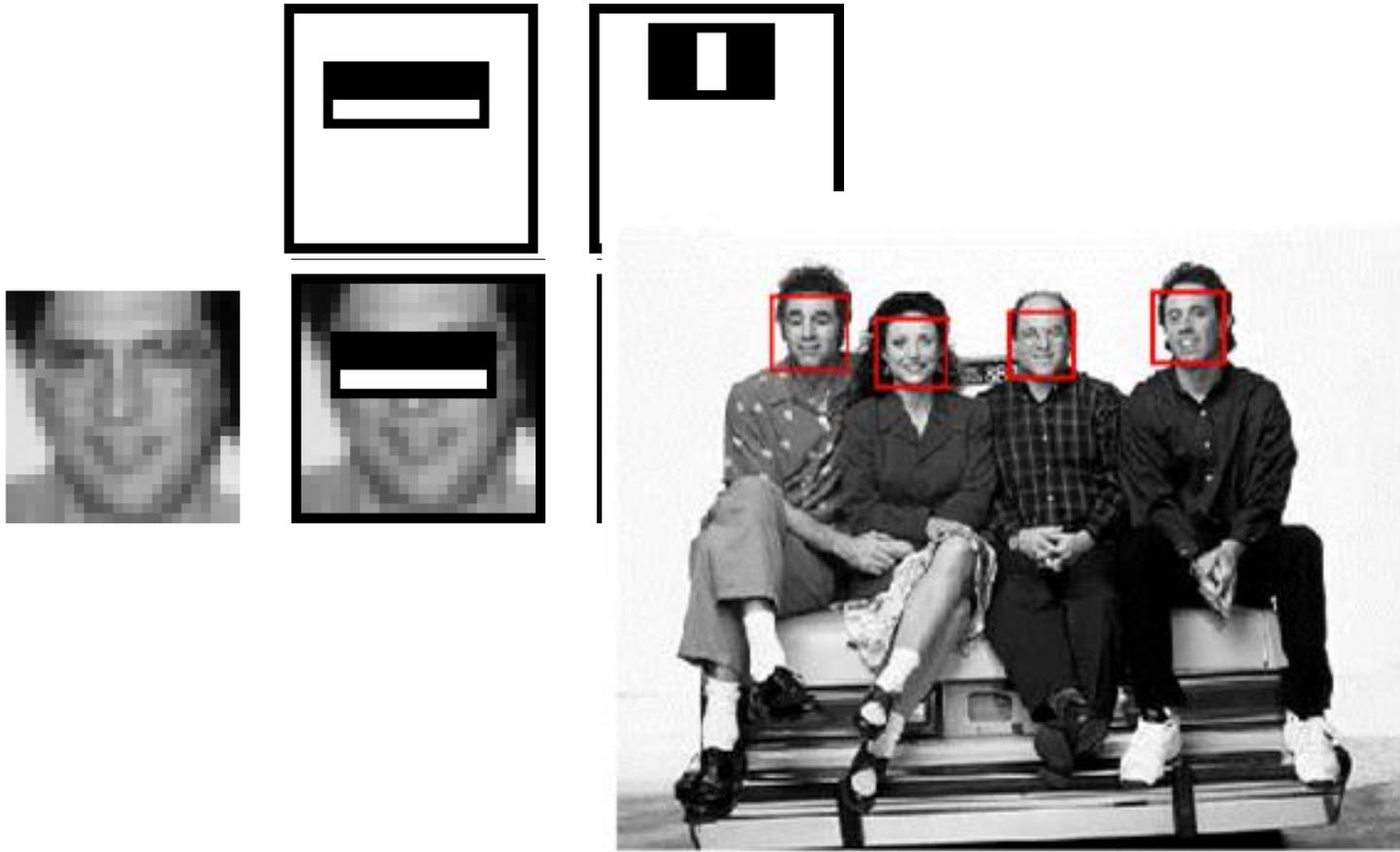
---

PCA extracts the eigenvectors of  $\mathbf{A}$

- Gives a set of vectors  $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \dots$
- Each one of these vectors is a direction in face space
  - what do these look like?



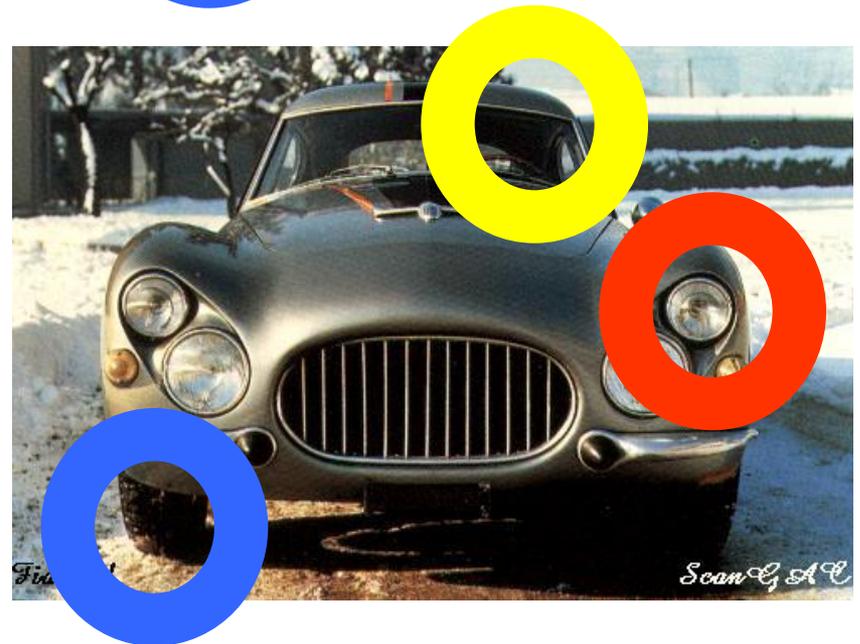
# Viola-Jones Face Detector: Results



# Moving forward

- Faces are pretty well-behaved
  - Mostly the same basic shape
  - Lie close to a low-dimensional subspace
- Not all objects are as nice

# Different appearance, similar parts





# Bag of Words Models

Adapted from slides by Rob Fergus  
and Svetlana Lazebnik

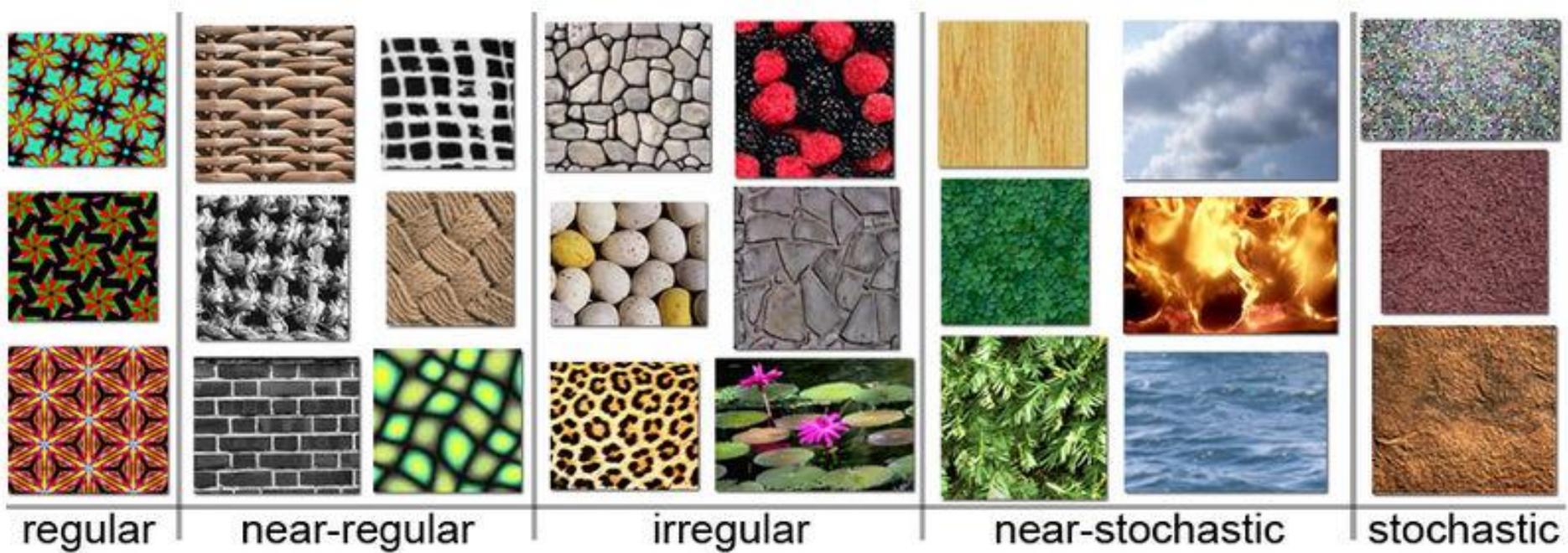
**Object**



**Bag of 'words'**



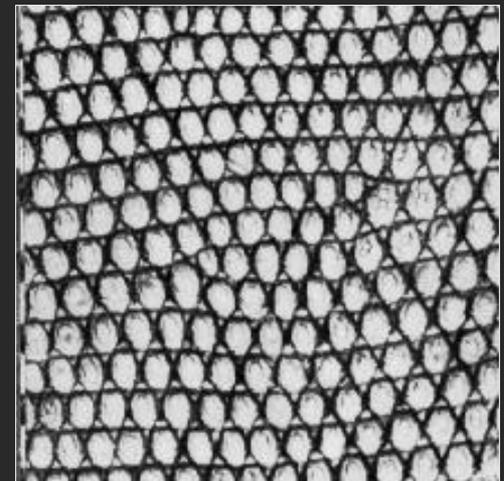
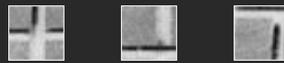
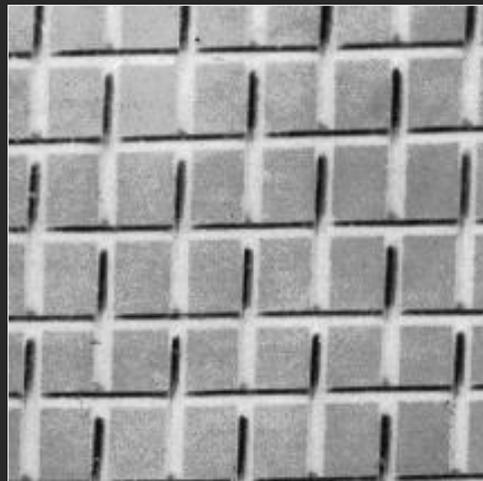
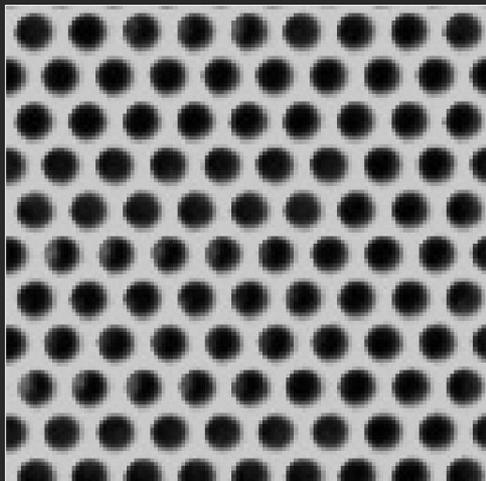
# Origin 1: Texture Recognition



Example textures (from Wikipedia)

