Clustering

Computer Vision

CS 543 / ECE 549

University of Illinois

Derek Hoiem

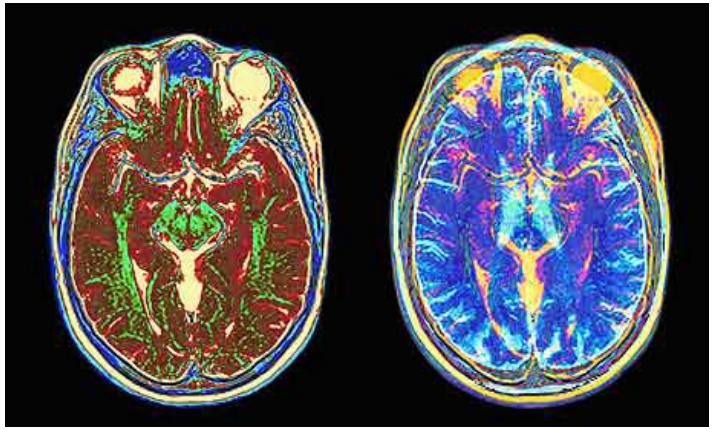
Today's class

- Fitting and alignment
  - One more algorithm: ICP
  - Review of all the algorithms

- Clustering algorithms
  - K-means
  - Hierarchical clustering
  - Spectral clustering

What if you want to align but have no prior matched pairs?

- Hough transform and RANSAC not applicable
- Important applications



Medical imaging: match brain scans or contours

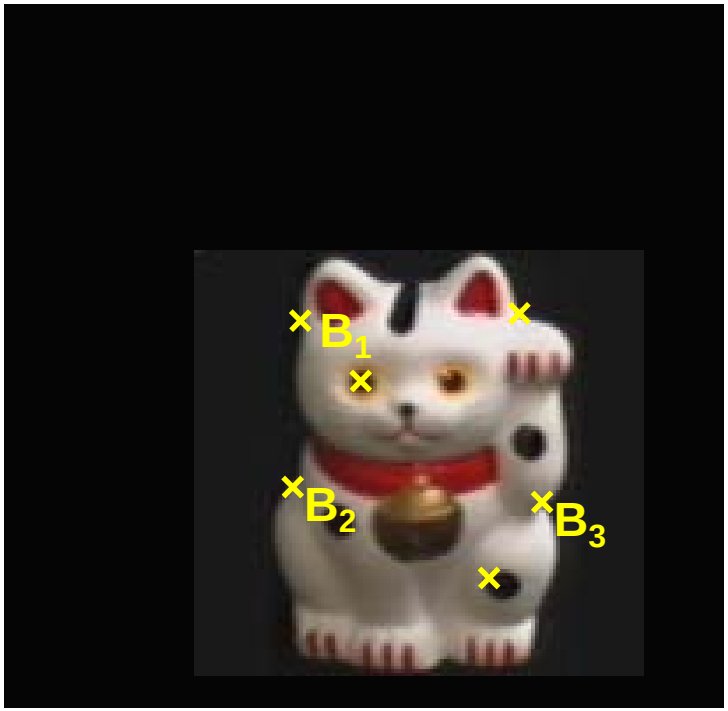
Robotics: match point clouds

Iterative Closest Points (ICP) Algorithm

Goal: estimate transform between two dense sets of points

1. **Assign** each point in {Set 1} to its nearest neighbor in {Set 2}
2. **Estimate** transformation parameters
   - e.g., least squares or robust least squares
3. **Transform** the points in {Set 1} using estimated parameters
4. **Repeat** steps 2-4 until change is very small

Example: solving for translation

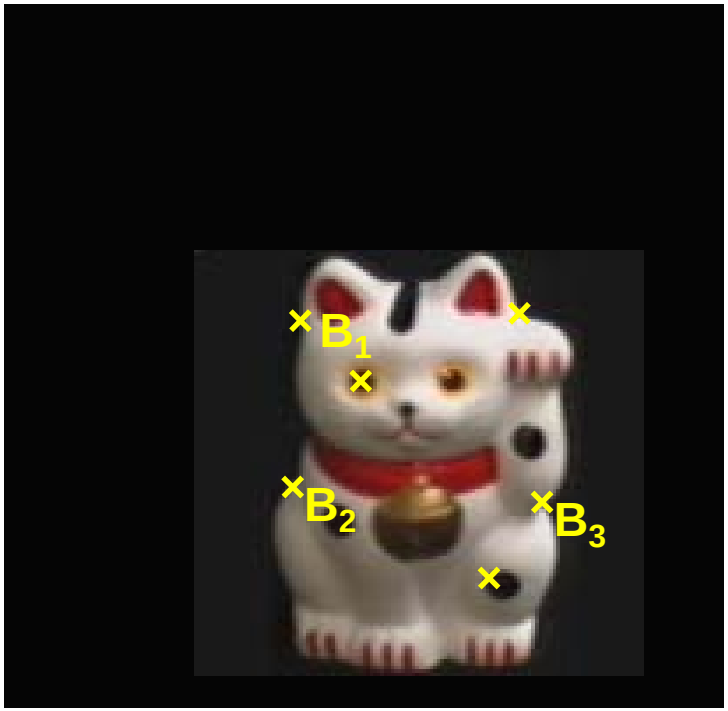


Given matched points in {A} and {B}, estimate the translation of the object

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Example: solving for translation

(t_x, t_y)



