University of Nottingham
UK | CHINA | MALAYSIA

COMP3065
Computer Vision

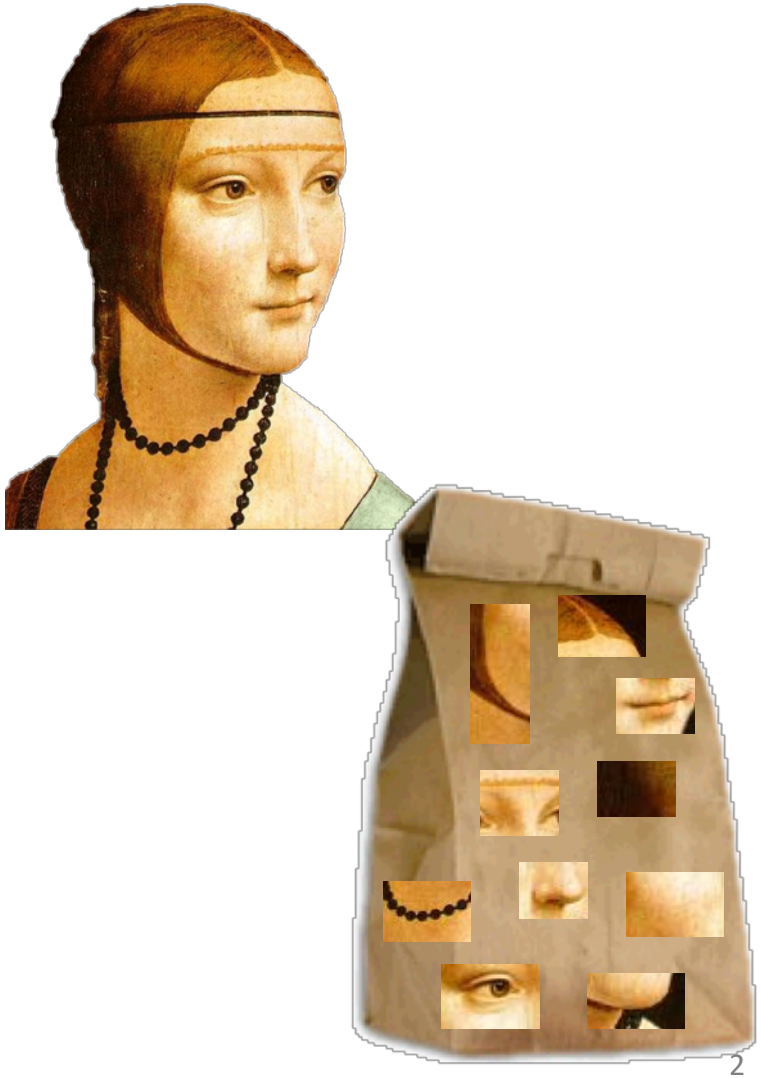
Topic 11 – Bag of Features

Dr. Zheng LU

2019 Spring

Bags of Features

- Some features are obviously good representations of some objects e.g. HoGs and people.
- Sometimes its not clear what features should be used.
- Bag of Features methods analyse the large set of very specific features generated by a training set of images and identify a small set of useful, more generic features.



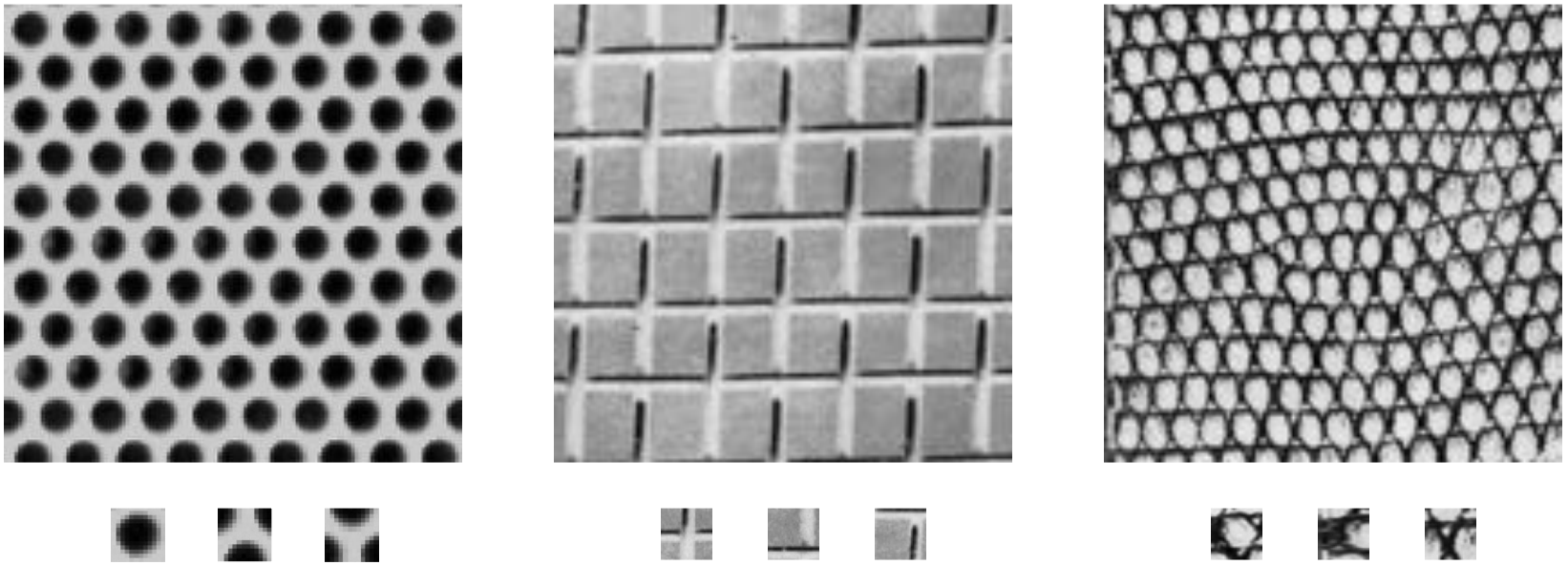
Origin 1: Bag-of-Words Models

- Orderless document representation: frequencies of words from a dictionary.

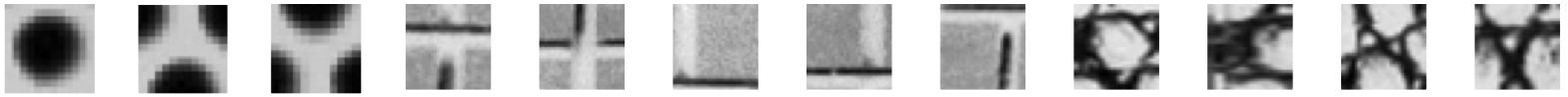
US Presidential Speeches Tag Cloud
http://chir.ag/phernalia/preztags/

Origin 2: Texture Recognition

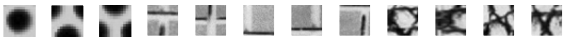
- Texture is characterized by the repetition of basic elements or *textons*
- For stochastic textures, it is the identity of the textons, not their spatial arrangement, that matters