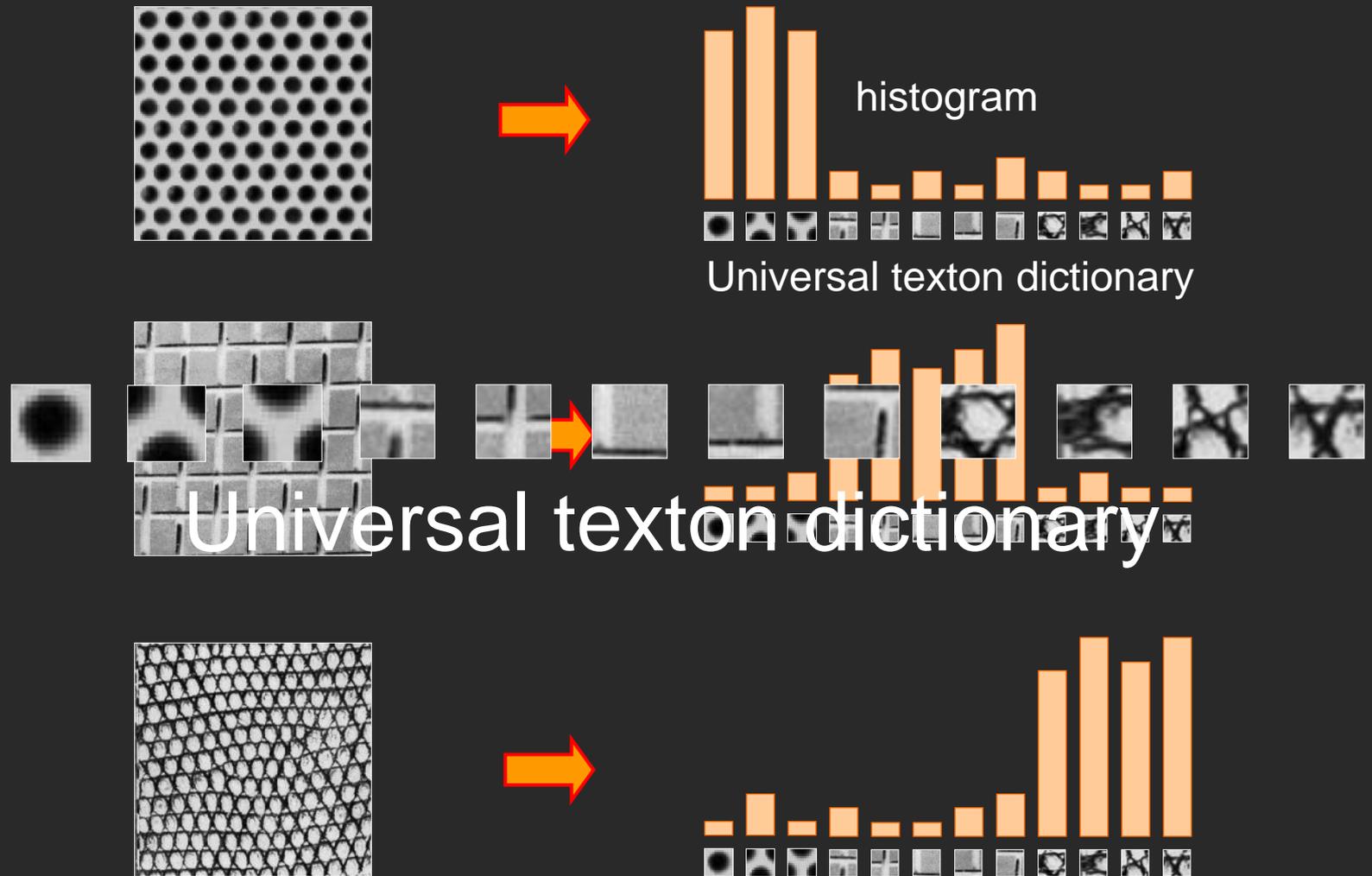
# Origin 1: Texture recognition

- 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)

2007-01-23: State of the Union Address

George W. Bush (2001-)

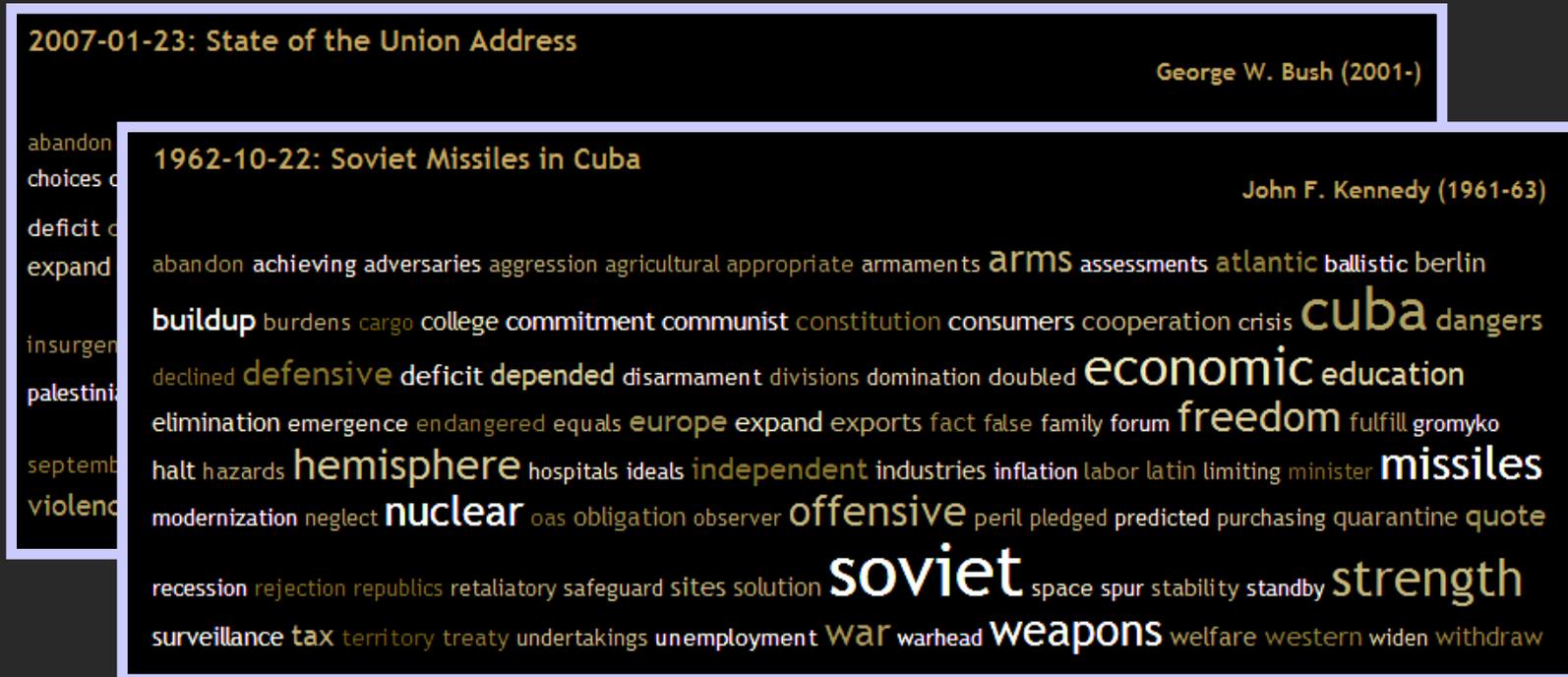
abandon accountable affordable afghanistan africa aided ally anbar armed army **baghdad** bless **challenges** chamber chaos  
choices civilians coalition commanders **commitment** confident confront congressman constitution corps debates deduction  
deficit deliver **democratic** deploy dikembe diplomacy disruptions earmarks **economy** einstein **elections** eliminates  
expand **extremists** failing faithful families **freedom** fuel **funding** god haven ideology immigration impose  
insurgents iran **iraq** islam julie lebanon love madam marine math medicare moderation neighborhoods nuclear offensive  
palestinian payroll province pursuing **qaeda** radical regimes resolve retreat rieman sacrifices science sectarian senate  
september **shia** stays strength students succeed sunni **tax** territories **terrorists** threats uphold victory  
violence violent **war** washington weapons wesley

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

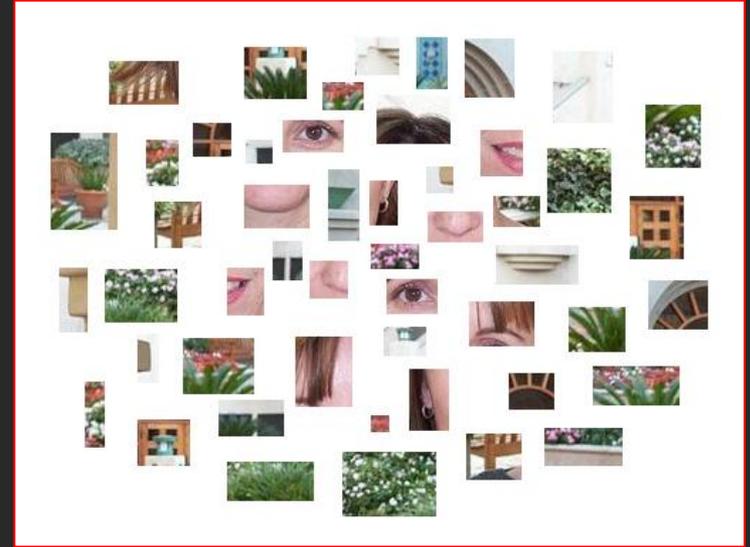
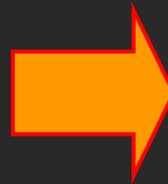
- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)



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*

# Bags of features for object recognition

## Caltech6 dataset



class	bag of features	bag of features	Parts-and-shape model
	Zhang et al. (2005)	Willamowski et al. (2004)	Fergus et al. (2003)
airplanes	<b>98.8</b>	97.1	90.2
cars (rear)	98.3	<b>98.6</b>	90.3
cars (side)	<b>95.0</b>	87.3	88.5
faces	<b>100</b>	99.3	96.4
motorbikes	<b>98.5</b>	98.0	92.5
spotted cats	<b>97.0</b>	—	90.0

# Bag of features

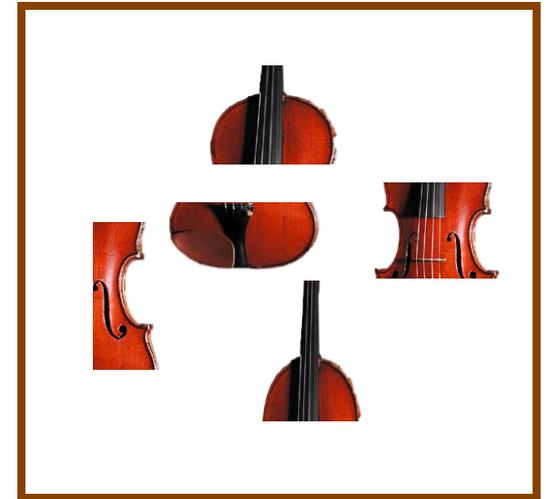
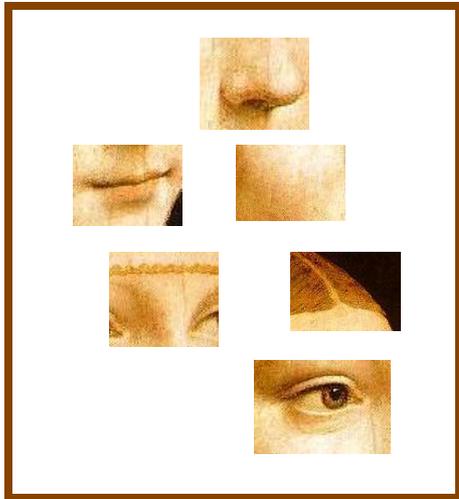
---