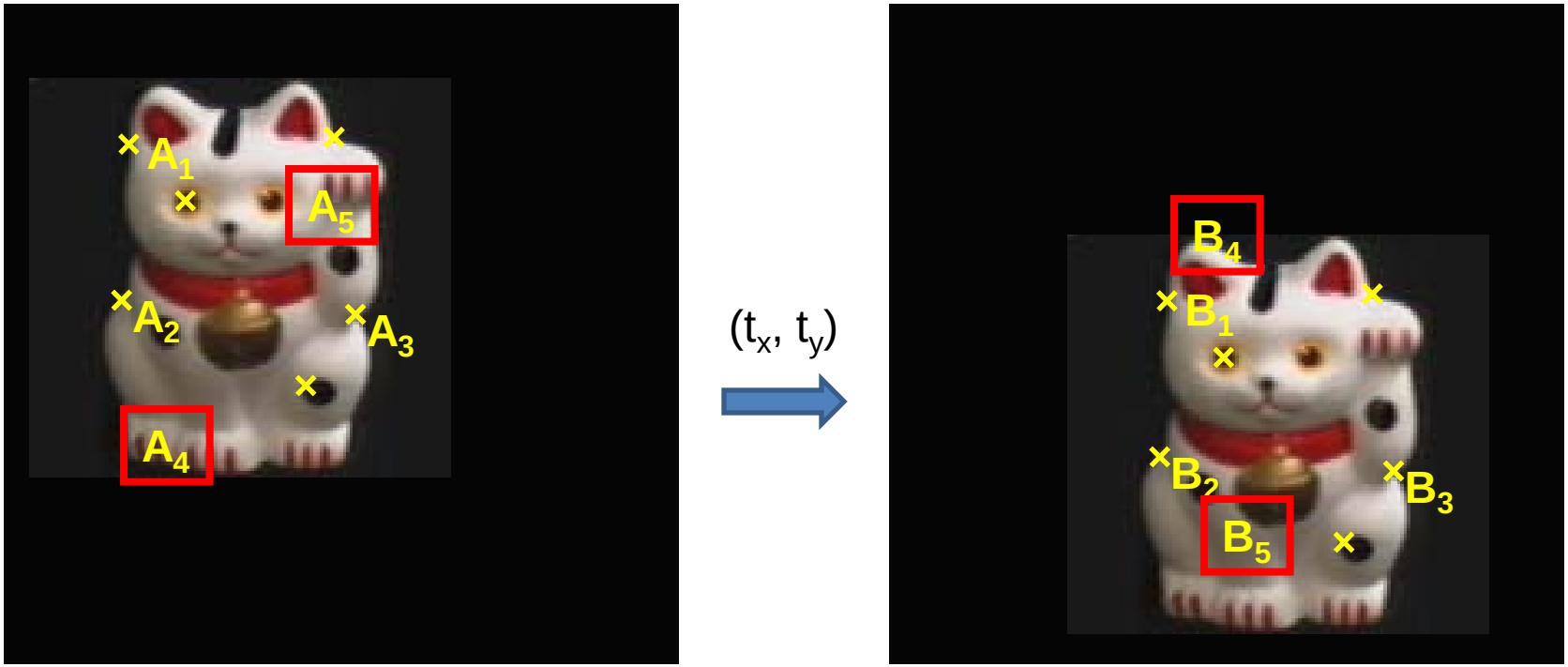
Least squares solution

1. Write down objective function
2. Derived solution
   a) Compute derivative
   b) Compute solution
3. Computational solution
   a) Write in form Ax=b
   b) Solve using pseudo-inverse or eigenvalue decomposition

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_1^B - x_1^A \\ y_1^B - y_1^A \\ \vdots \\ x_n^B - x_n^A \\ y_n^B - y_n^A \end{bmatrix}$$

Example: solving for translation



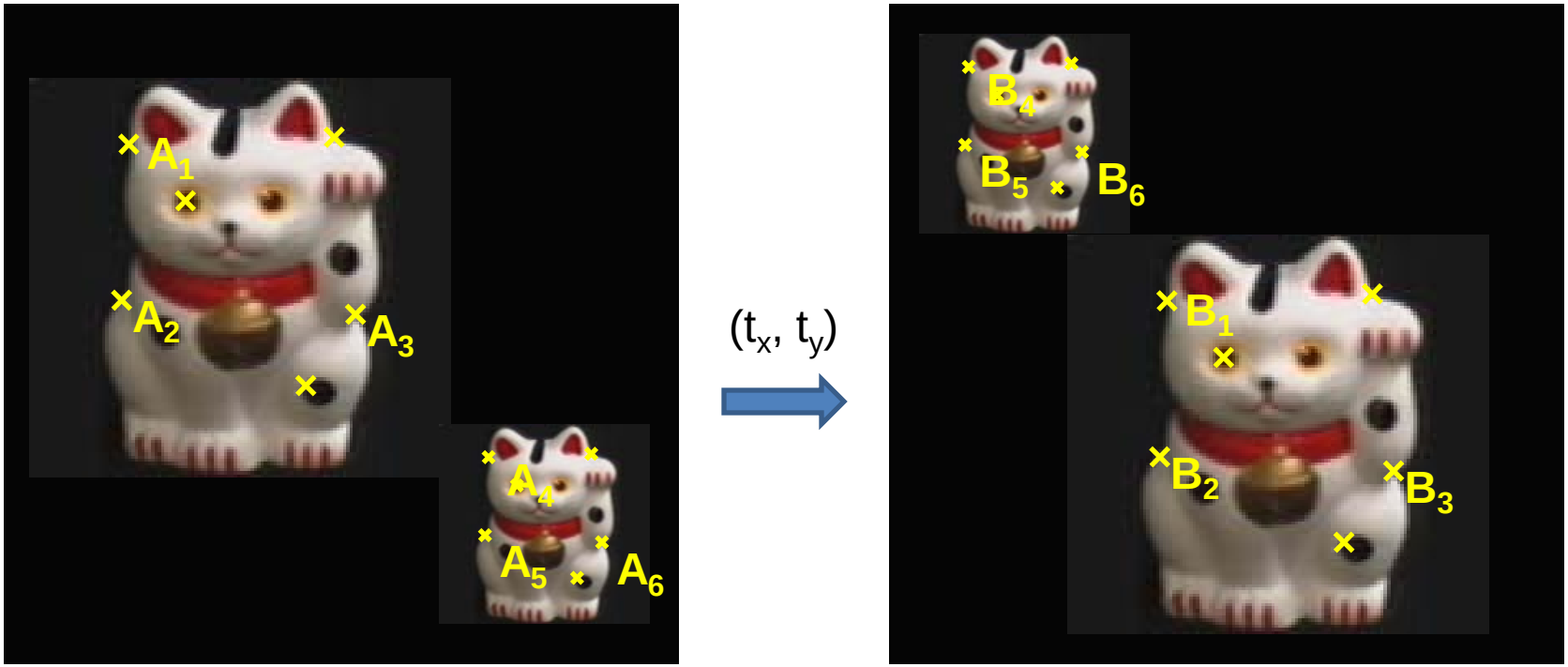
Problem: outliers

RANSAC solution

1. Sample a set of matching points (1 pair)
2. Solve for transformation parameters
3. Score parameters with number of inliers
4. Repeat steps 1-3 N times