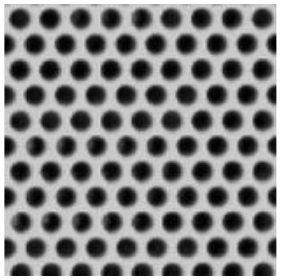
Origin 2: Texture recognition

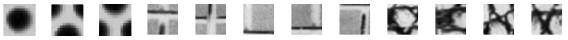


Origin 2: Texture recognition

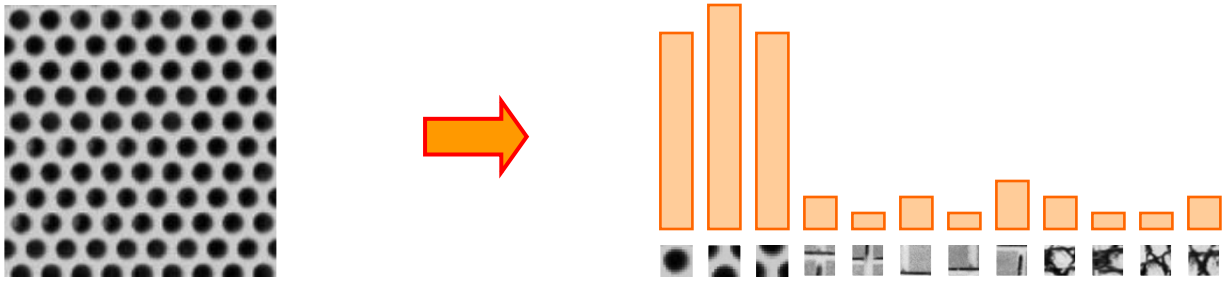


Origin 2: Texture recognition

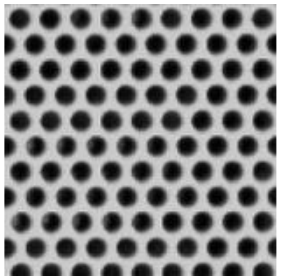


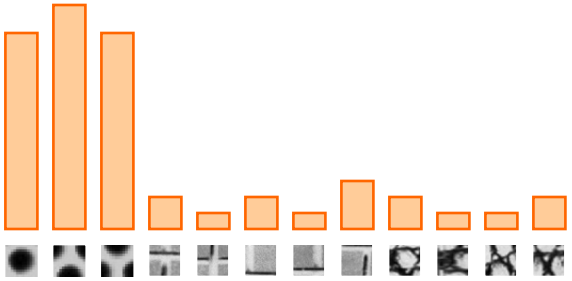


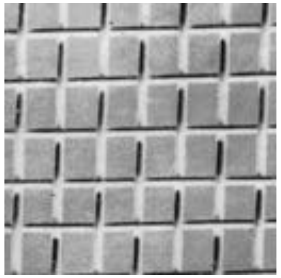
Origin 2: Texture recognition



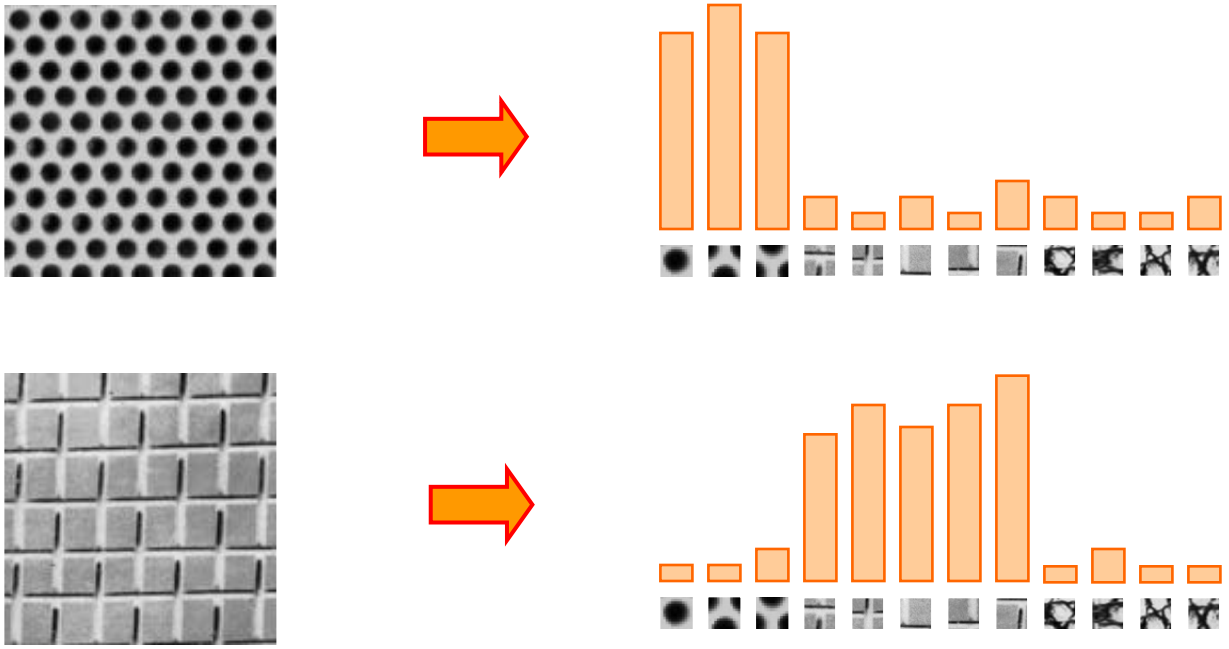
Origin 2: Texture recognition



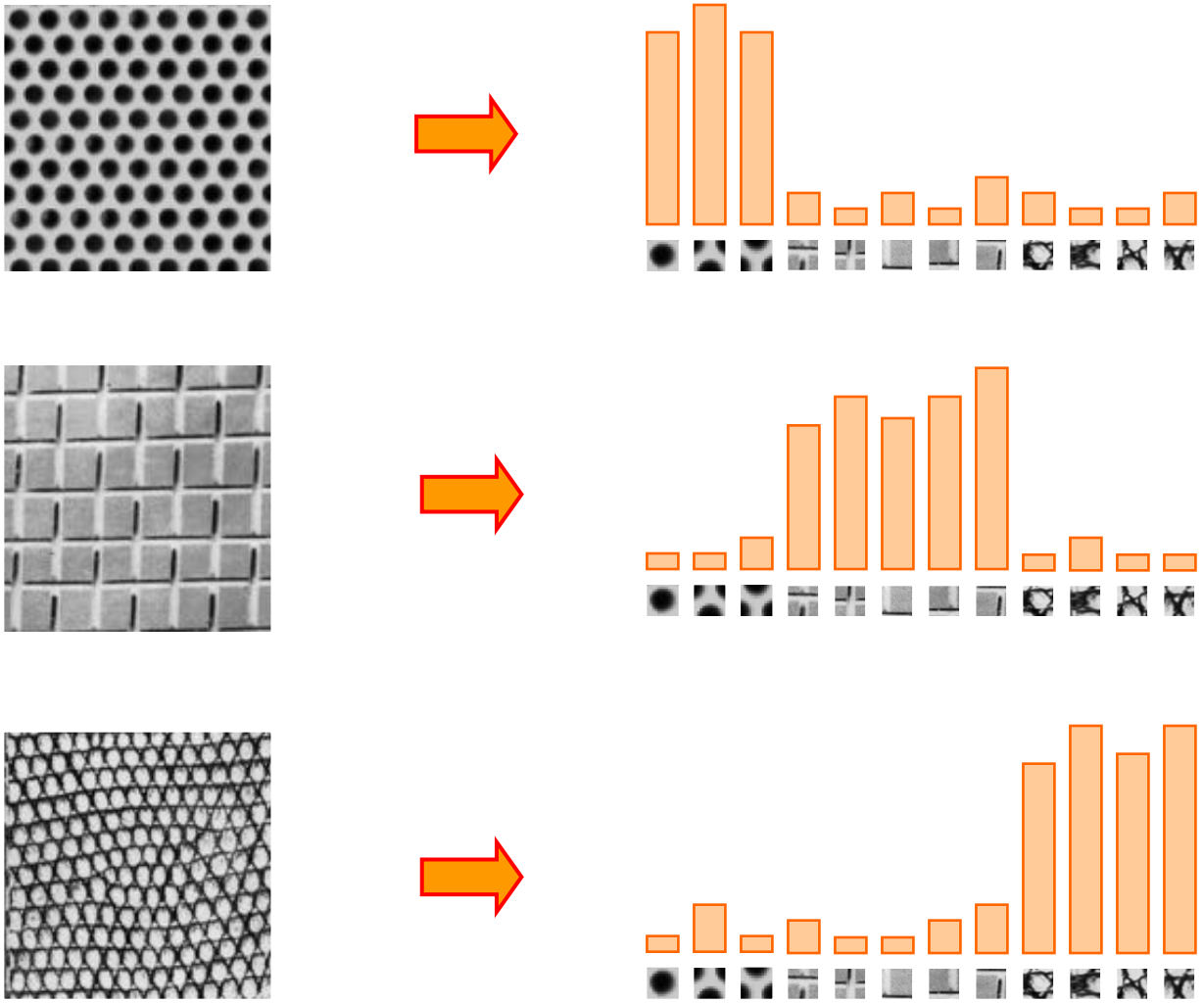




Origin 2: Texture recognition



Origin 2: Texture recognition



Bags of Features for Object Recognition

- 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
  - For each feature, find the closest visual word in the dictionary.
  - Build a histogram to represent the image.