- 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

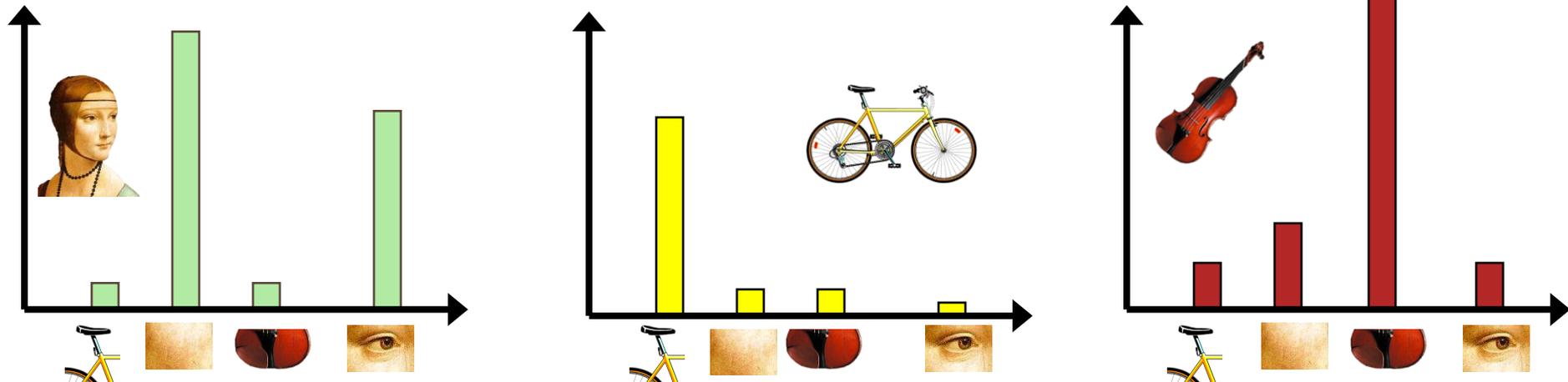
---

1. Extract features
2. Learn “visual vocabulary”
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”

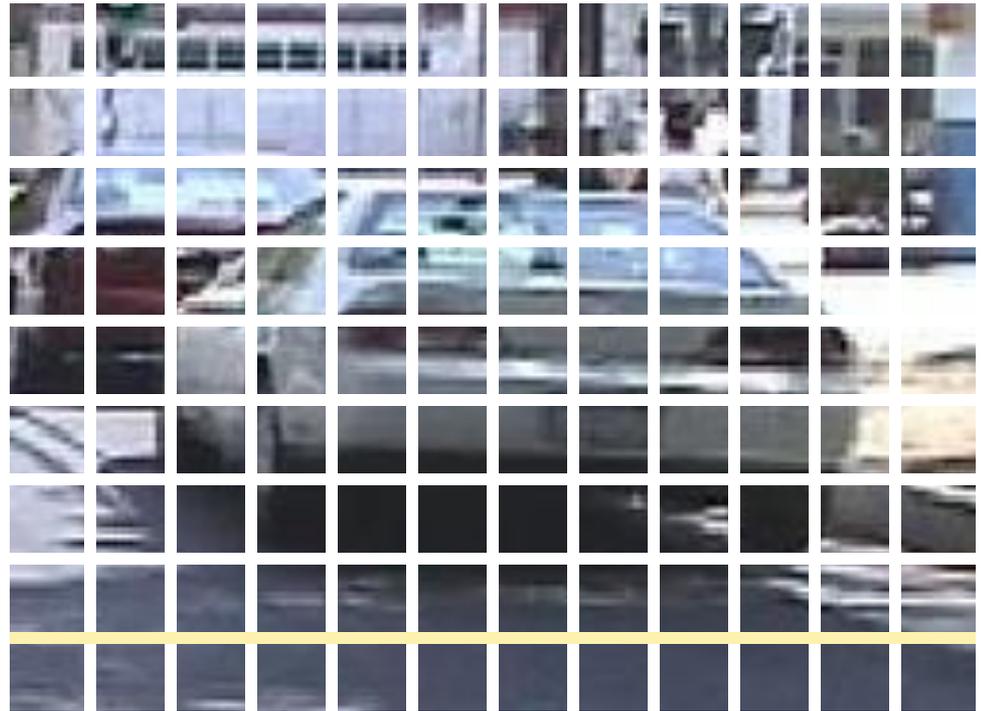


# 1. Feature extraction

---

## Regular grid

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



# 1. Feature extraction

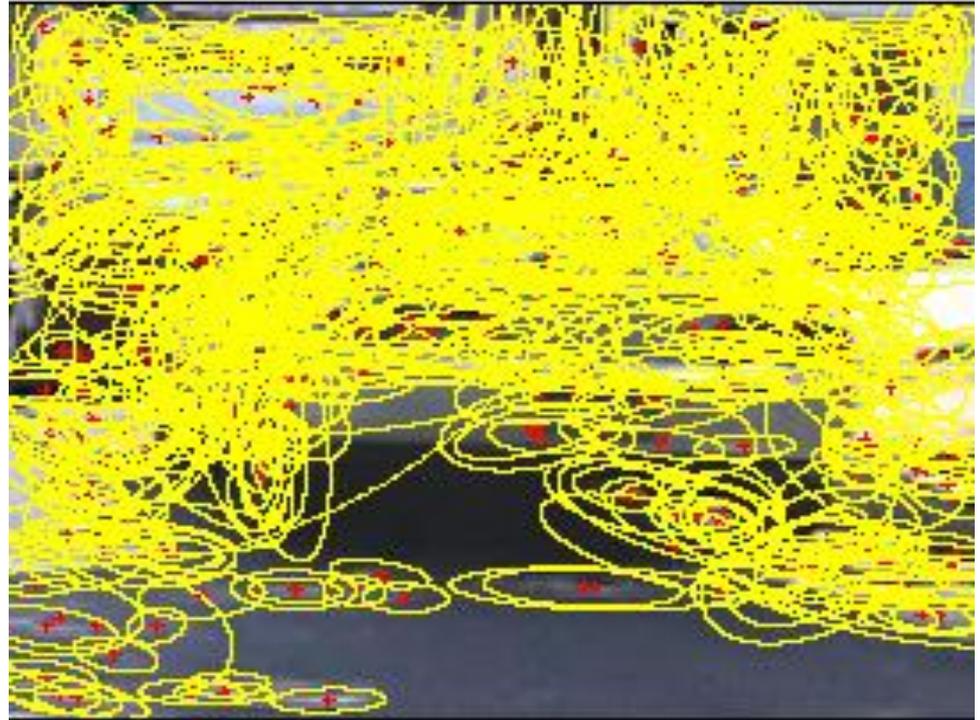
---

## 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

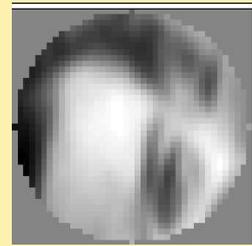
## Other methods

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

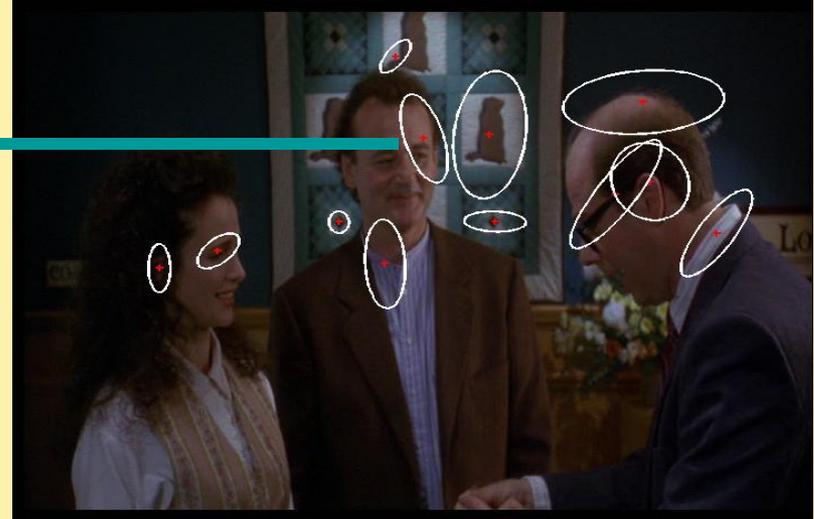
# 1. Feature extraction



**Compute  
SIFT  
descriptor**  
[Lowe'99]



**Normalize  
patch**



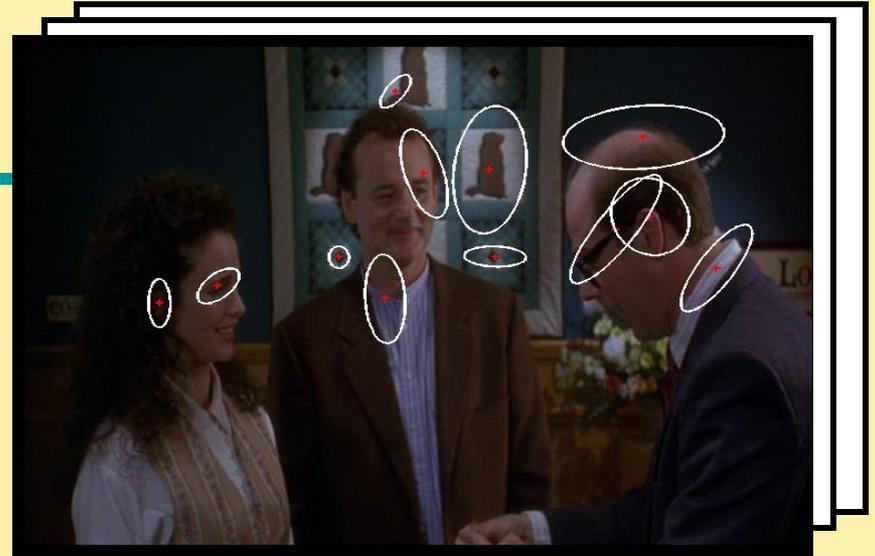
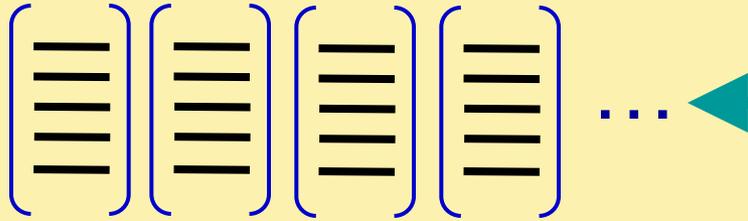
**Detect patches**

[Mikojczyk and Schmid '02]

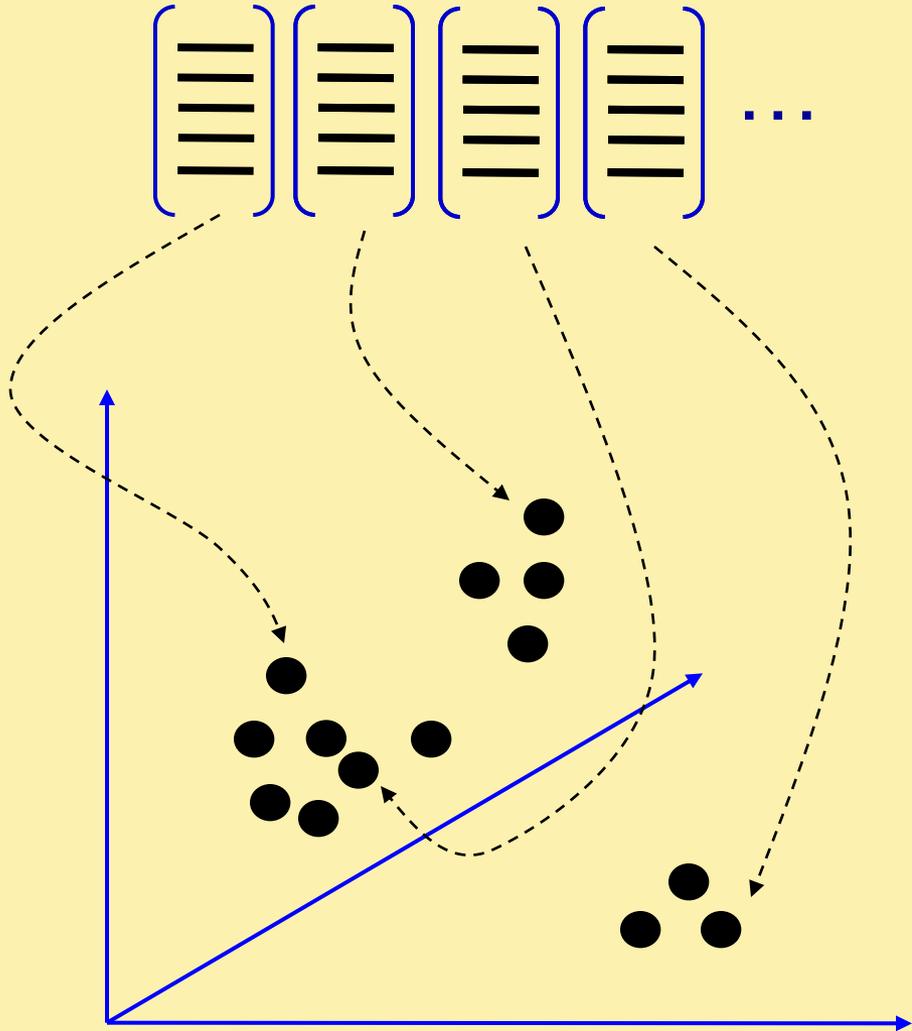
[Mata, Chum, Urban & Pajdla, '02]

[Sivic & Zisserman, '03]

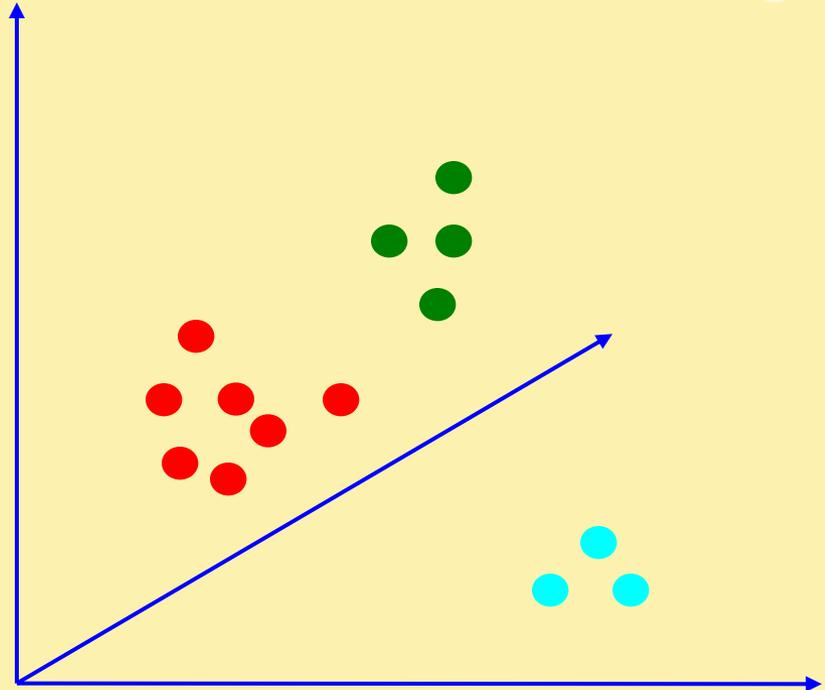
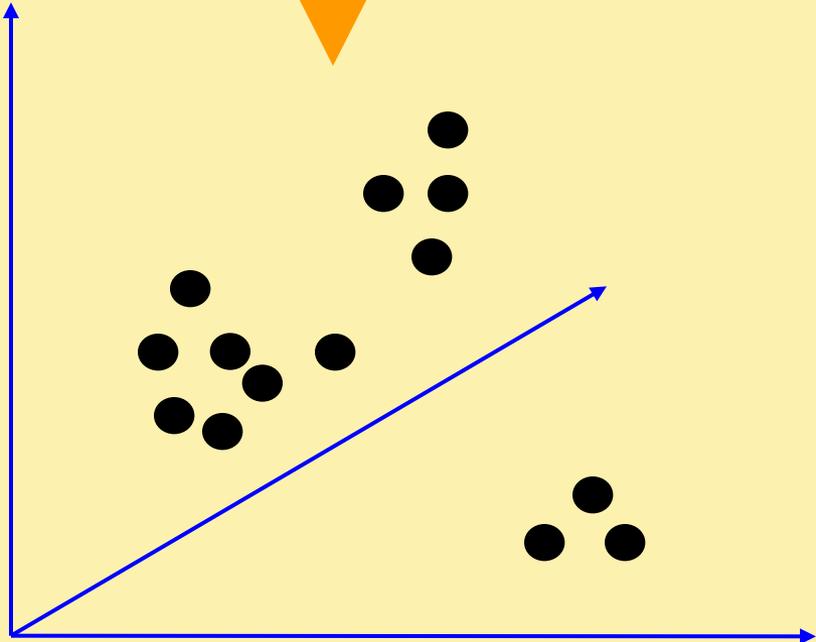
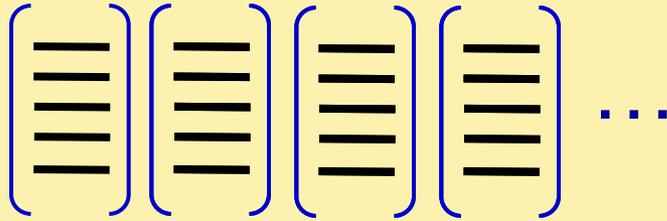
# 1. Feature extraction