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Example: solving for translation



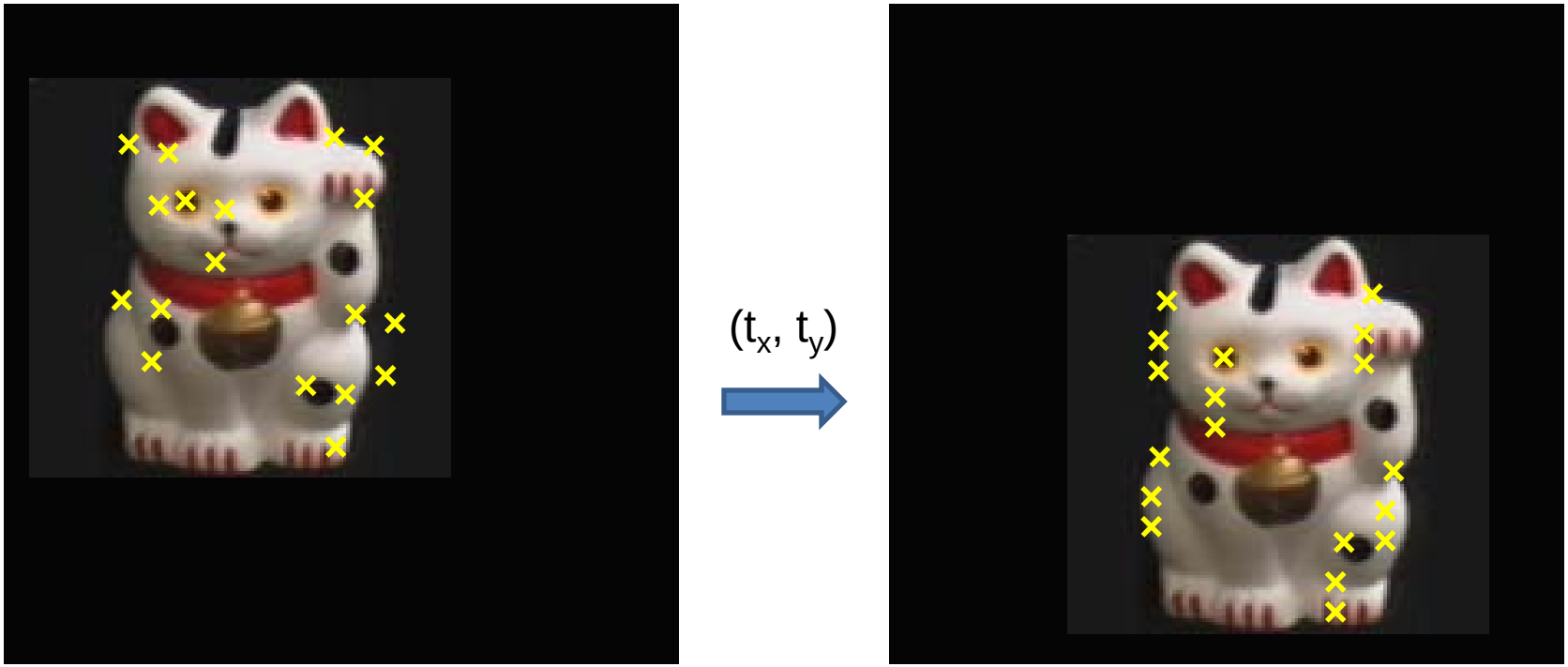
Problem: outliers, multiple objects, and/or many-to-one matches

Hough transform solution

1. Initialize a grid of parameter values
2. Each matched pair casts a vote for consistent values
3. Find the parameters with the most votes
4. Solve using least squares with inliers

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Example: solving for translation



Problem: no initial guesses for correspondence

ICP solution

1. Find nearest neighbors for each point
2. Compute transform using matches
3. Move points using transform
4. Repeat steps 1-3 until convergence

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Clustering: group together similar points and represent them with a single token

Key Challenges:

1) What makes two points/images/patches similar?

2) How do we compute an overall grouping from pairwise similarities?

Why do we cluster?

- **Summarizing data**
  - Look at large amounts of data
  - Patch-based compression or denoising
  - Represent a large continuous vector with the cluster number
- **Counting**
  - Histograms of texture, color, SIFT vectors
- **Segmentation**
  - Separate the image into different regions
- **Prediction**
  - Images in the same cluster may have the same labels

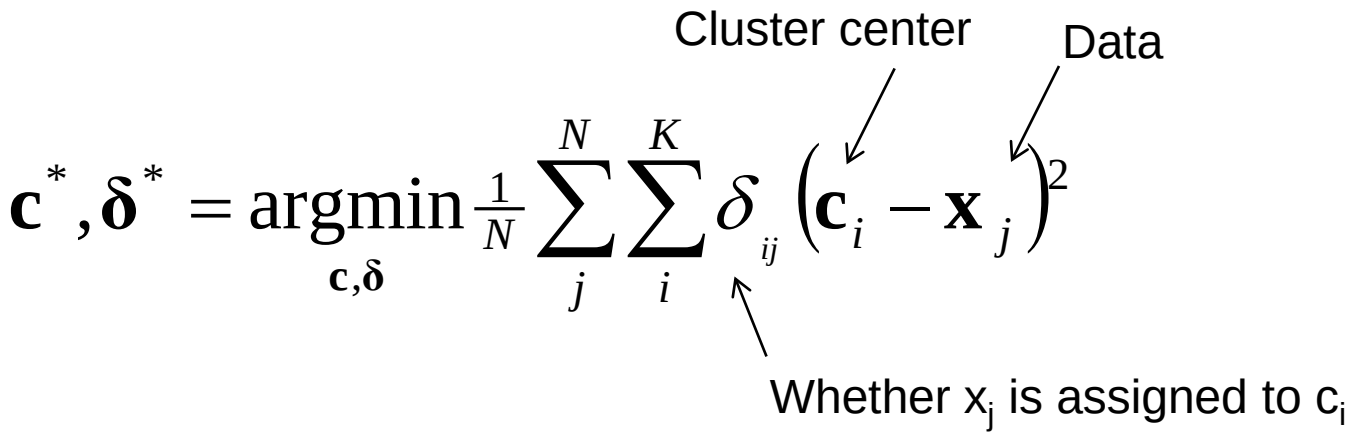
How do we cluster?