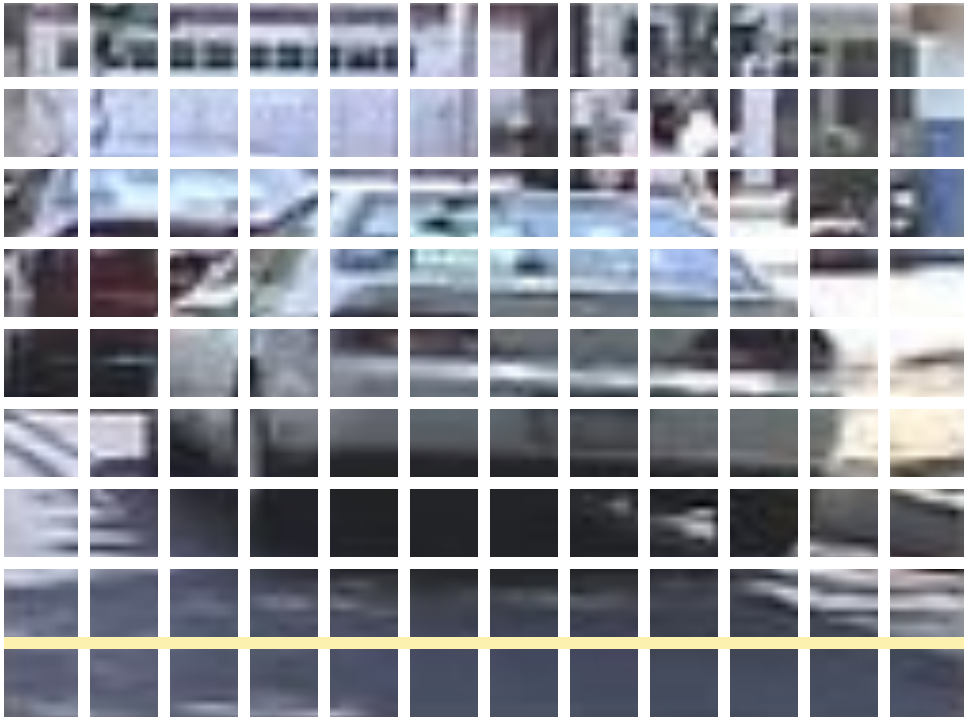
Bags of Features for Object Recognition

- 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
  - For each feature, find the closest visual word in the dictionary.
  - Build a histogram to represent the image.

face, flowers, building

Feature Extraction

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



Feature Extraction

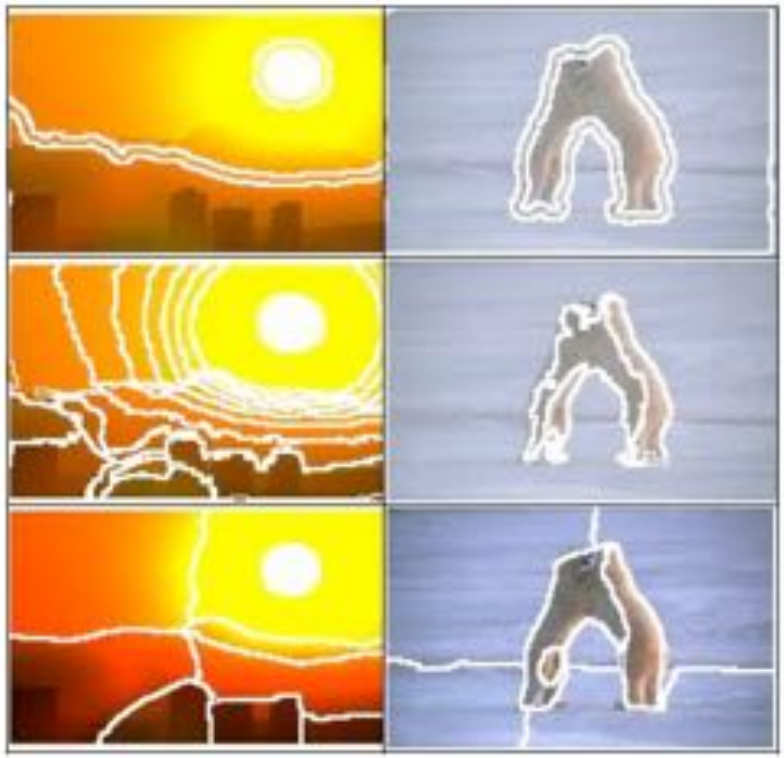
- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic et al. 2005

Feature Extraction

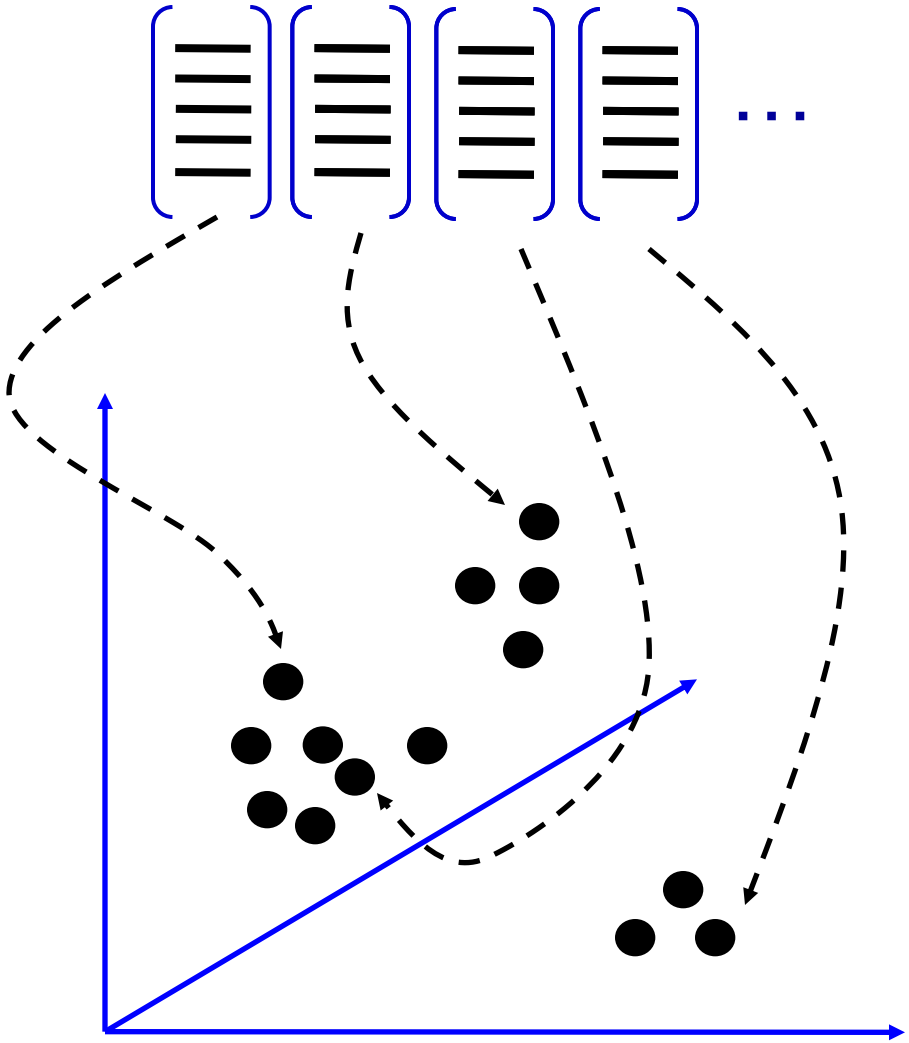
- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic et al. 2005