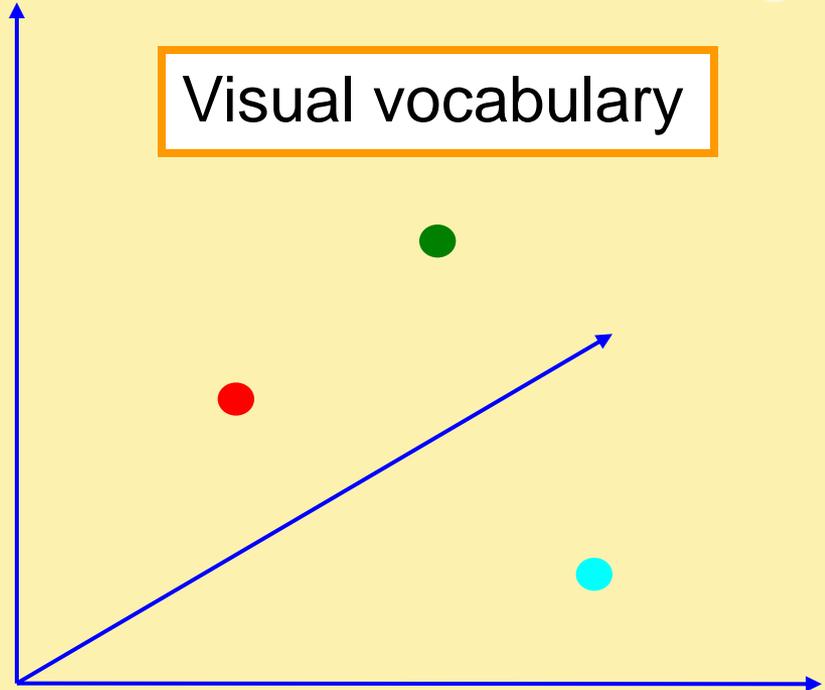
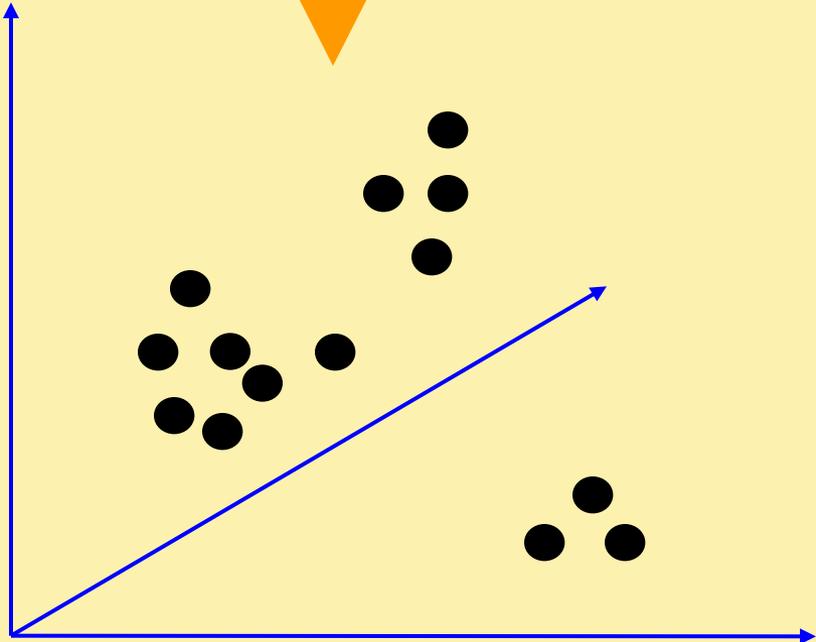
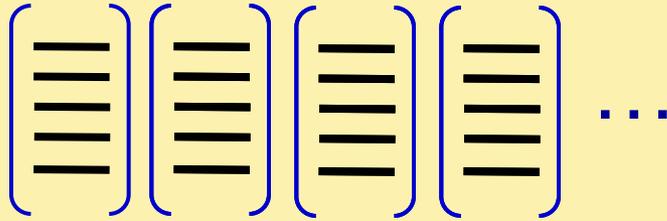
# 2. Learning the visual vocabulary



# 2. Learning the visual vocabulary



# 2. Learning the visual vocabulary



# 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:

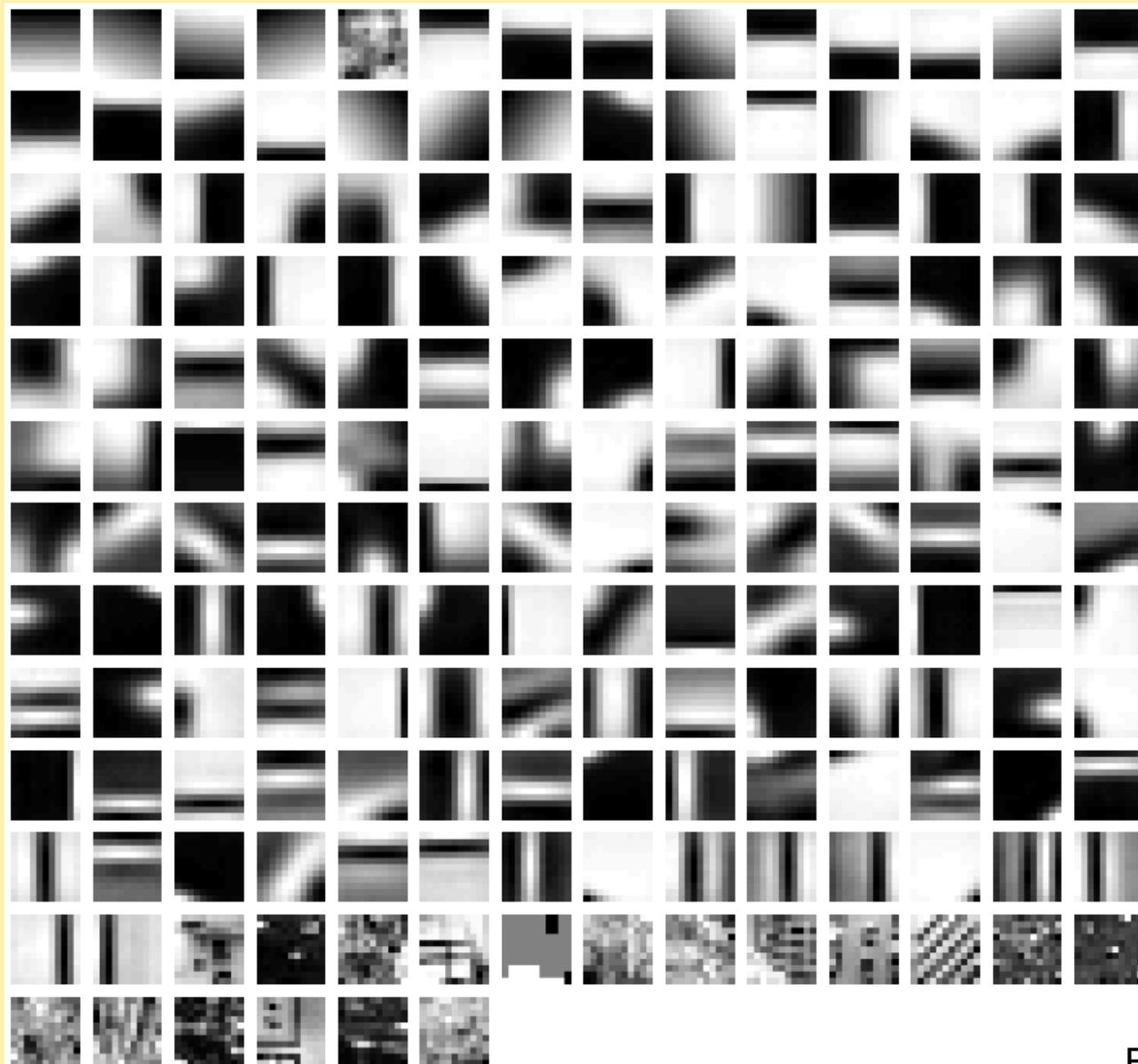
- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each data point to the nearest center
  - Recompute each cluster center as the mean of all points assigned to it

# From clustering to vector quantization

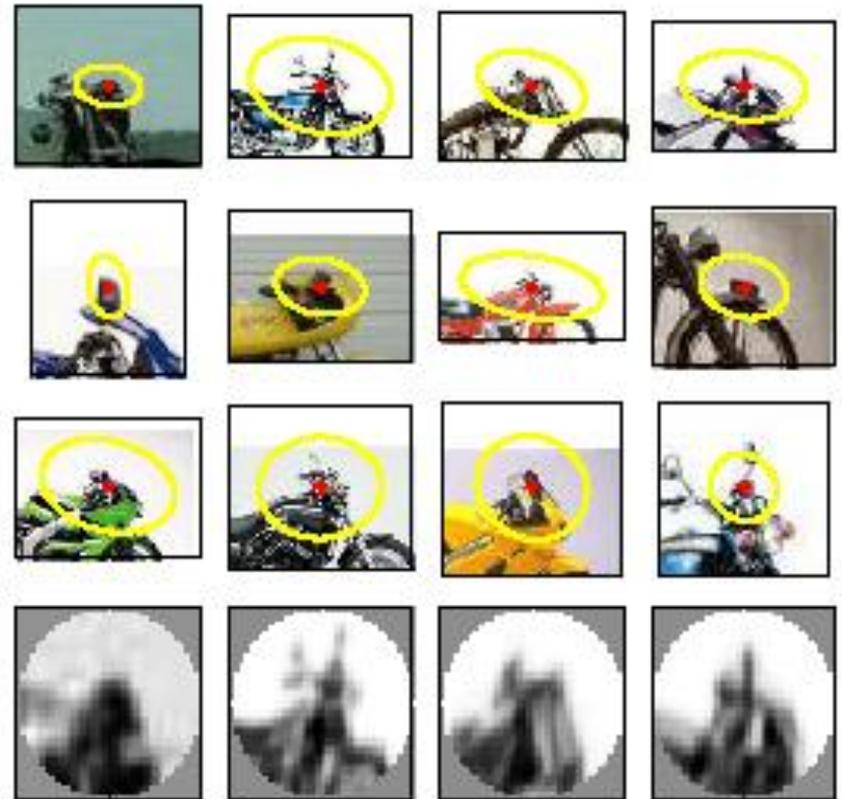
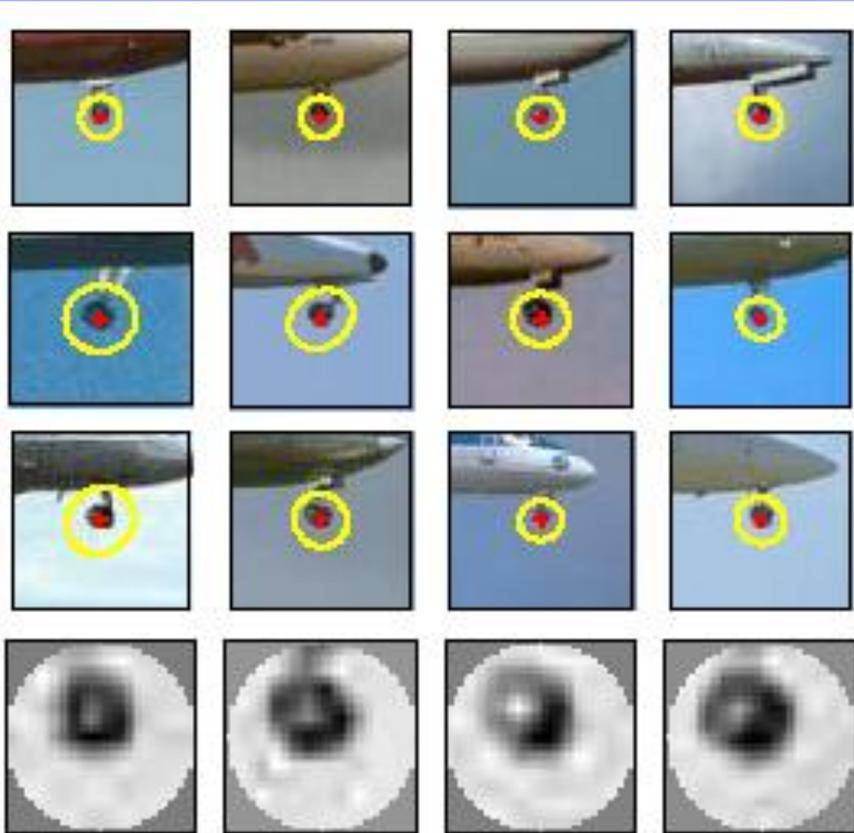
---