- K-means
  - Iteratively re-assign points to the nearest cluster center
- Agglomerative clustering
  - Start with each point as its own cluster and iteratively merge the closest clusters
- Spectral clustering
  - Split the nodes in a graph based on assigned links with similarity weights

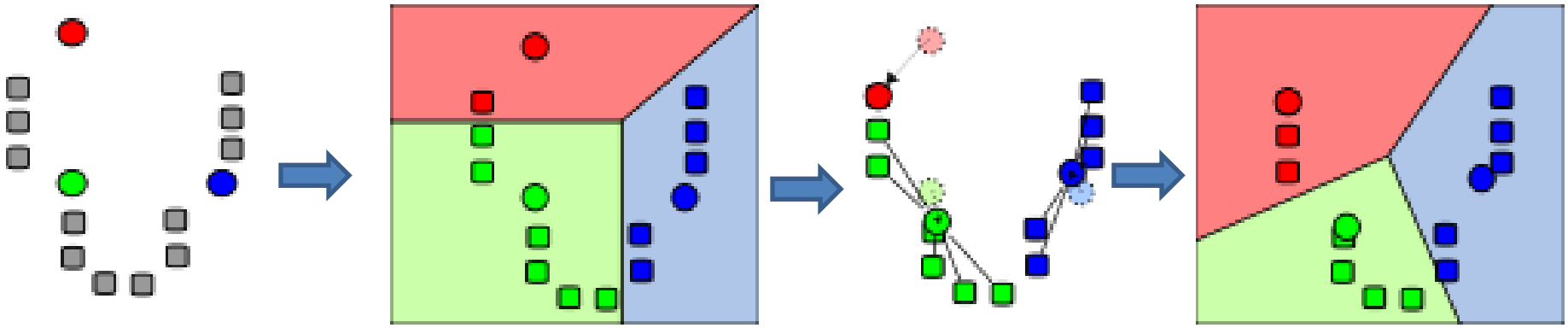
Clustering for Summarization

Goal: cluster to minimize variance in data given clusters

– Preserve information



K-means



0. Initialize Cluster Centers

1. Assign Points to Clusters

2. Re-compute Means

Repeat (1) and (2)

Illustration Source: wikipedia

K-means

1. Initialize cluster centers: $\mathbf{c}^0$; t=0

2. Assign each point to the closest center

$$\boldsymbol{\delta}^t = \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij} \left(\mathbf{c}_i^{t-1} - \mathbf{x}_j \right)^2$$

3. Update cluster centers as the mean of the points

$$\mathbf{c}^t = \underset{\mathbf{c}}{\operatorname{argmin}} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij}^t \left(\mathbf{c}_i - \mathbf{x}_j \right)^2$$

4. Repeat 2-3 until no points are re-assigned (t=t+1)

K-means: design choices

- Initialization
  - Randomly select K points as initial cluster center
  - Or greedily choose K points to minimize residual
- Distance measures
  - Traditionally Euclidean, could be others
- Optimization
  - Will converge to a *local minimum*
  - May want to perform multiple restarts

Illustration Source: wikipedia

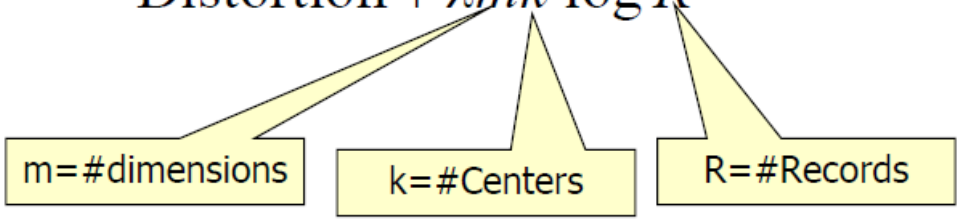
How to choose the number of clusters?

- Minimum Description Length (MDL) principal for model comparison

- Minimize Schwarz Criterion
  - also called Bayes Information Criteria (BIC)

$$\text{Distortion} + \lambda\,(\#\text{parameters})\log R$$

$$= \text{Distortion} + \lambda mk \log R$$



How to evaluate clusters?

- Generative
  - How well are points reconstructed from the clusters?
- Discriminative
  - How well do the clusters correspond to labels?
    - Purity
  - Note: unsupervised clustering does not aim to be discriminative

How to choose the number of clusters?