Feature Extraction

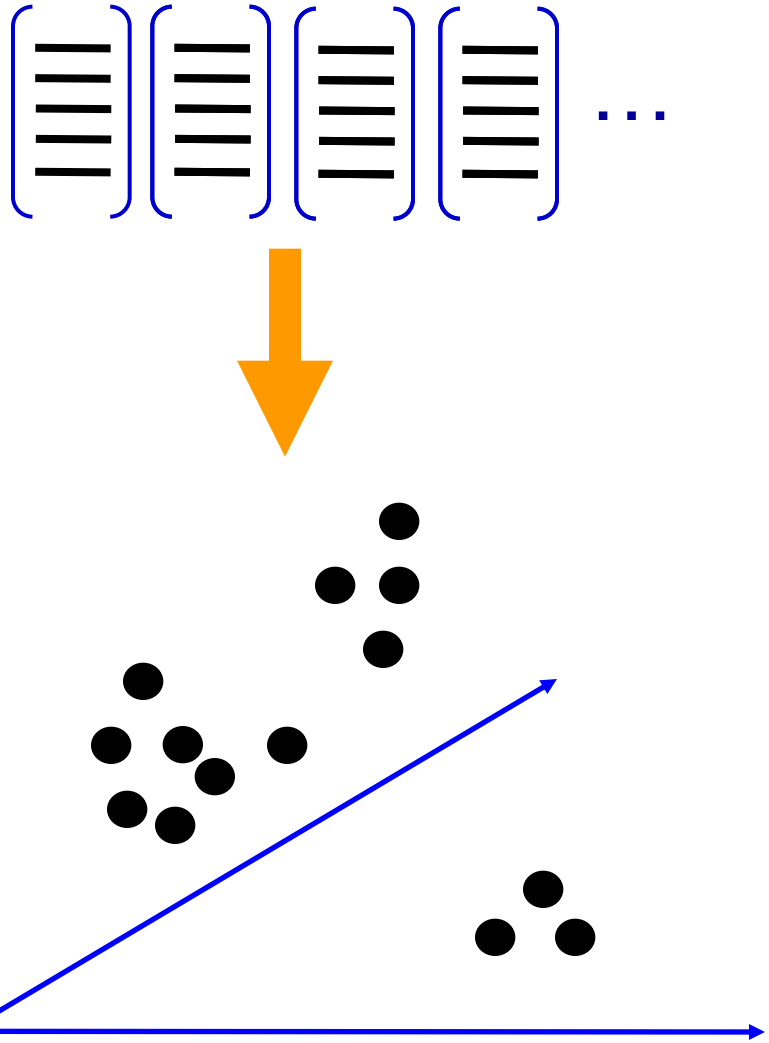
- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic et al. 2005
- Other methods
  - Random sampling
    (Vidal-Naquet & Ullman, 2002)
  - Segmentation-based patches
    (Barnard et al. 2003)