- 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



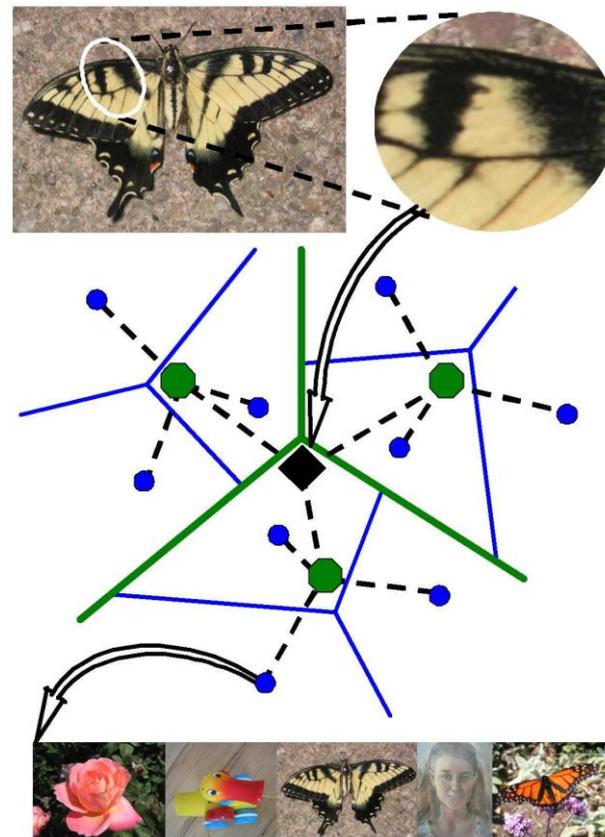
# Image patch examples of visual words



# 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)



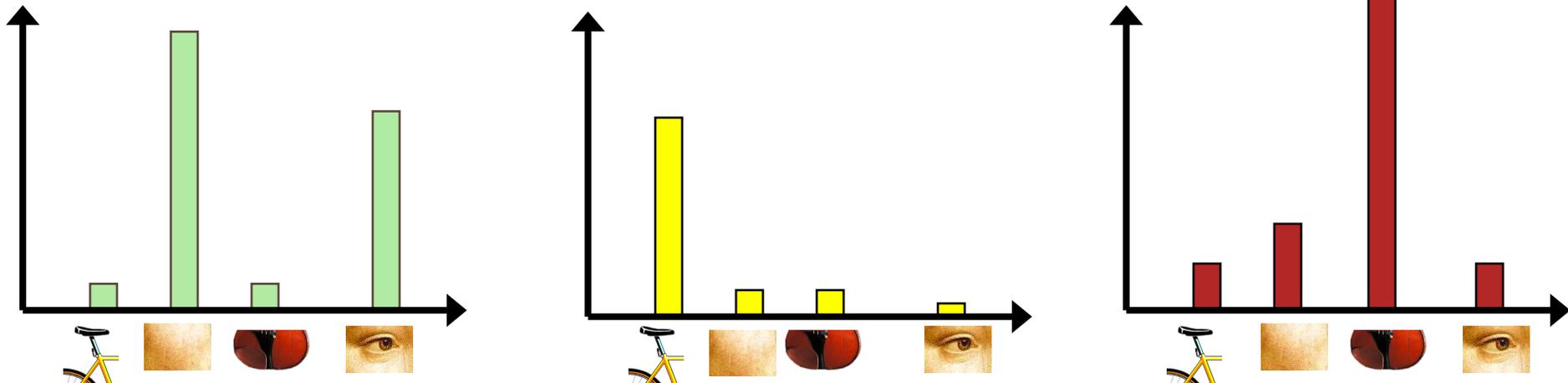
# 3. Image representation



# Image classification

---

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

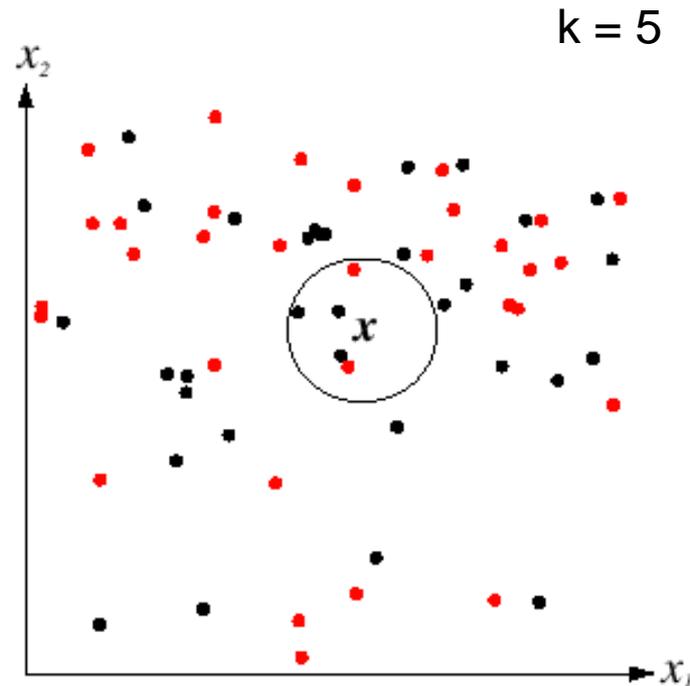


# Uses of BoW representation

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

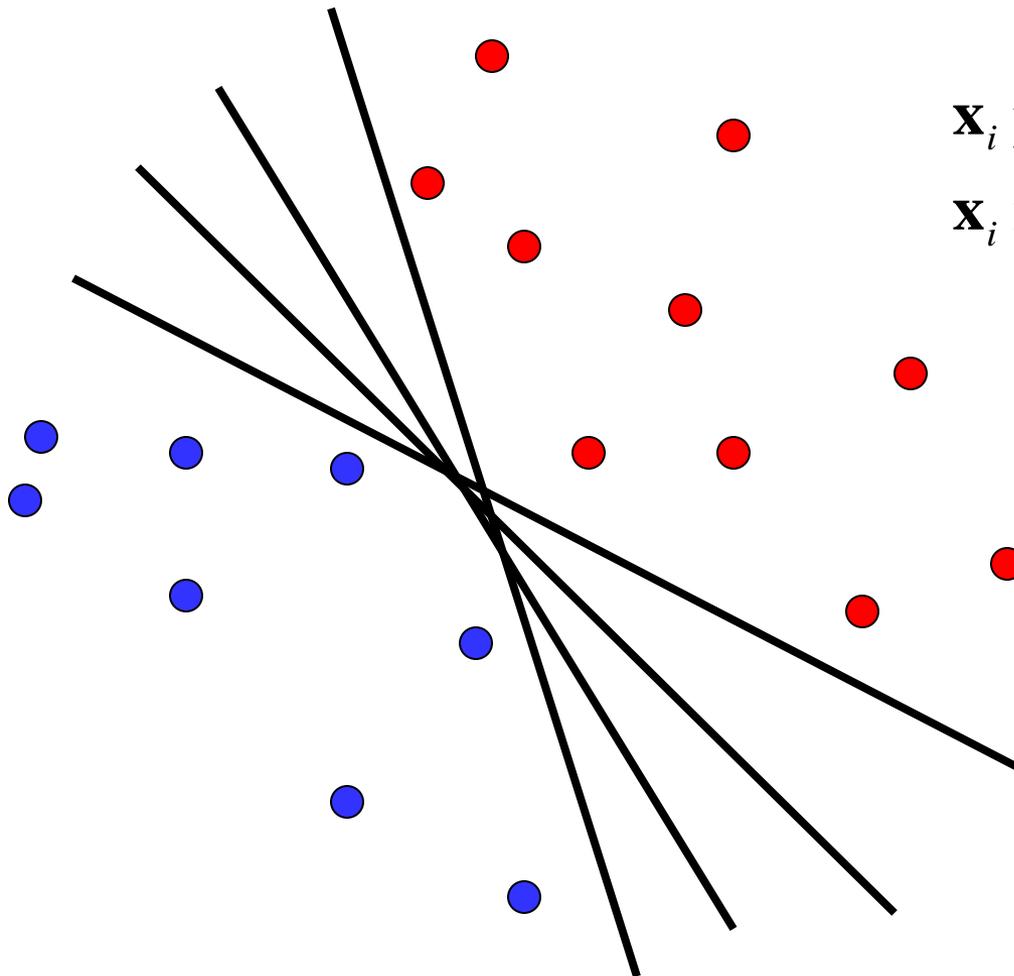
# K nearest neighbors

- For a new point, find the  $k$  closest points from training data
- Labels of the  $k$  points “vote” to classify
- Works well provided there is lots of data and the distance function is good



# Linear classifiers

- Find linear function (*hyperplane*) to separate positive and negative examples



$$\mathbf{x}_i \text{ positive: } \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$$

$$\mathbf{x}_i \text{ negative: } \mathbf{x}_i \cdot \mathbf{w} + b < 0$$

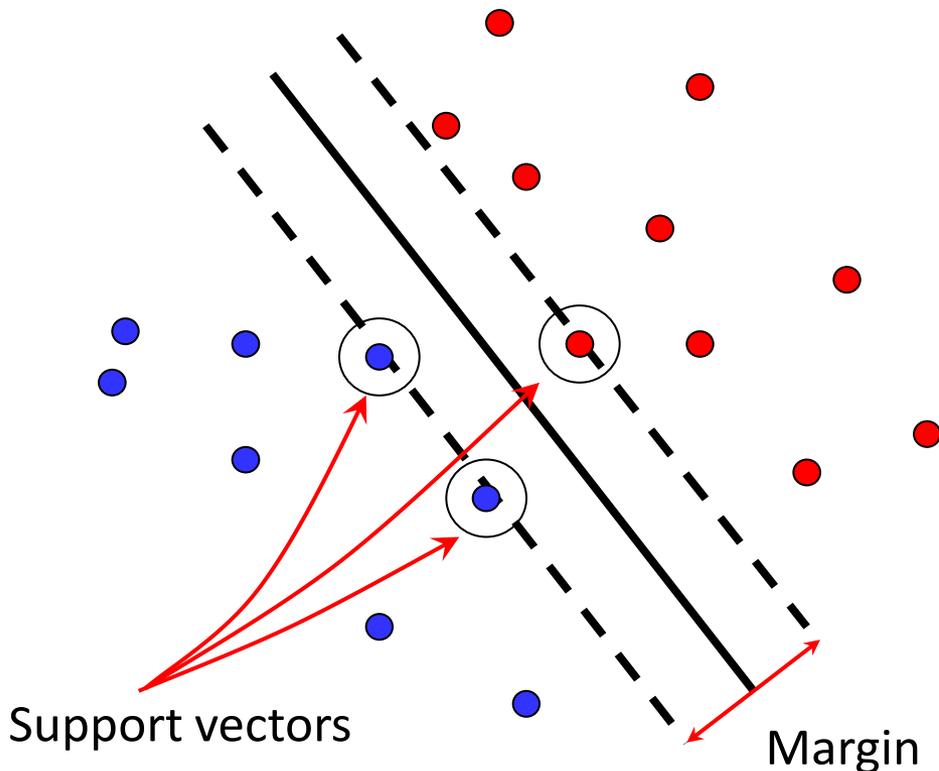
Which hyperplane  
is best?

# Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples

# Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples

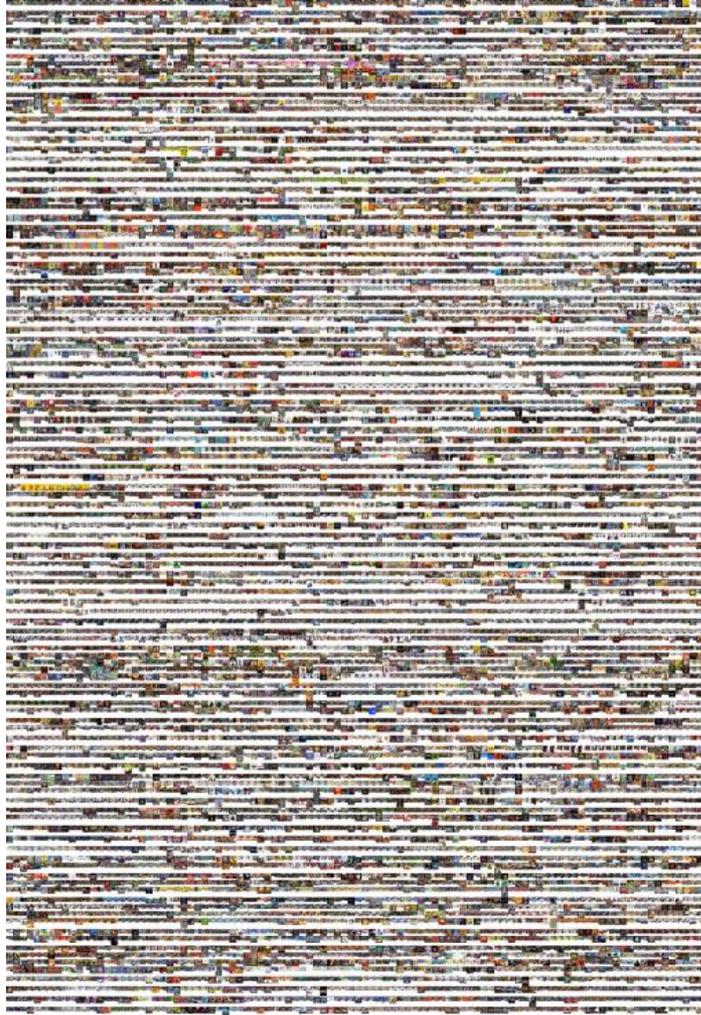


$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

$$\text{For support, vectors,} \quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$$

# 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 its *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  $l$ :

*term frequency of  $j$  in  $l$*  **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

query image



top 6 results



- 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

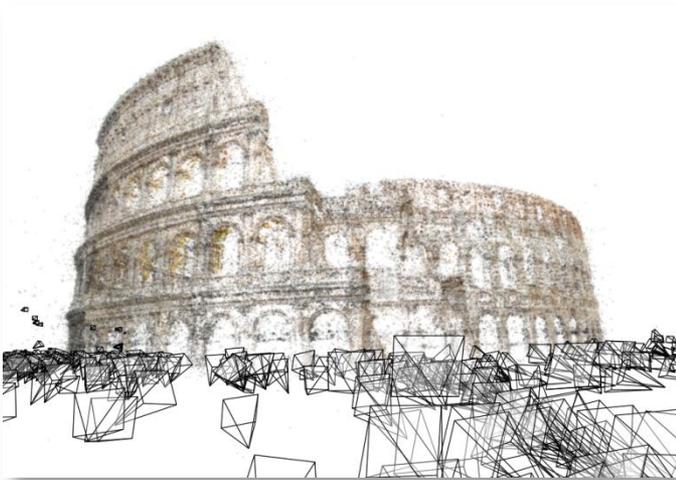


real-time retrieval from a database of 40,000 CD covers  
Nister & Stewenius, **Scalable Recognition with a Vocabulary Tree**

# Large-scale image matching

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

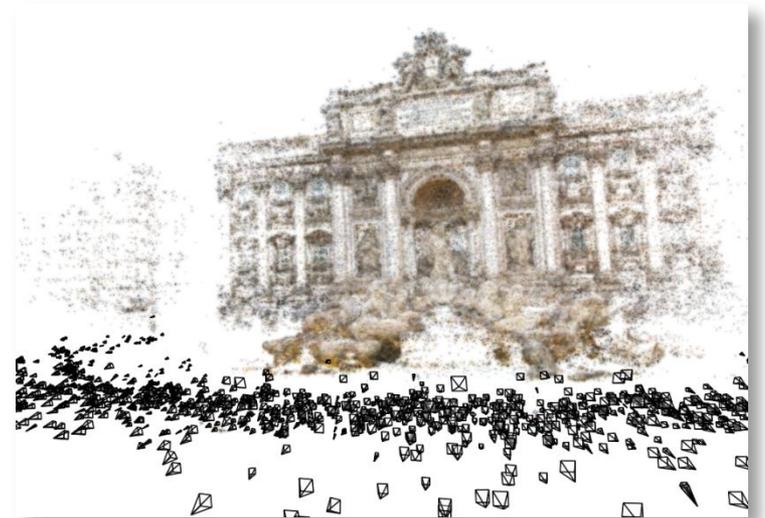
# ...into 3D models



Colosseum



St. Peter's Basilica

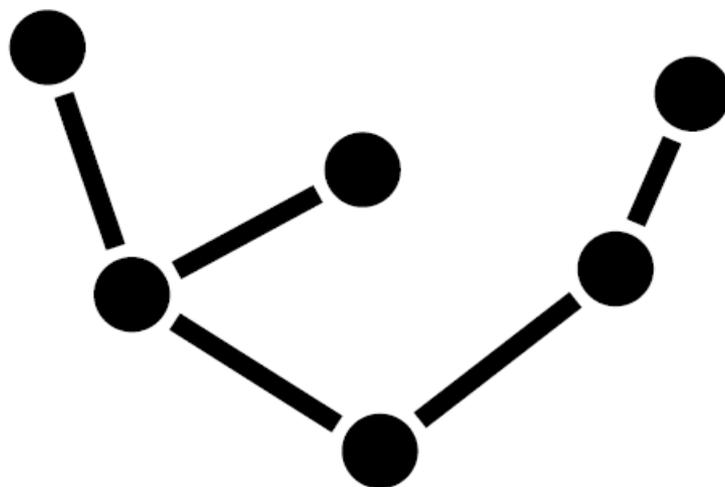


Trevi Fountain

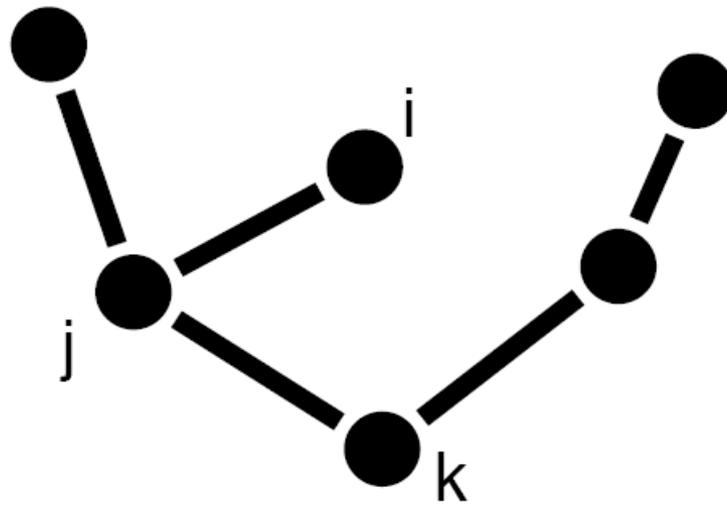
# Large-scale image matching

- 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

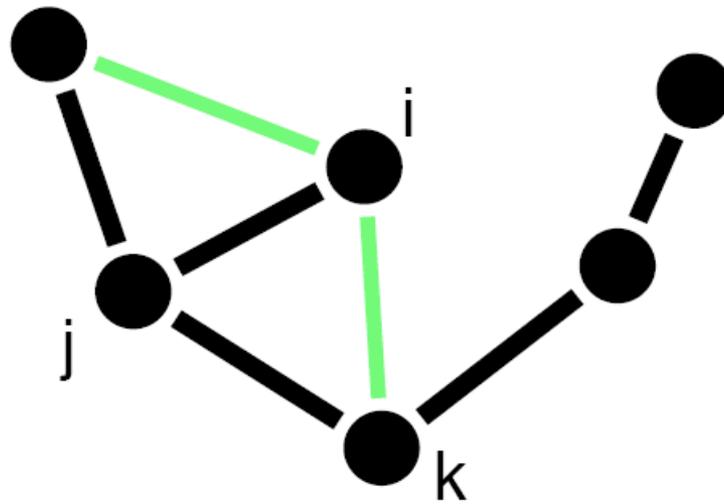
# Query Expansion



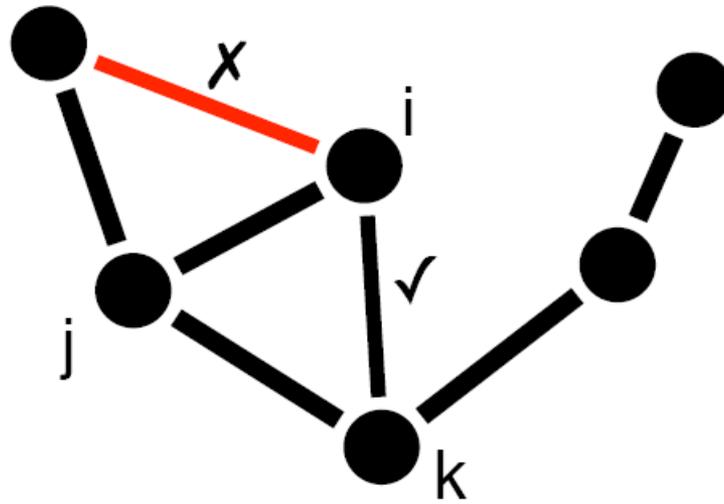
# Query Expansion



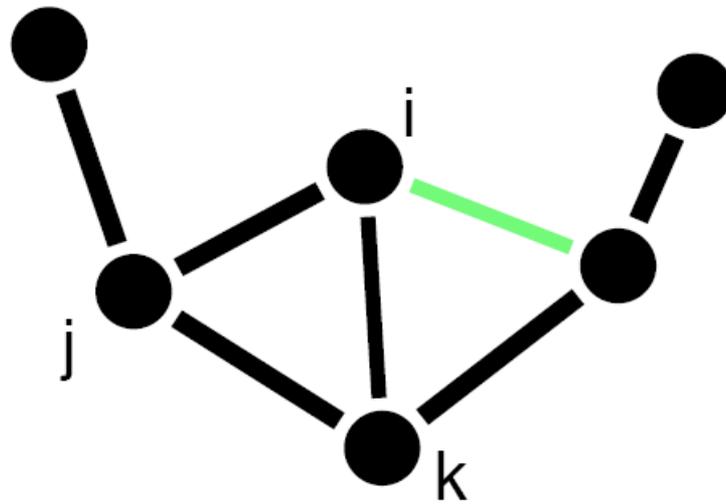
# Query Expansion



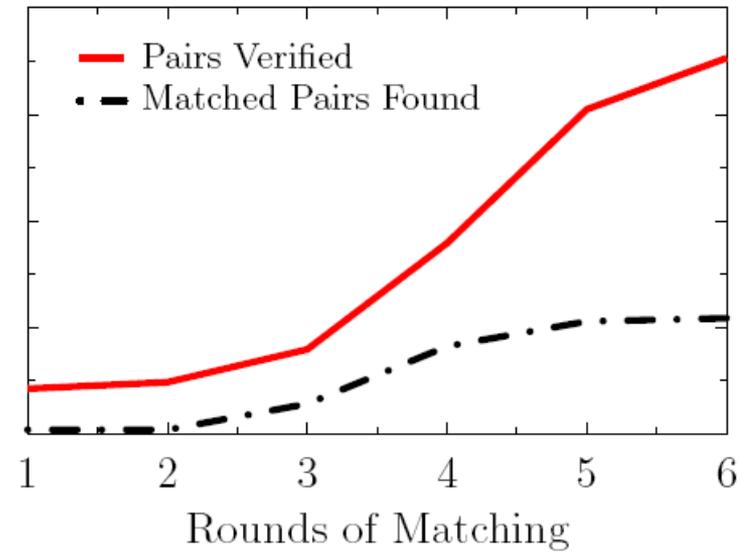
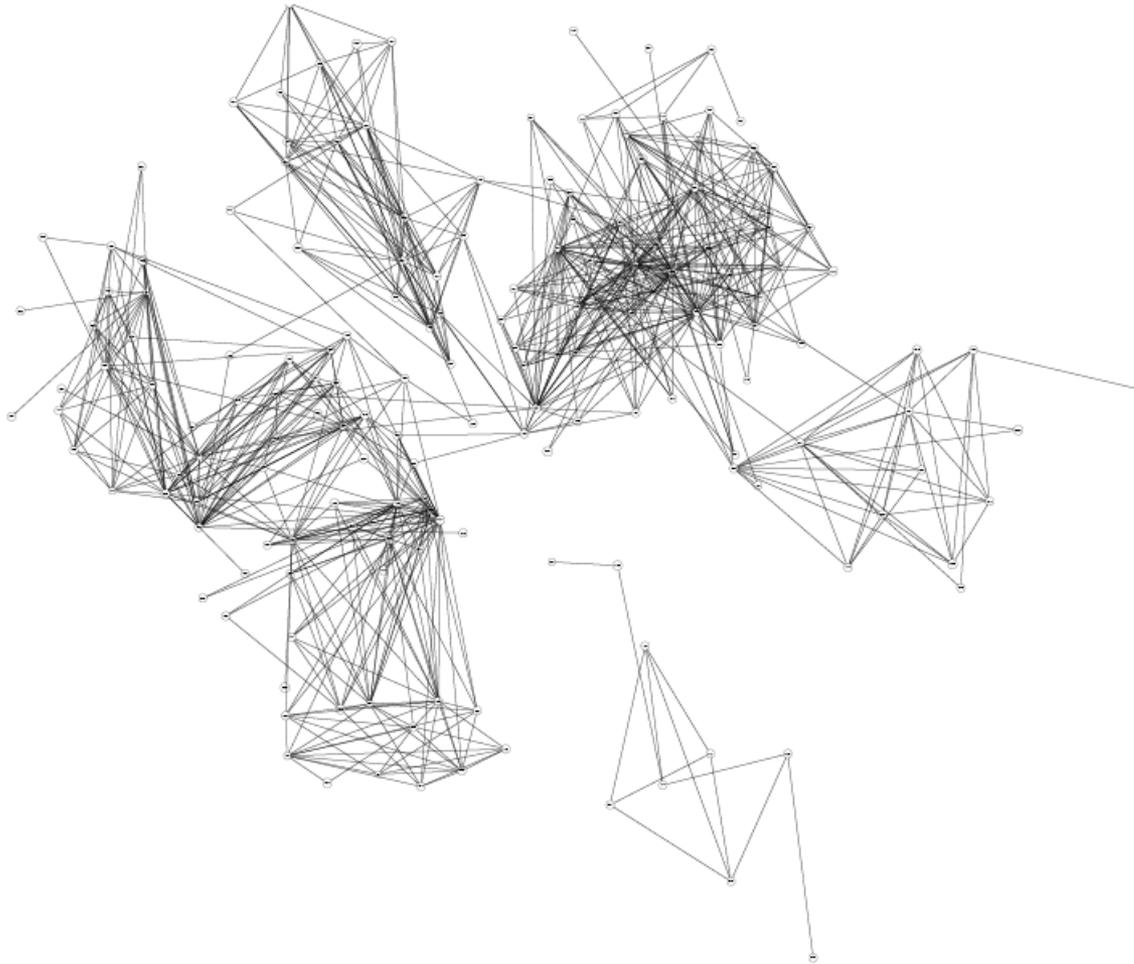
# Query Expansion