- Validation set
  - Try different numbers of clusters and look at performance
    - When building dictionaries (discussed later), more clusters typically work better

K-means demos

General
http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

Color clustering
http://www.cs.washington.edu/research/imagedatabase/demo/kmcluster/

Conclusions: K-means

Good

- Finds cluster centers that minimize conditional variance (good representation of data)
- Simple to implement, widespread application

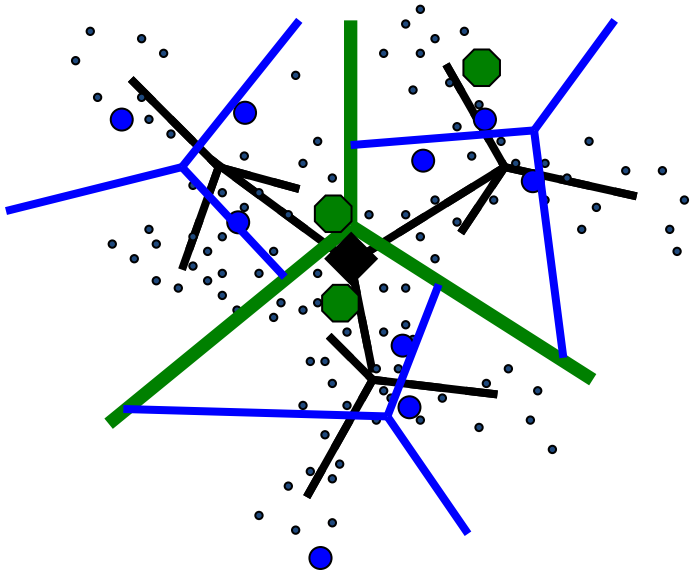
Bad

- Prone to local minima
- Need to choose K
- All clusters have the same parameters (e.g., distance measure is non-adaptive)
- Can be slow: each iteration is O(KNd) for N d-dimensional points

Building Visual Dictionaries

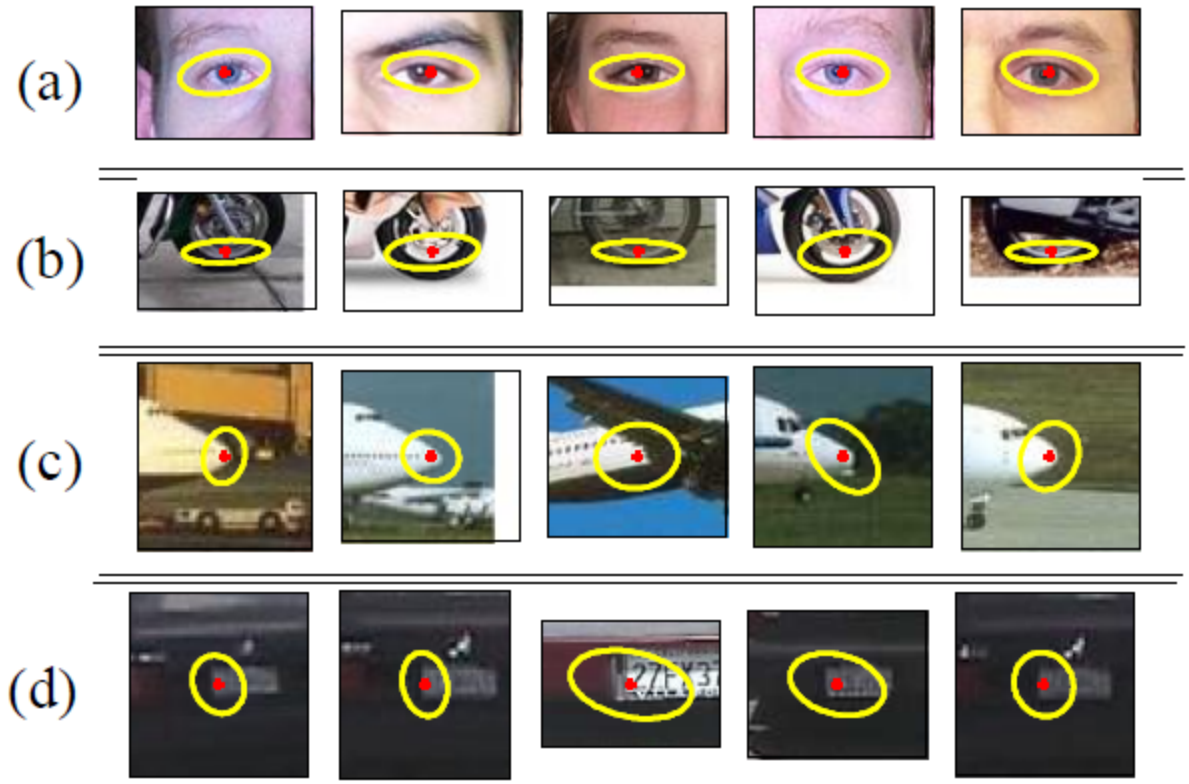
1. Sample patches from a database
   - E.g., 128 dimensional SIFT vectors

2. Cluster the patches
   - Cluster centers are the dictionary



3. Assign a codeword (number) to each new patch, according to the nearest cluster

Examples of learned codewords



Most likely codewords for 4 learned “topics”
EM with multinomial (problem 3) to get topics

http://www.robots.ox.ac.uk/~vgg/publications/papers/sivic05b.pdf Sivic et al. ICCV 2005

Common similarity/distance measures

- P-norms
  - City Block (L1)
  - Euclidean (L2)
  - L-infinity

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

$$\|\boldsymbol{x}\|_1 := \sum_{i=1}^{n} |x_i|$$

$$\|\boldsymbol{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}$$

$$\|\mathbf{x}\|_\infty := \max\left(|x_1|, \ldots, |x_n|\right)$$

Here x_i is the distance between two points

- Mahalanobis
  - Scaled Euclidean

$$d(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{N} \frac{(x_i - y_i)^2}{\sigma_i^2}}$$

- Cosine distance

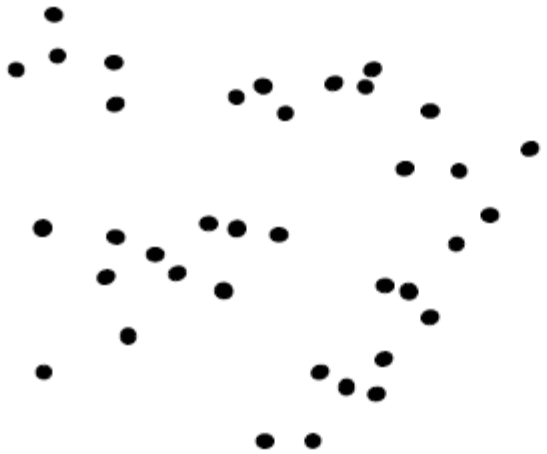
$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

K-medoids

- Just like K-means except
  - Represent the cluster with one of its members, rather than the mean of its members
  - Choose the member (data point) that minimizes cluster dissimilarity