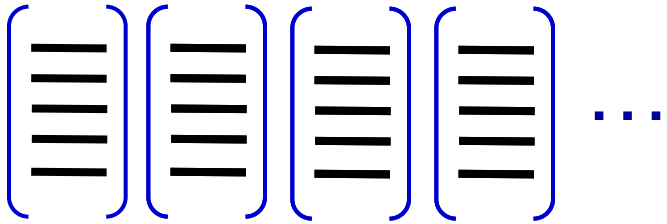
Learning Visual Vocabulary



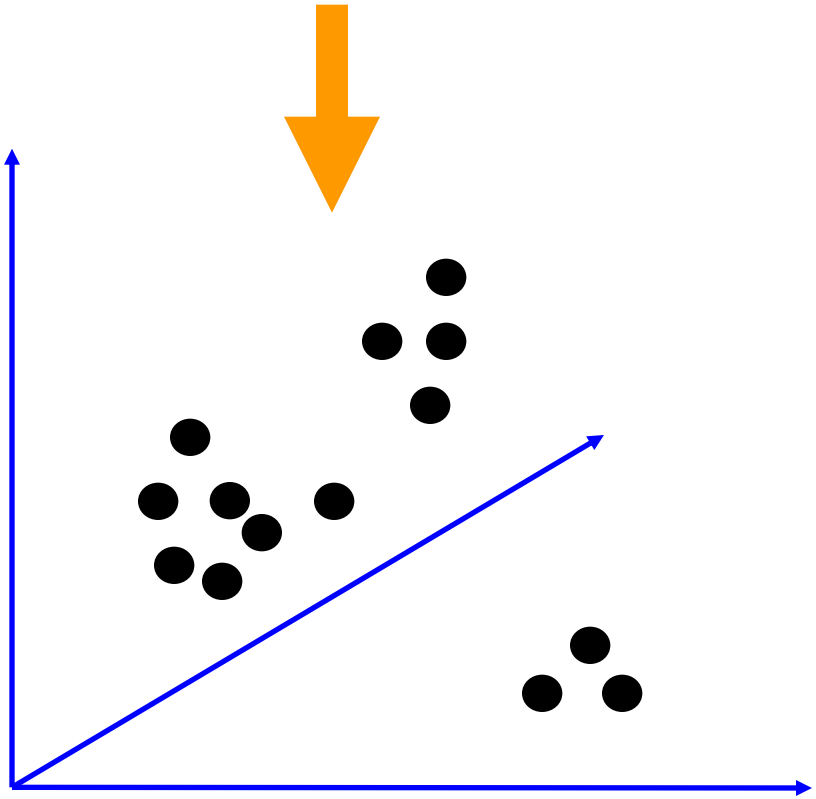
Learning Visual Vocabulary

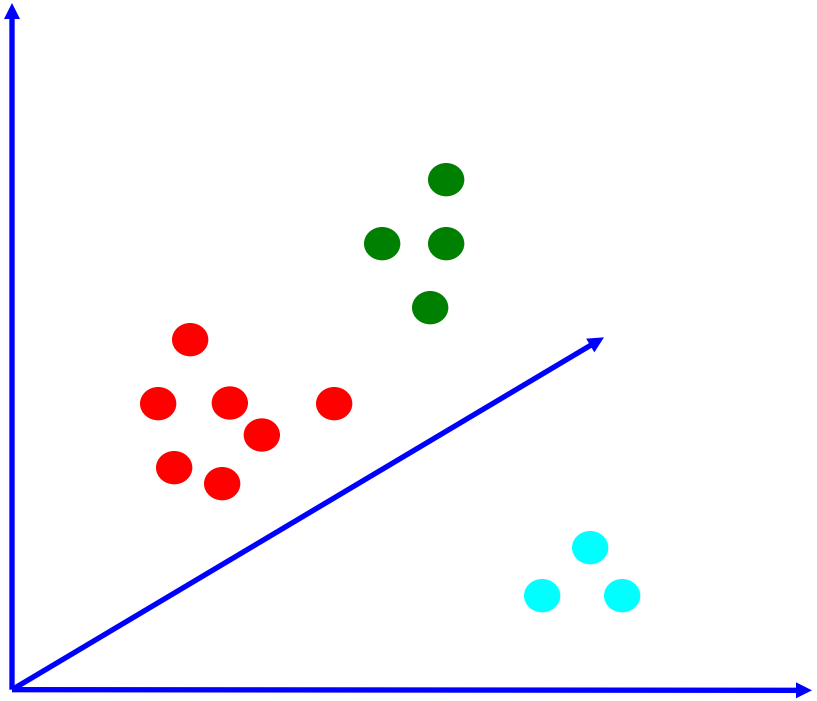


Learning Visual Vocabulary

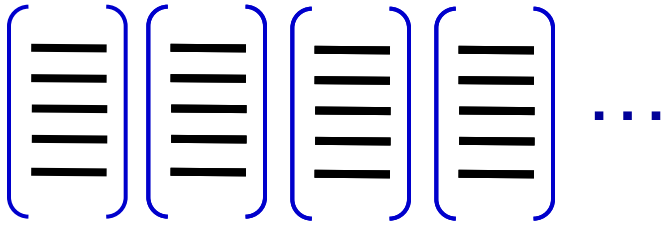


…

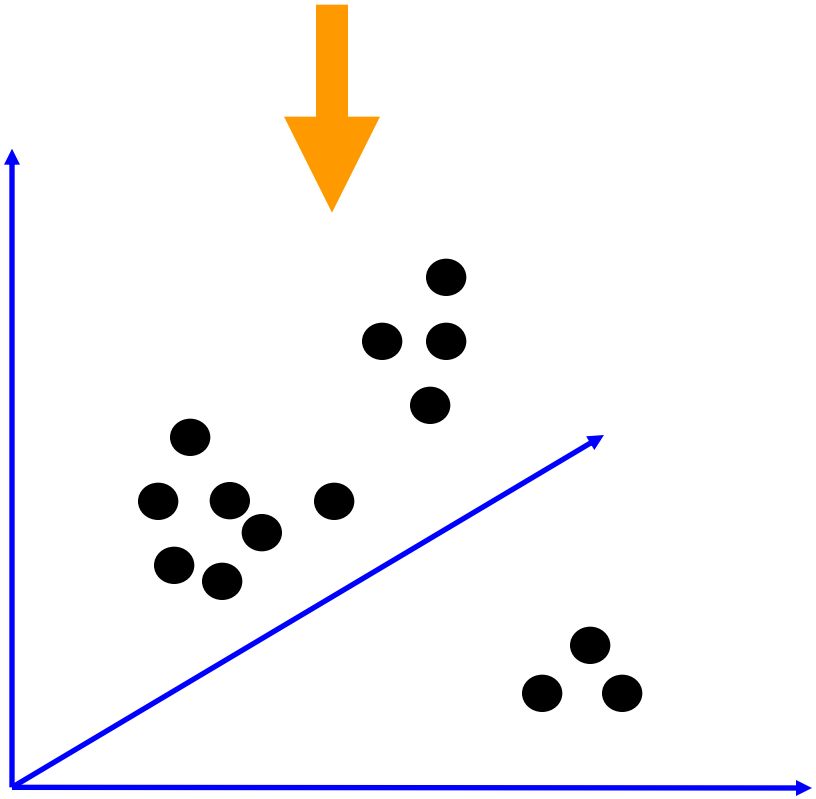


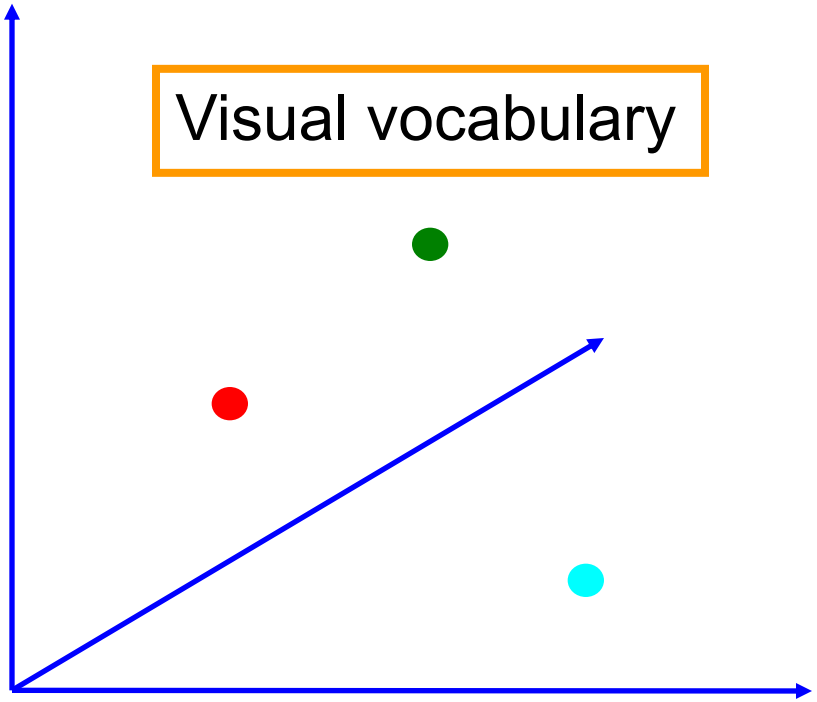


Learning Visual Vocabulary



…



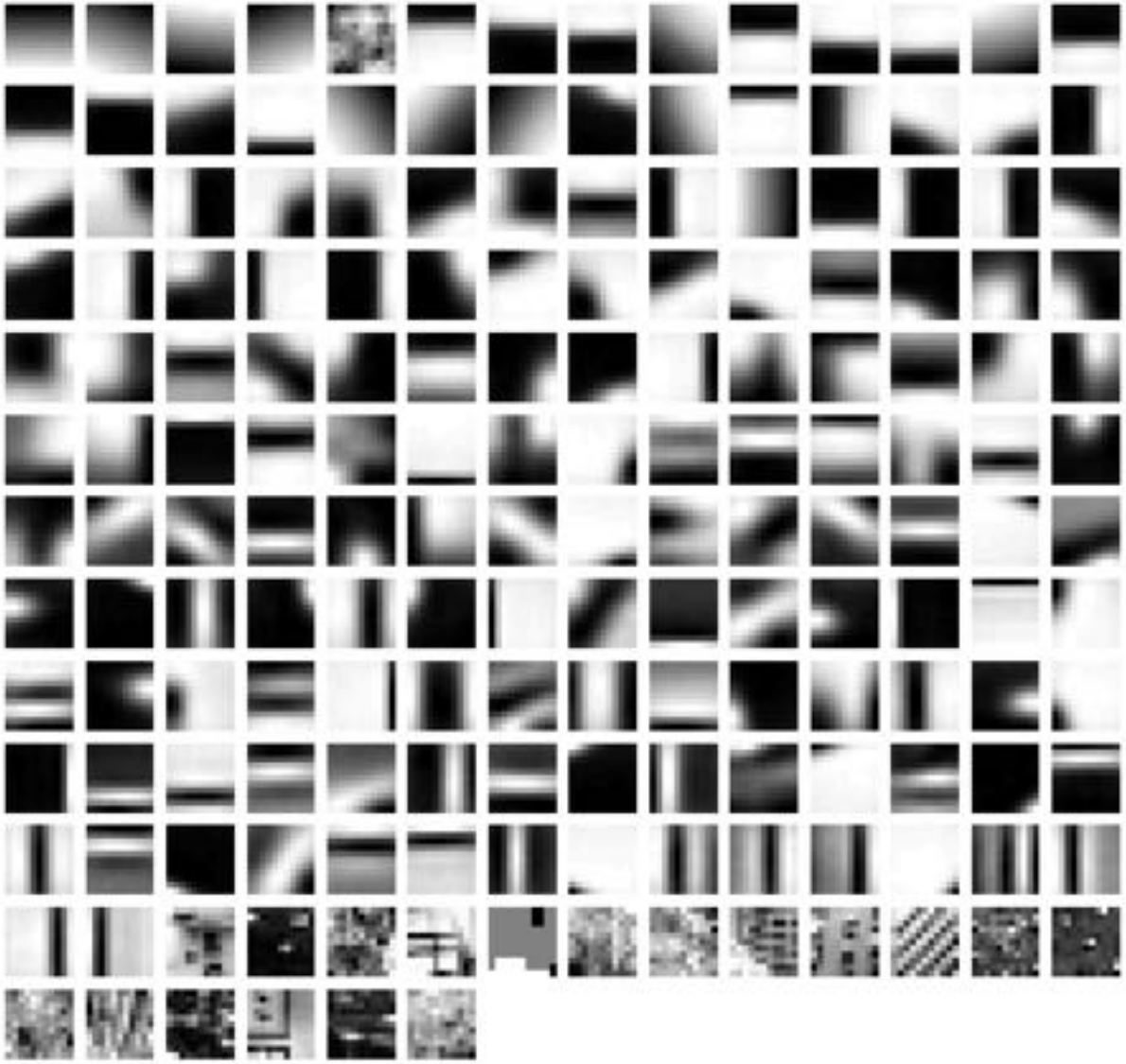


Clustering

K-means Clustering

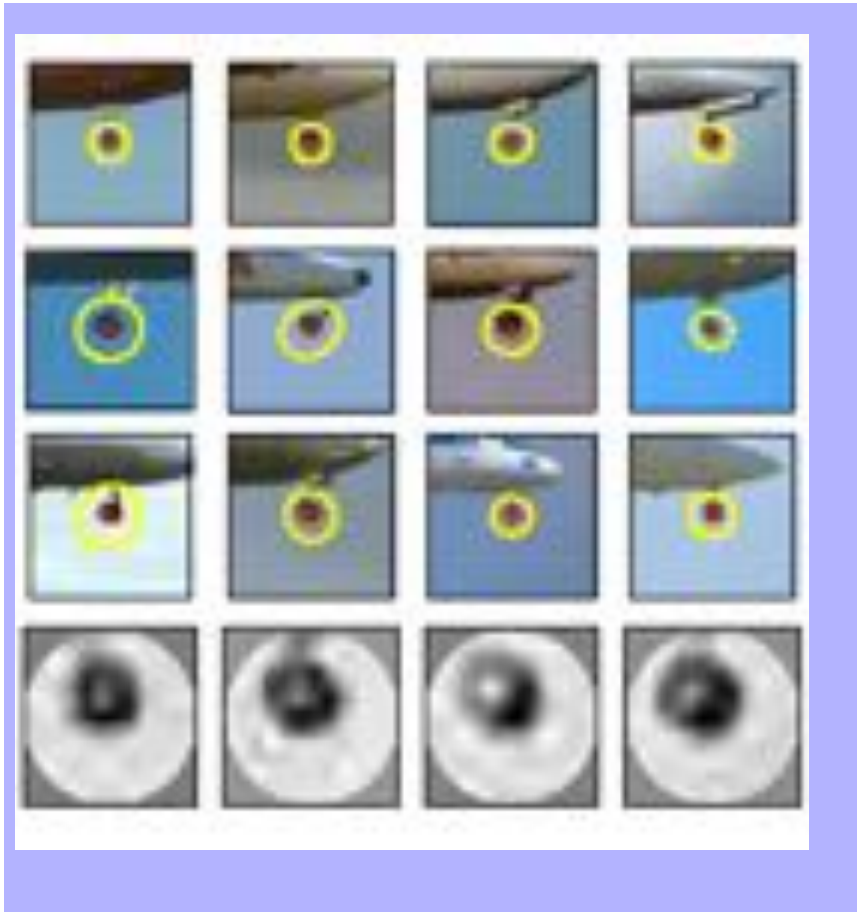
- Minimize distances between points and their nearest cluster centers: allows user to choose vocabulary size.
- Algorithm
  - Randomly initialize K cluster centers.
  - Iterate until convergence:
  - Assign each data point to the nearest center.
  - Recompute each cluster center as the mean of all points assigned to it.
- Cluster centers form a *codebook*
  - Provided the training set is sufficiently representative, the codebook will be "universal"
  - New feature points are mapped to the nearest cluster center (*codevector*)

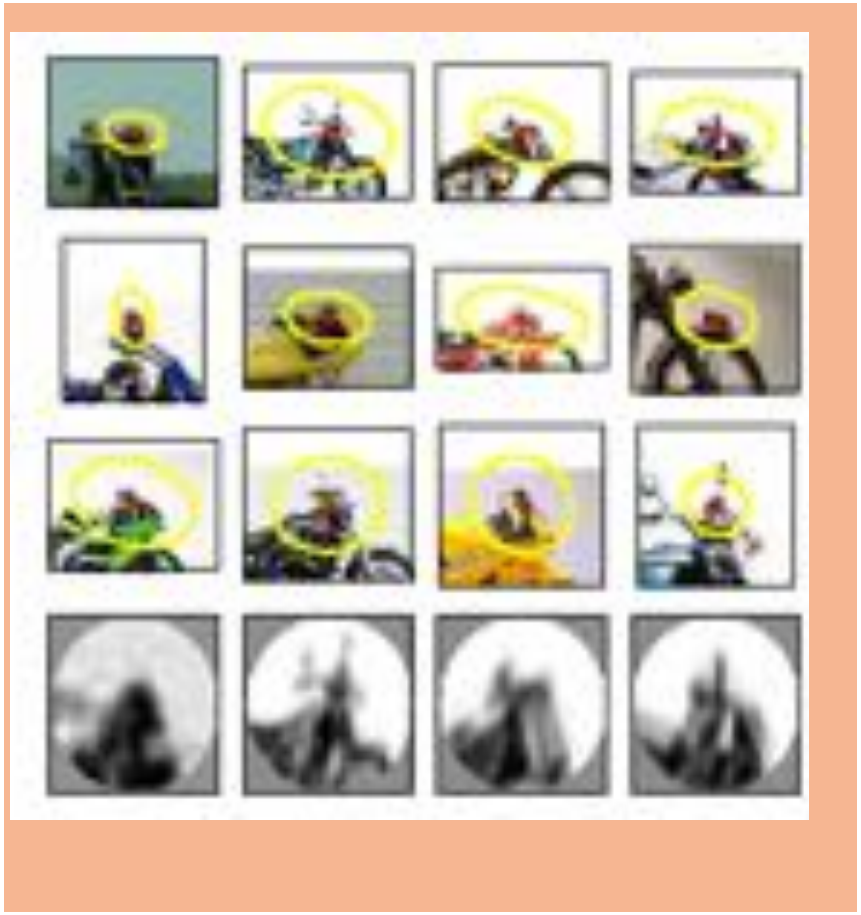
Visual Words



Fei-Fei et al. 2005

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.*
  - First map feature to nearest top level (green) codevector.
  - Then map feature to nearest (blue) child of chosen codevector.
  - Allows efficient, structured use of larger vocabularies.