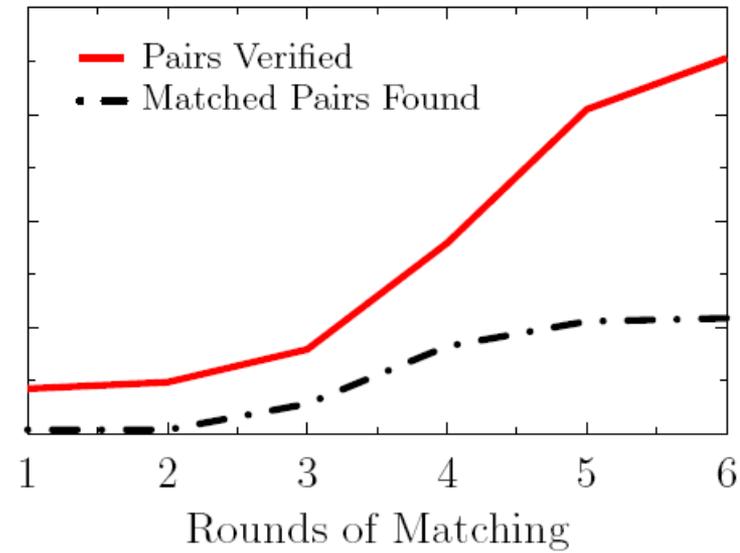
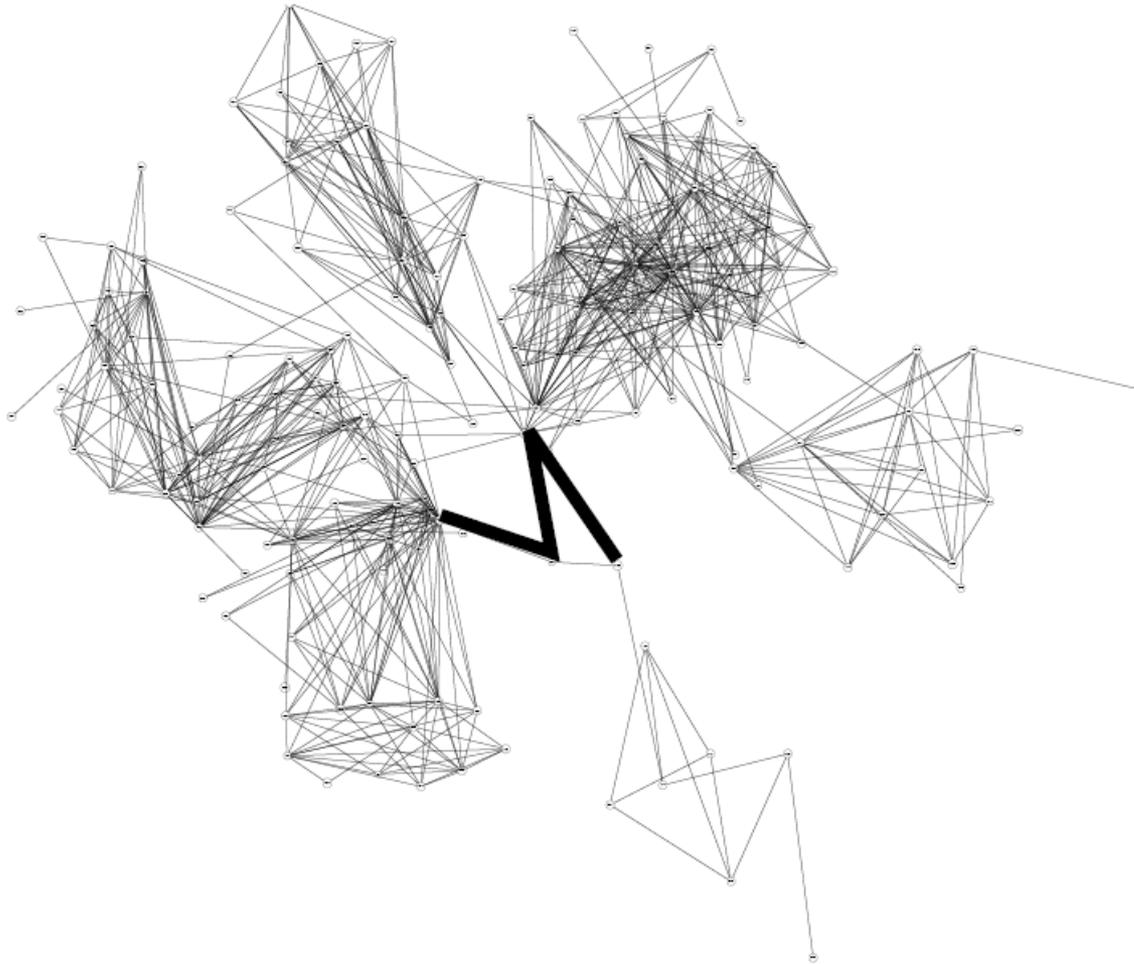
# Query Expansion



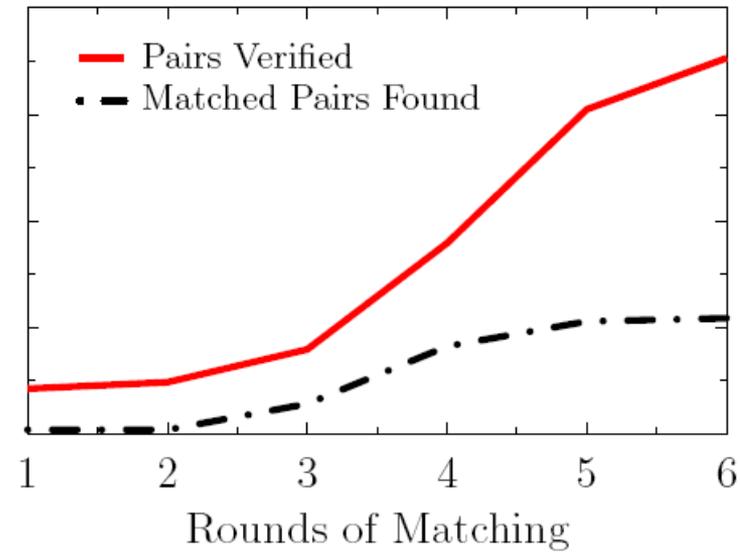
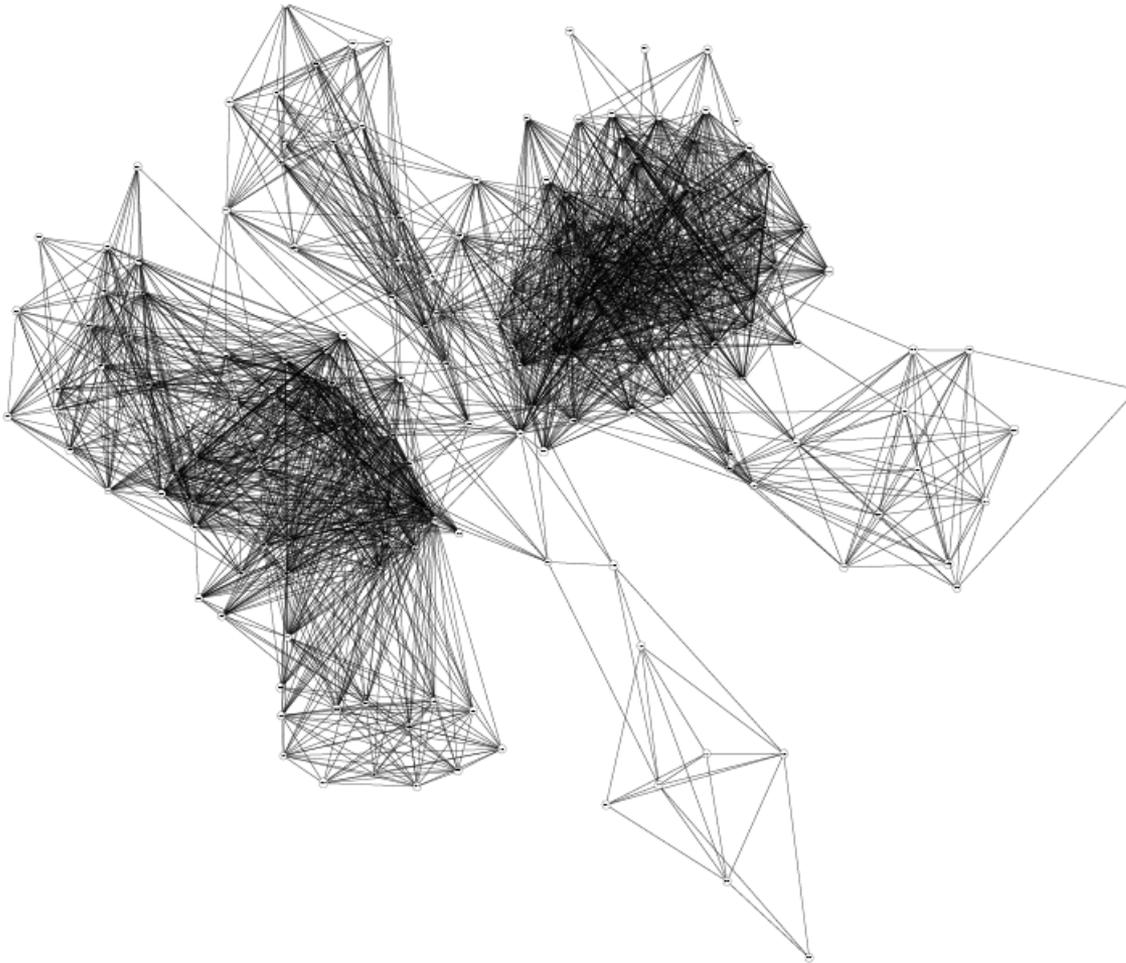
# Matching - Round 1