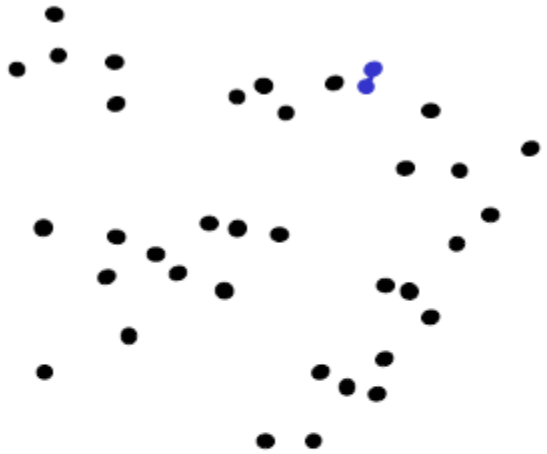
- Applicable when a mean is not meaningful
  - E.g., clustering values of hue or using L-infinity similarity

Agglomerative clustering



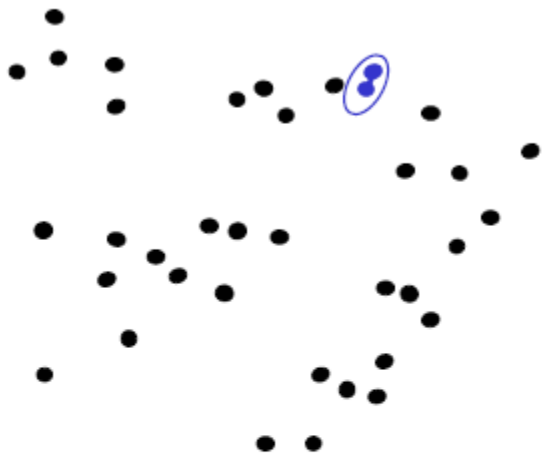
1. Say "Every point is its own cluster"

Agglomerative clustering



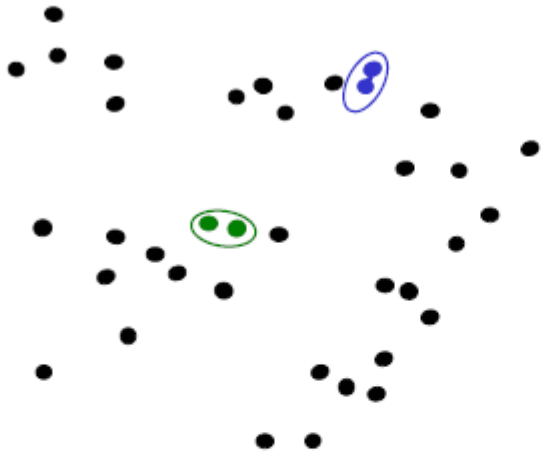
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters

Agglomerative clustering



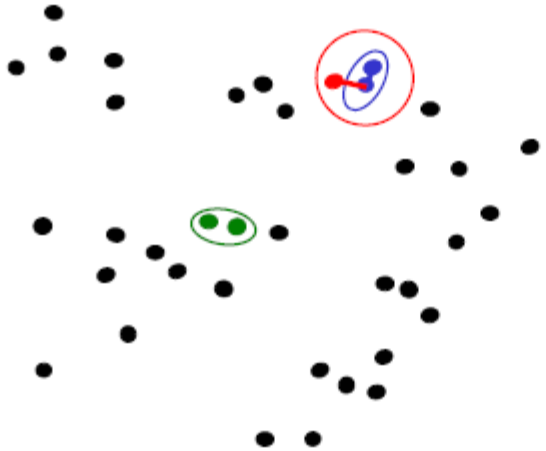
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster

Agglomerative clustering



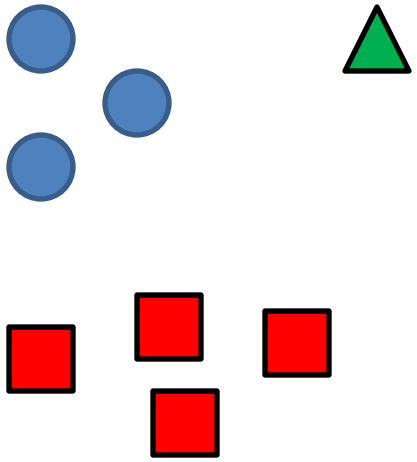
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

Agglomerative clustering



1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

Agglomerative clustering

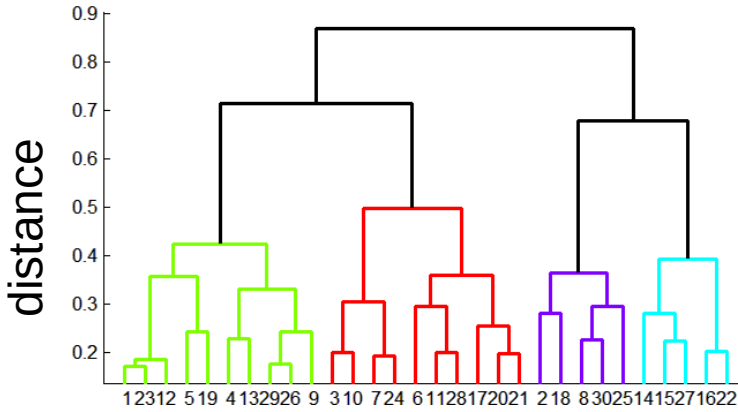


How to define cluster similarity?

- Average distance between points, maximum distance, minimum distance
- Distance between means or medoids

How many clusters?