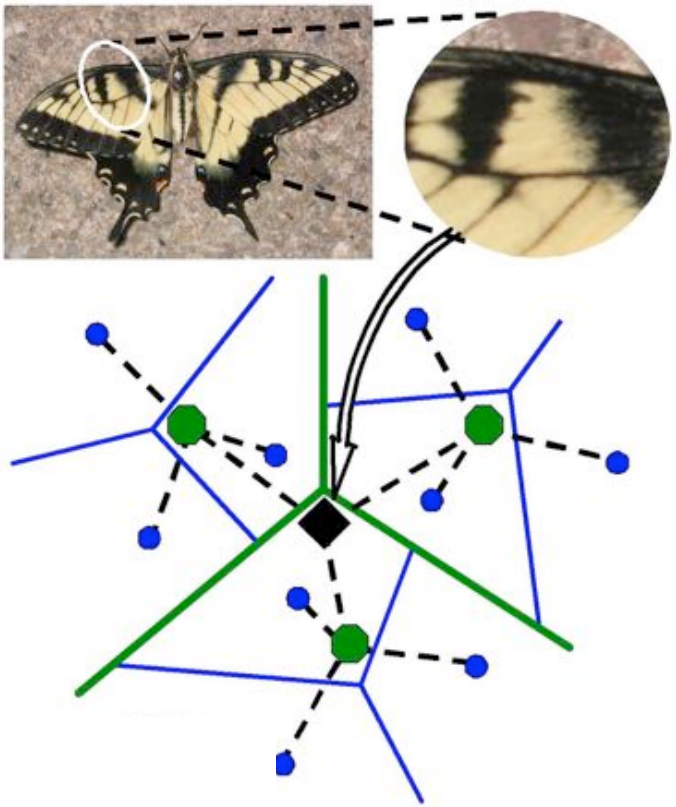
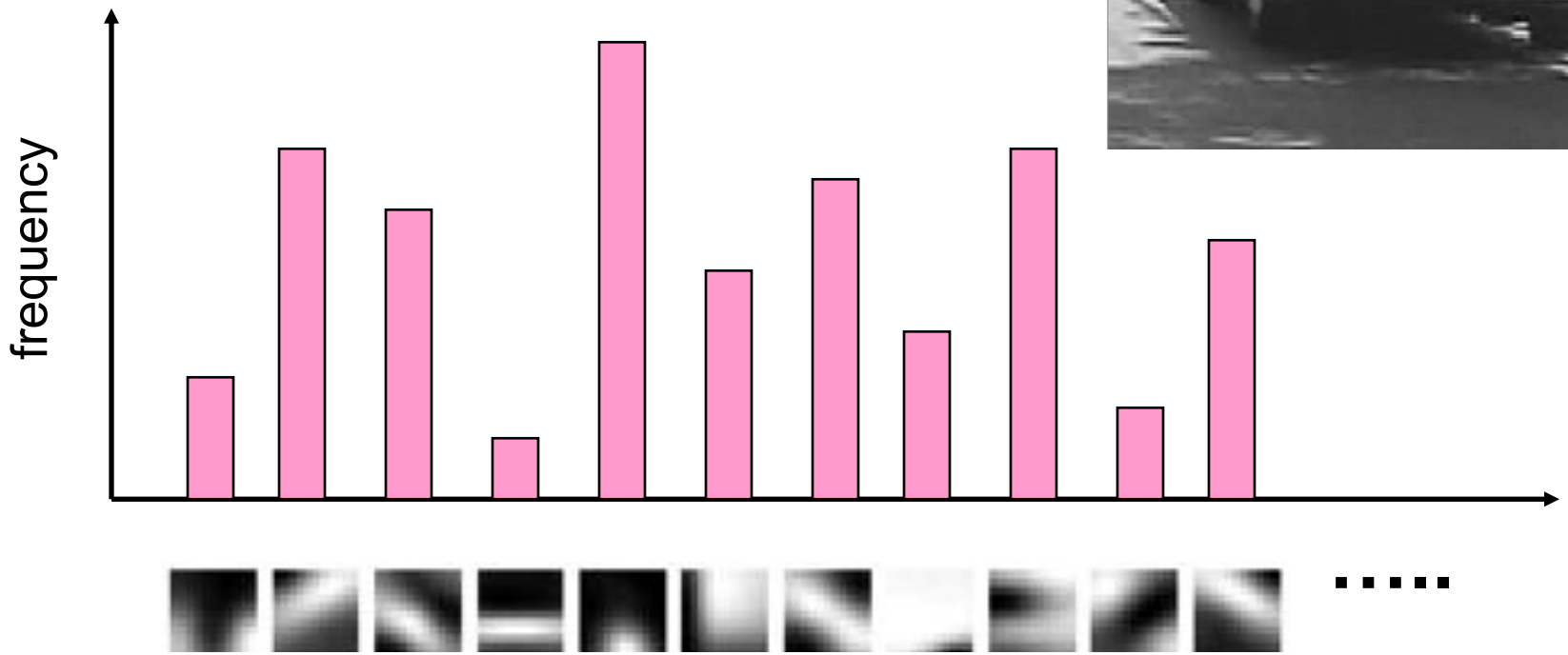


Image Representation

- Feature vector for standard classifier
  - e.g support vector machine.
- Can achieve success rates ~95%



Large-Scale Image Search

- Bag-of-words/features models have been useful in matching an image to a large database of object instances.
  - e.g. Caltech games dataset (11,400 images of games covers)

Large-Scale Image Search

- Bag-of-words/features models have been useful in matching an image to a large database of object instances.
  - e.g. Caltech games dataset (11,400 images of games covers)

Large-Scale Image Search

- Bag-of-words/features models have been useful in matching an image to a large database of object instances.
  - e.g. Caltech games dataset (11,400 images of games covers)

Large-Scale Image Search

- Bag-of-words/features models have been useful in matching an image to a large database of object instances.
  - e.g. Caltech games dataset (11,400 images of games covers)

Large-Scale Image Search

- Bag-of-words/features models have been useful in matching an image to a large database of object instances.
  - e.g. Caltech games dataset (11,400 images of games covers)

How do I find images like/of this 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

- In text documents, some words are more discriminative than others

  the, and, or vs. ***puppy, Brexit, Cher***

- the bigger the 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
- Instead of computing a regular histogram, weight each word (or feature) j by it's *inverse document frequency*