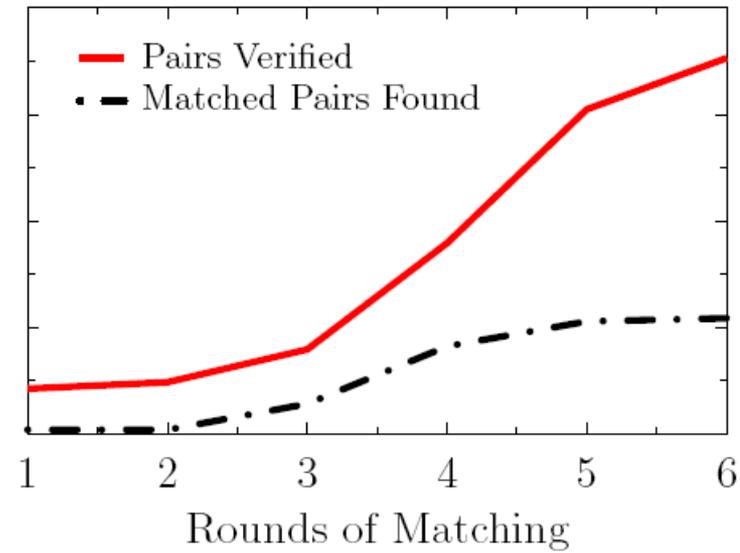
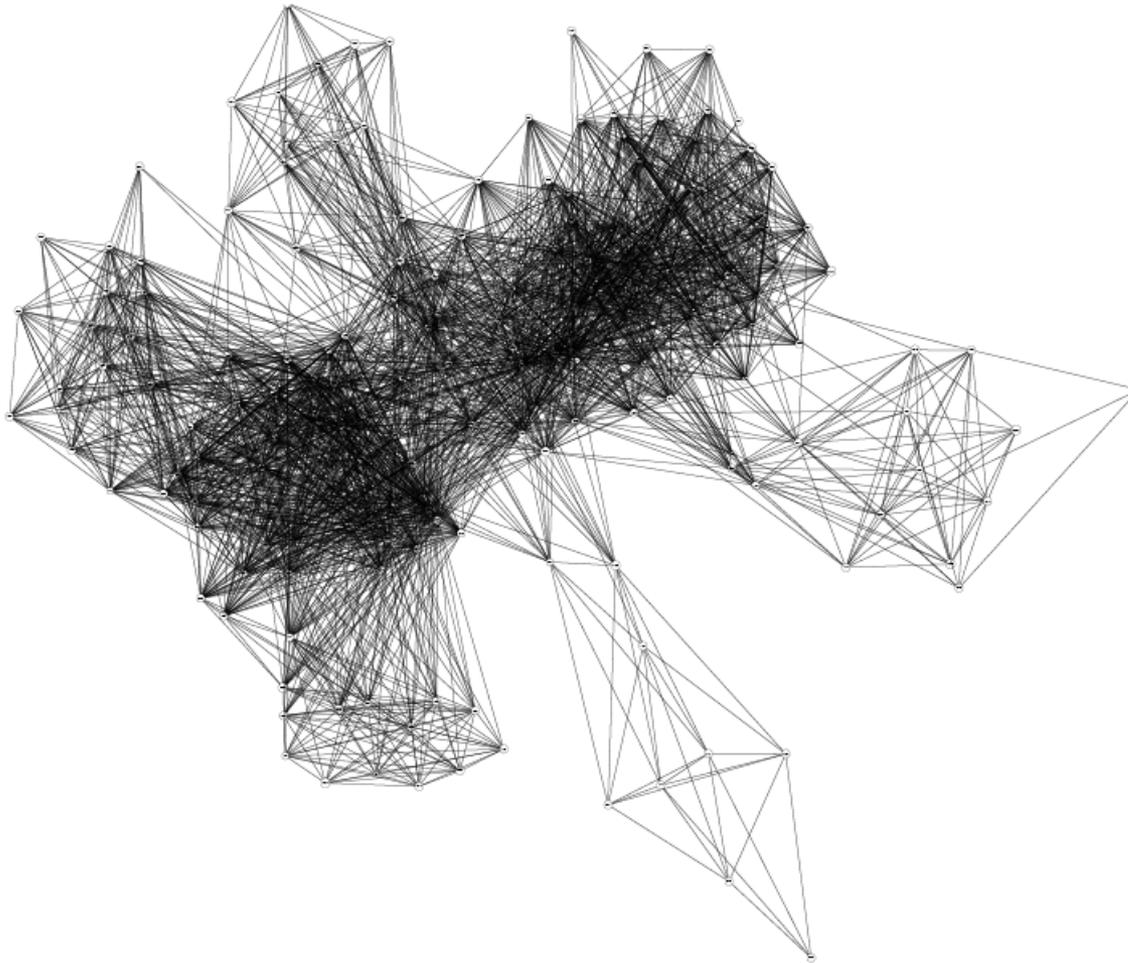
# Matching - Round 2



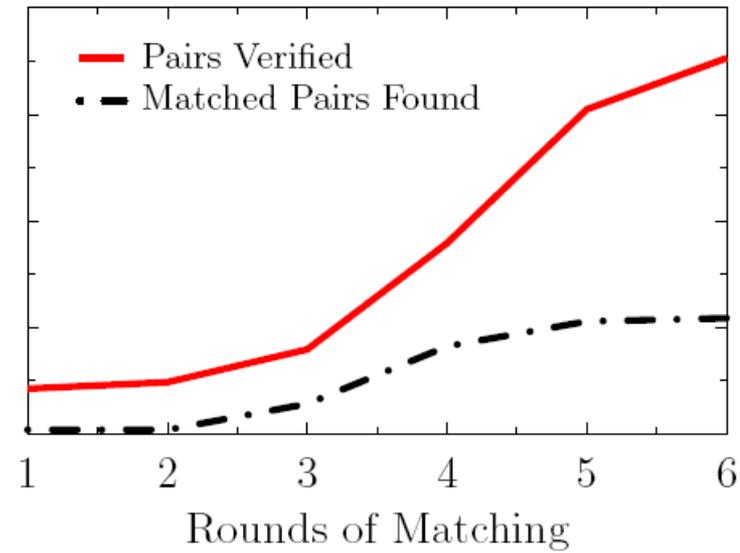
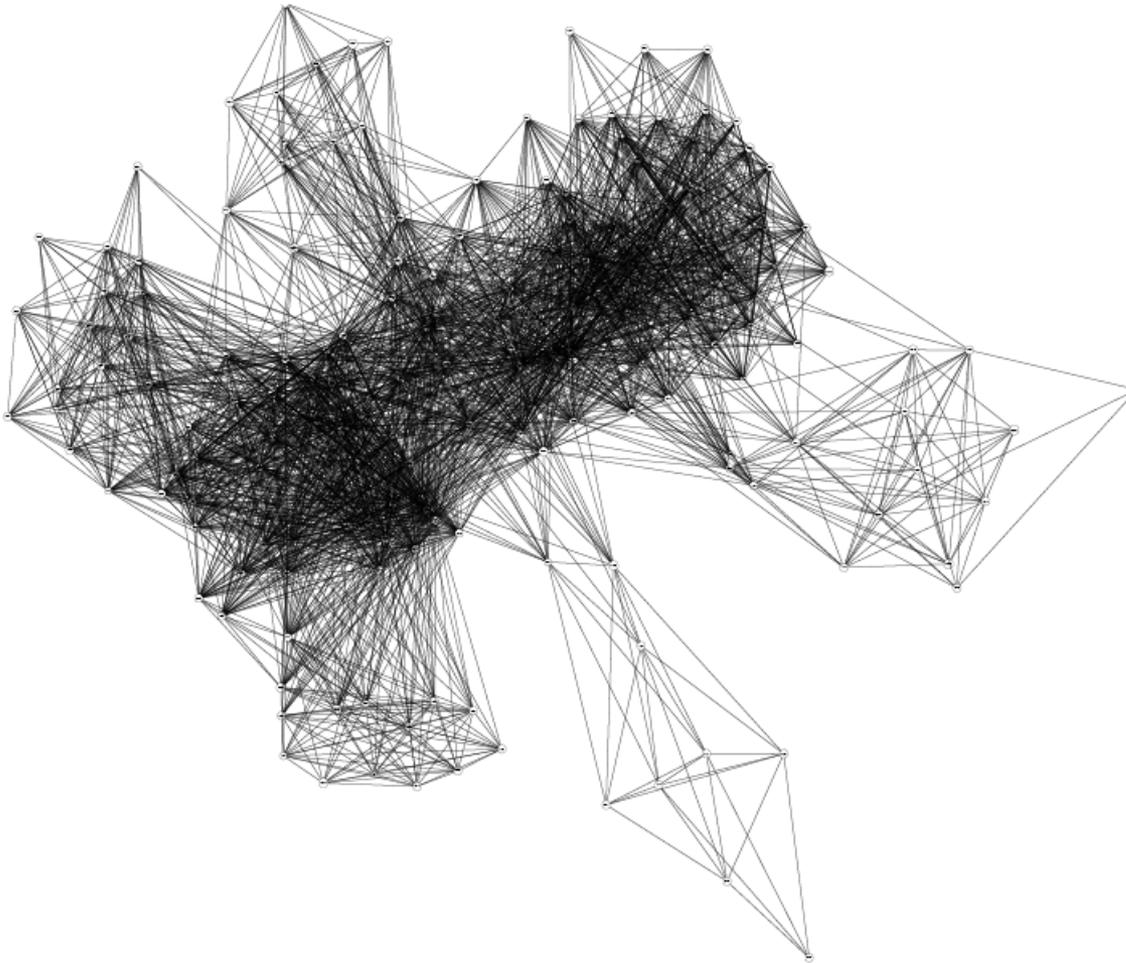
# Matching - Round 3



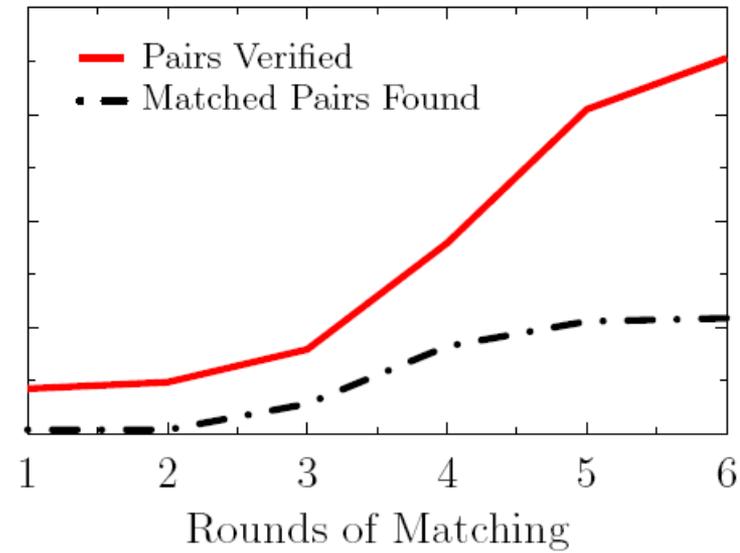
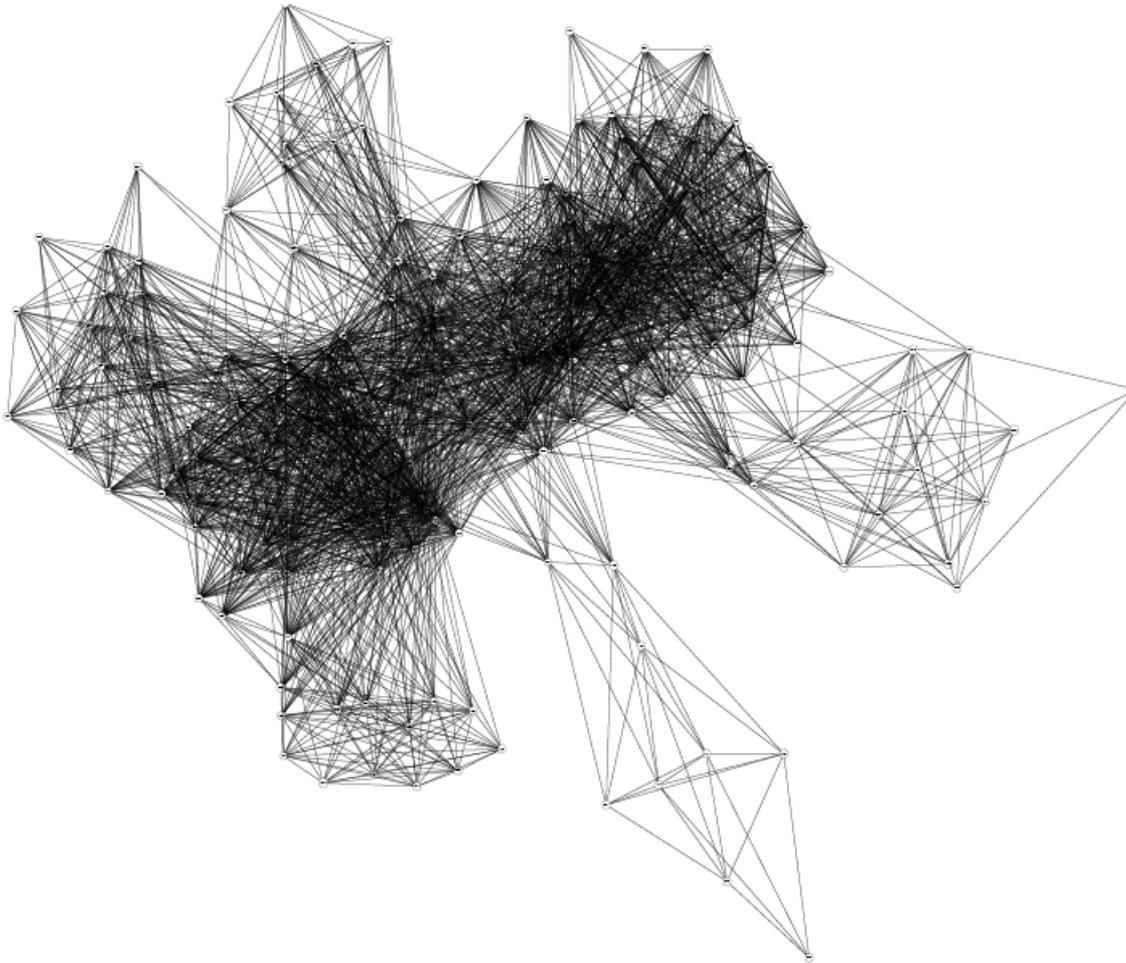
# Matching - Round 4



# Matching - Round 5



# Matching - Round 6



# Matching Statistics

<b>Dataset</b>	<b>Size</b>	<b>Matches possible</b>	<b>Matches Tried</b>	<b>Matches Found</b>	<b>Time</b>
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

# What about spatial info?