- Clustering creates a dendrogram (a tree)
- Threshold based on max number of clusters or based on distance between merges



Agglomerative clustering demo

http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletH.html

Conclusions: Agglomerative Clustering

Good

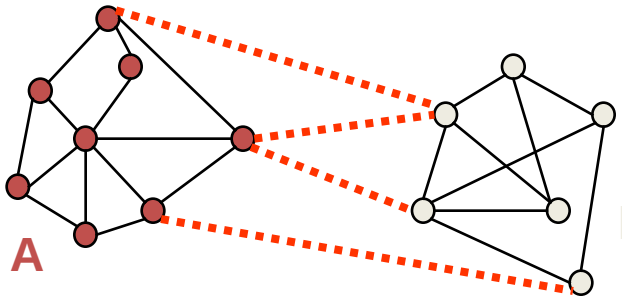
- Simple to implement, widespread application
- Clusters have adaptive shapes
- Provides a hierarchy of clusters

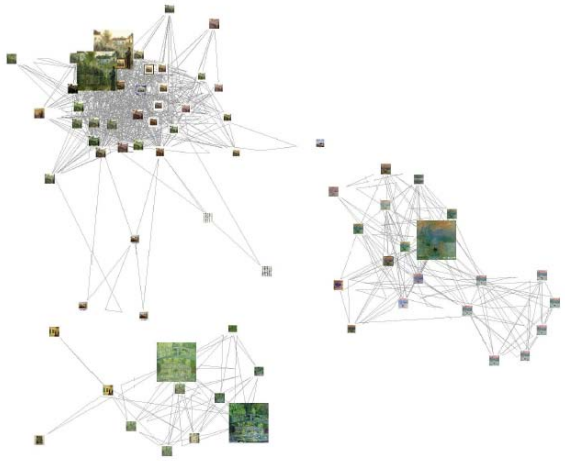
Bad

- May have imbalanced clusters
- Still have to choose number of clusters or threshold
- Need to use an “ultrametric” to get a meaningful hierarchy

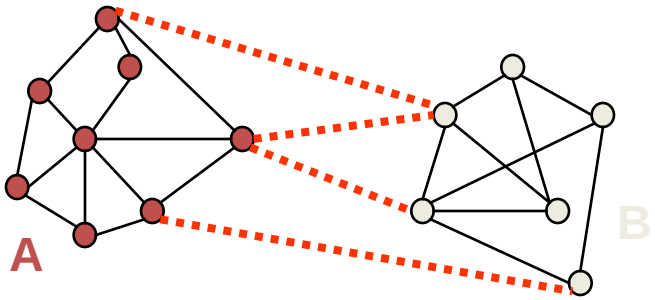
Spectral clustering

Group points based on links in a graph





Cuts in a graph



Normalized Cut

- a cut penalizes large segments
- fix by normalizing for size of segments

$$Ncut(A,B) = \frac{cut(A,B)}{volume(A)} + \frac{cut(A,B)}{volume(B)}$$

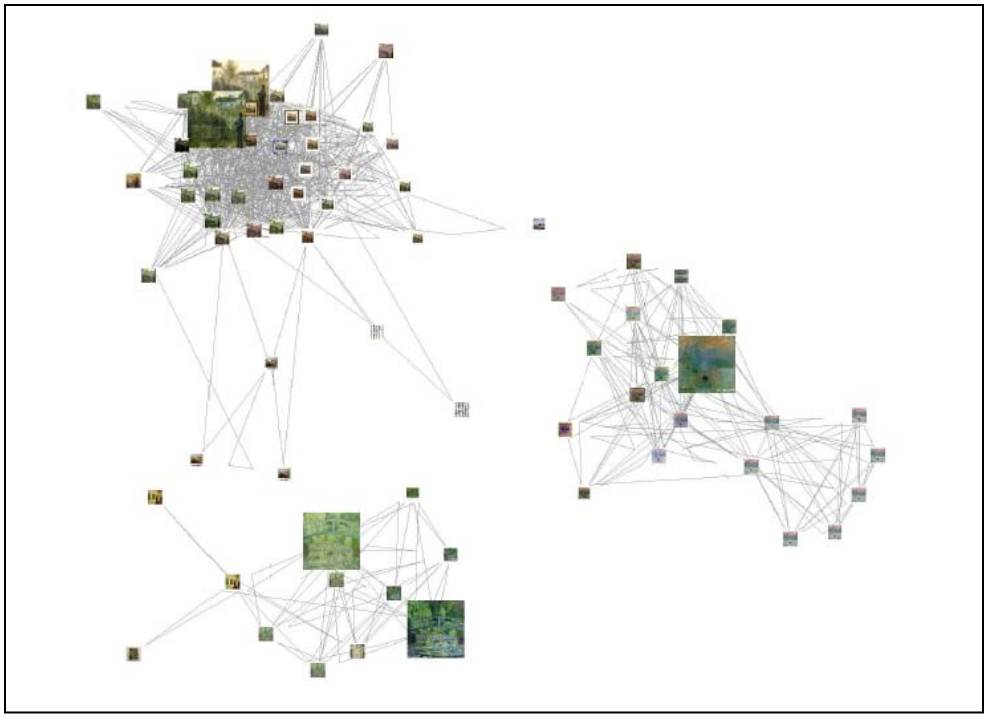
- volume(A) = sum of costs of all edges that touch A

Source: Seitz

Normalized cuts for segmentation

Visual PageRank

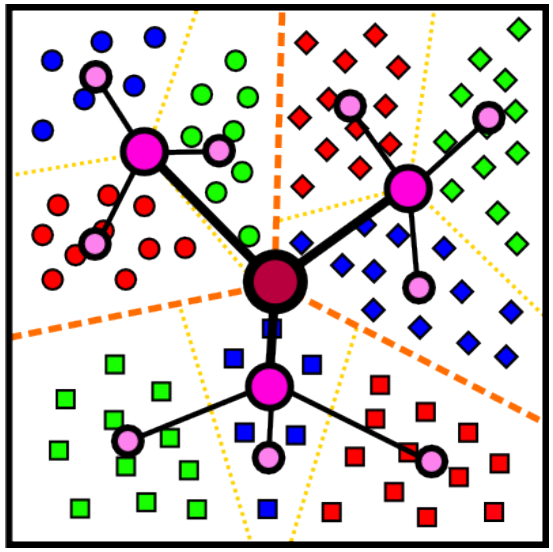
- Determining importance by random walk
  - What's the probability that you will randomly walk to a given node?
    - Create adjacency matrix based on visual similarity
    - Edge weights determine probability of transition



Jing Baluja 2008

Which algorithm to use?