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

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}
```

- 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.
  - Use measures of histogram similarity to select best from those.

Large Scale Image Search

query image

Large Scale Image Search

query image

Large Scale Image Search

query image

top 6 results

Large Scale Image Search

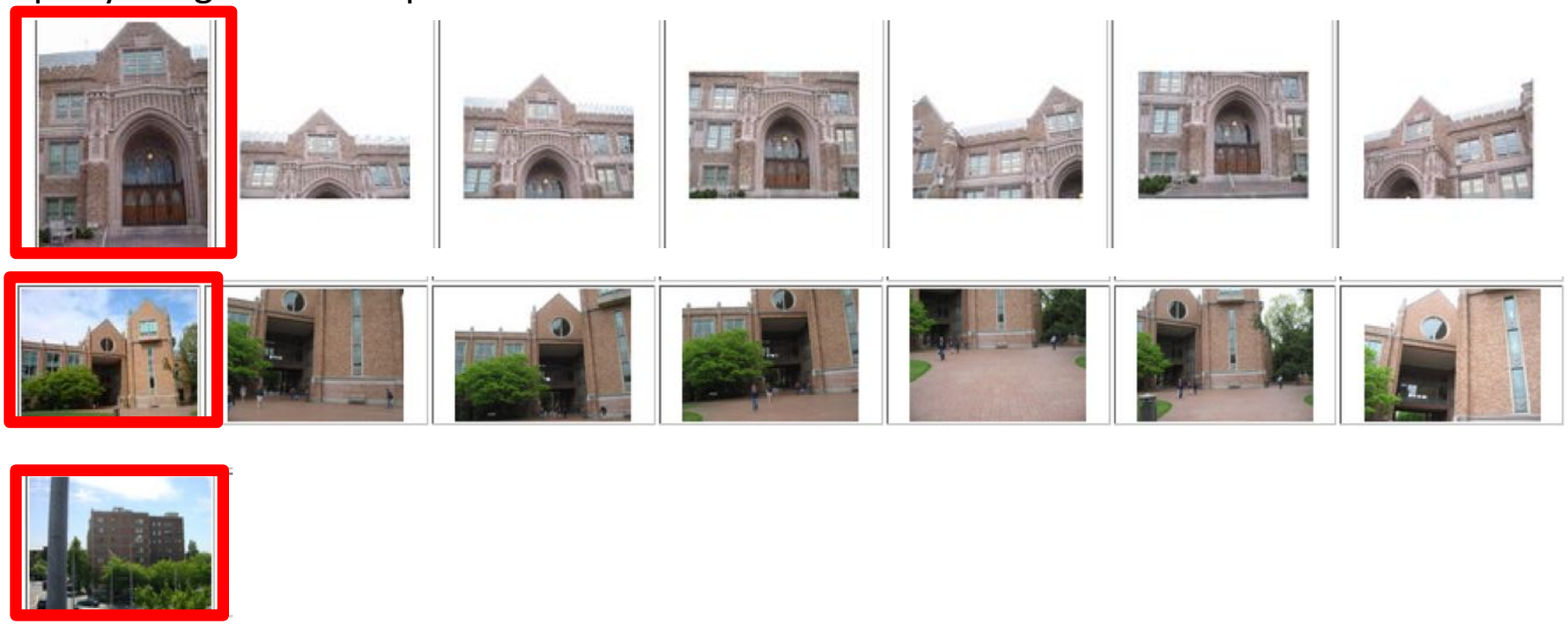
query image

top 6 results

Large Scale Image Search

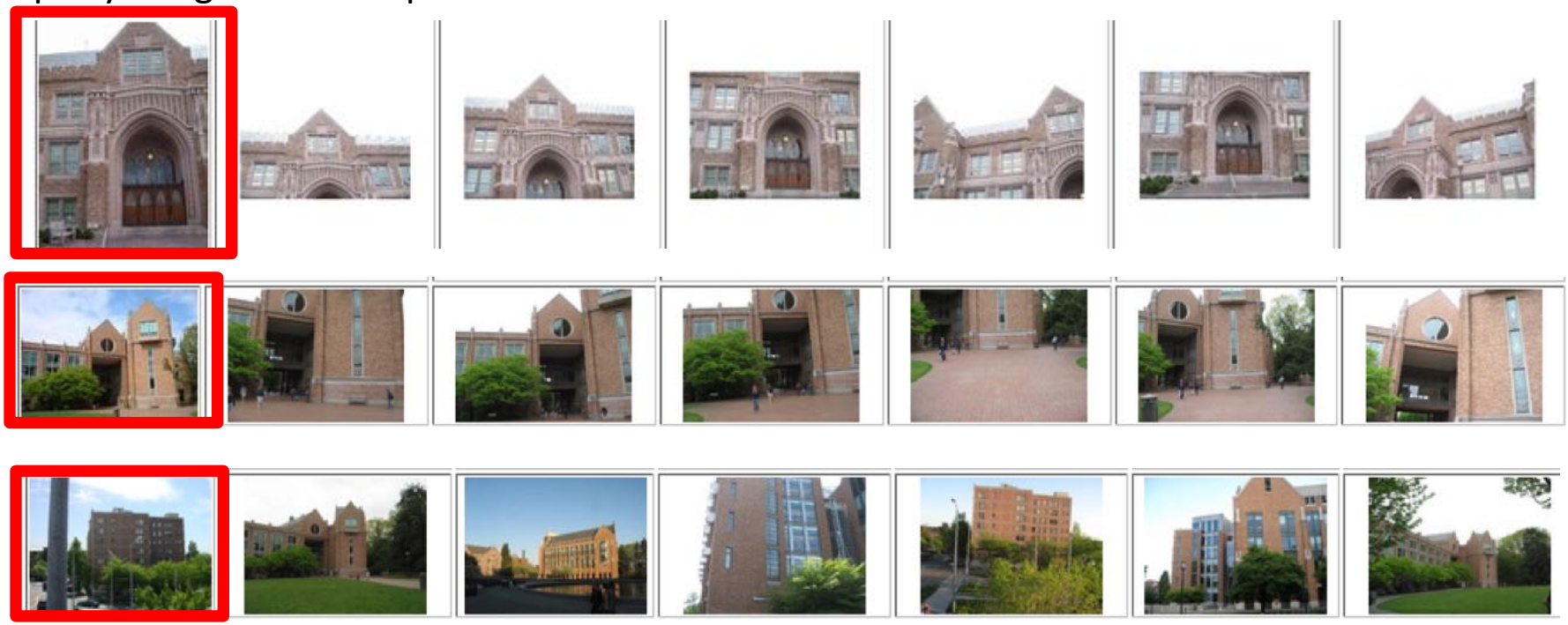
query image

top 6 results



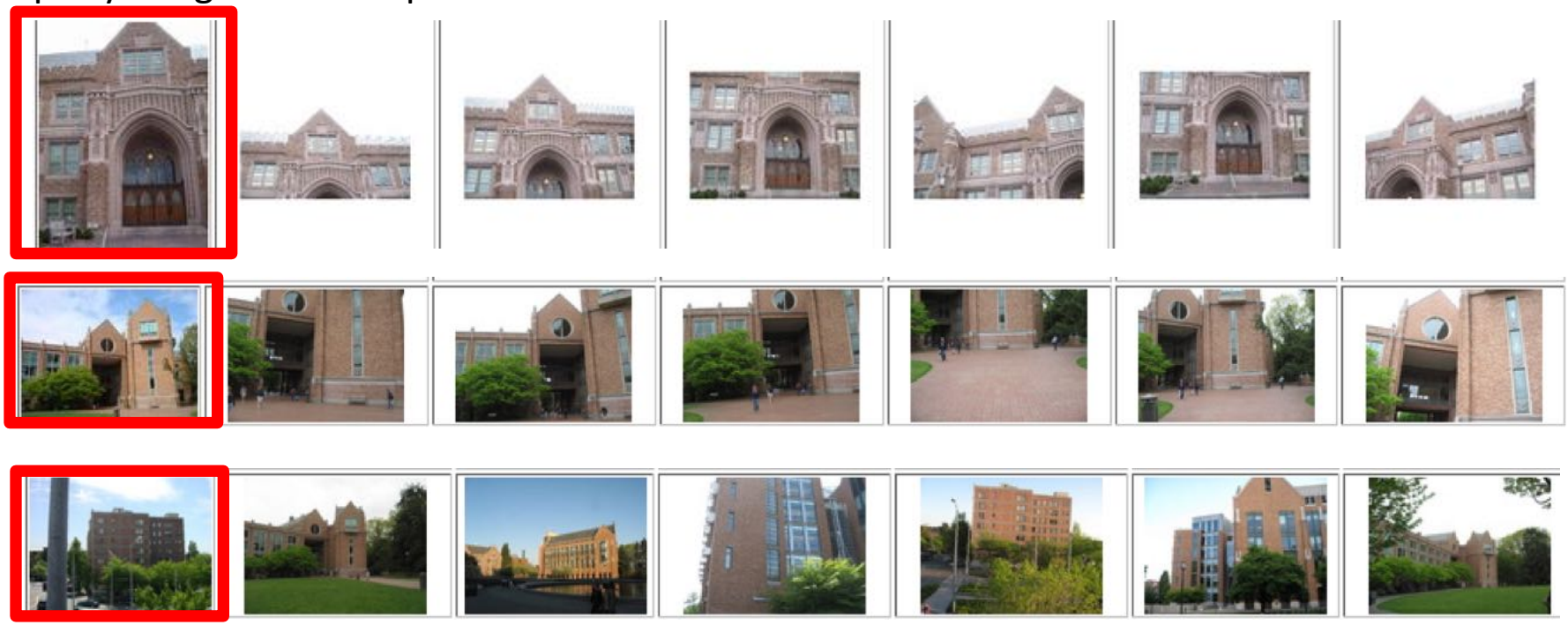
Large Scale Image Search

query image top 6 results



Large Scale Image Search

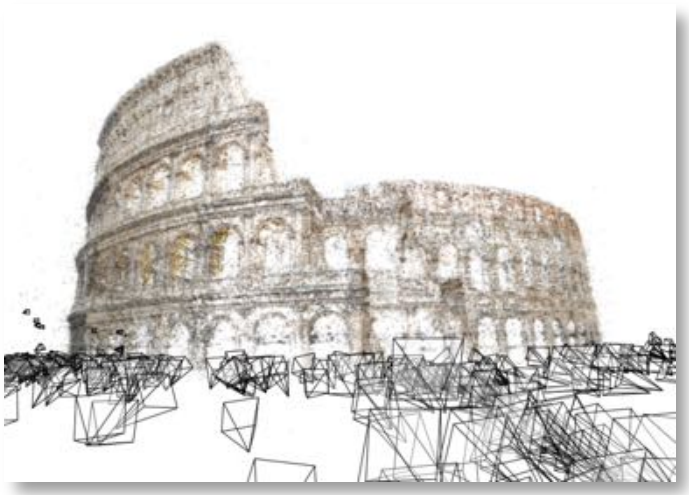
query image top 6 results



- Works well for CD covers, movie posters
- Real-time performance possible
- performance degrades as the database grows

Large Scale Image Matching

- Ability to efficiently match large numbers of images allows us to pull images of particular objects from public databases and use them to, e.g. build 3D models
- From 1,000,000 images of ‘Rome’……



Colosseum

Trevi Fountain

Conclusion

- Good features are crucial to object class detection.
- Bag of words is a generic approach that identifies useful (slightly more) abstract features by clustering features found in a training set.
- Histograms of these abstract features provide compact representations of images.
- Histograms can be
  - Weighted by inverse document frequency
  - The feature vectors supplied to a classifier
  - Matched to extract images of a given object from a large scale database