- Quantization/Summarization: K-means
  - Aims to preserve variance of original data
  - Can easily assign new point to a cluster

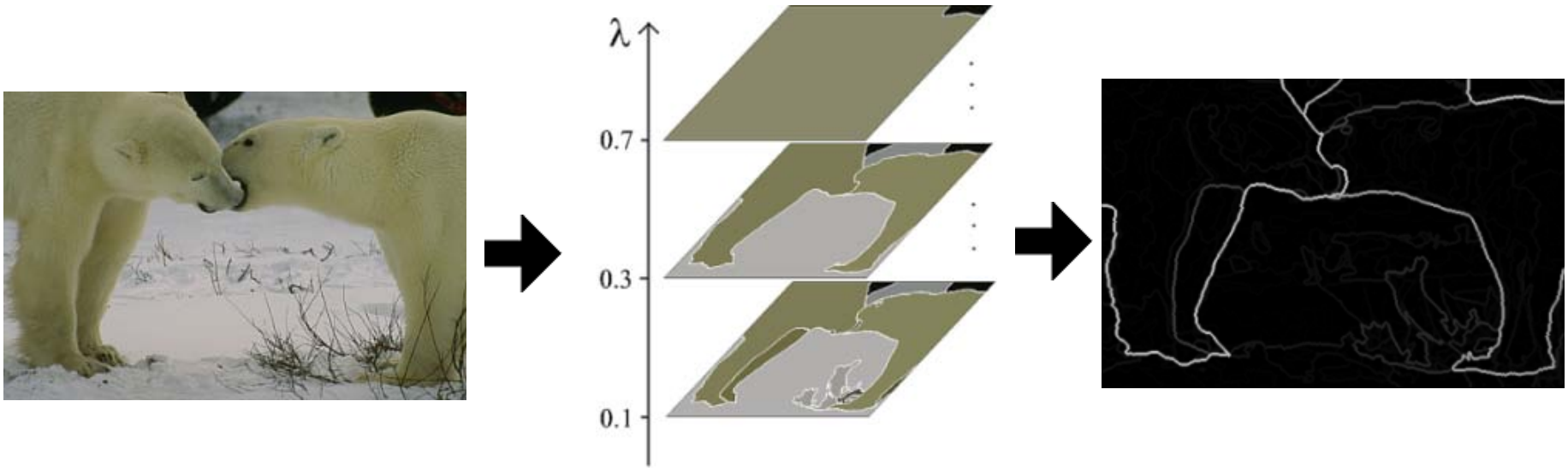


Quantization for computing histograms

Summary of 20,000 photos of Rome using "greedy k-means"
http://grail.cs.washington.edu/projects/canonview/

Which algorithm to use?

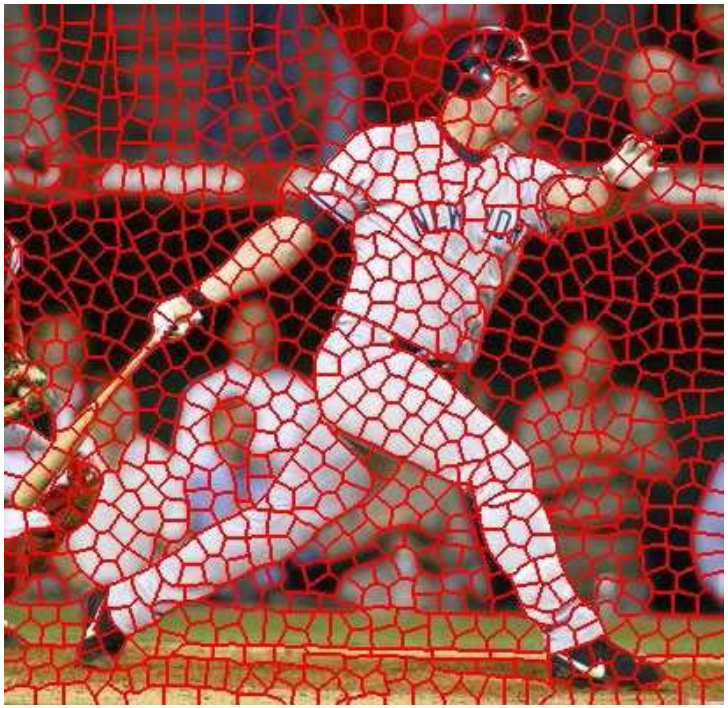
- Image segmentation: agglomerative clustering
  - More flexible with distance measures (e.g., can be based on boundary prediction)
  - Adapts better to specific data
  - Hierarchy can be useful



http://www.cs.berkeley.edu/~arbelaez/UCM.html

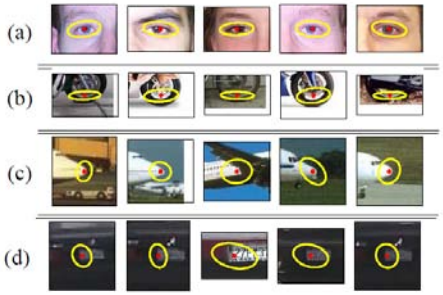
Which algorithm to use?

- Image segmentation: spectral clustering
  - Can provide more regular regions
  - Spectral methods also used to propagate global cues (e.g., Global pB)



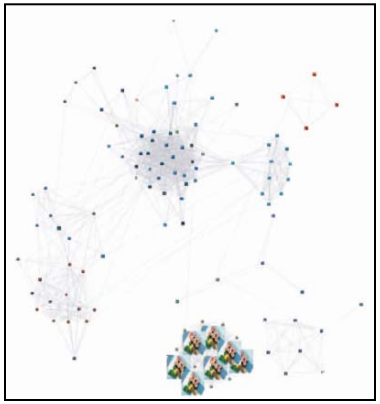
Things to remember

- K-means useful for summarization, building dictionaries of patches, general clustering



- Agglomerative clustering useful for segmentation, general clustering

- Spectral clustering useful for determining relevance, summarization, segmentation



Next class

- EM algorithm
  - Soft clustering
  - Mixture models
  - Hidden labels