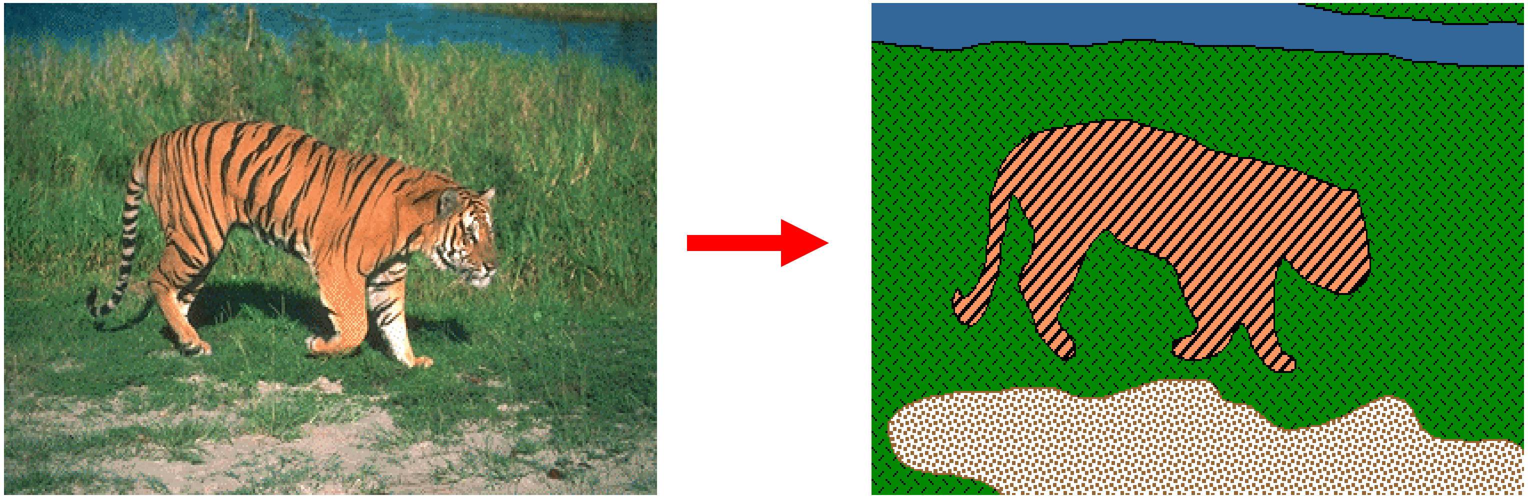
Segmentation (continued)



Reading:

Shapiro & Stockton, p. 279-289

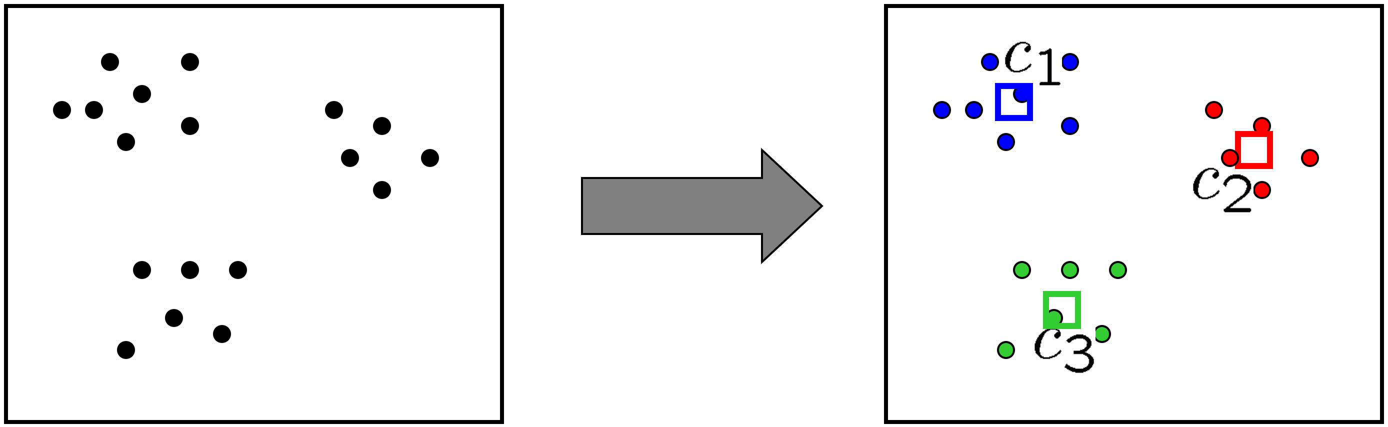
Reminder:

Office hours after class today in CSE 212

Clustering

Segmenting images by pixel color

- This is a clustering problem!



Objective

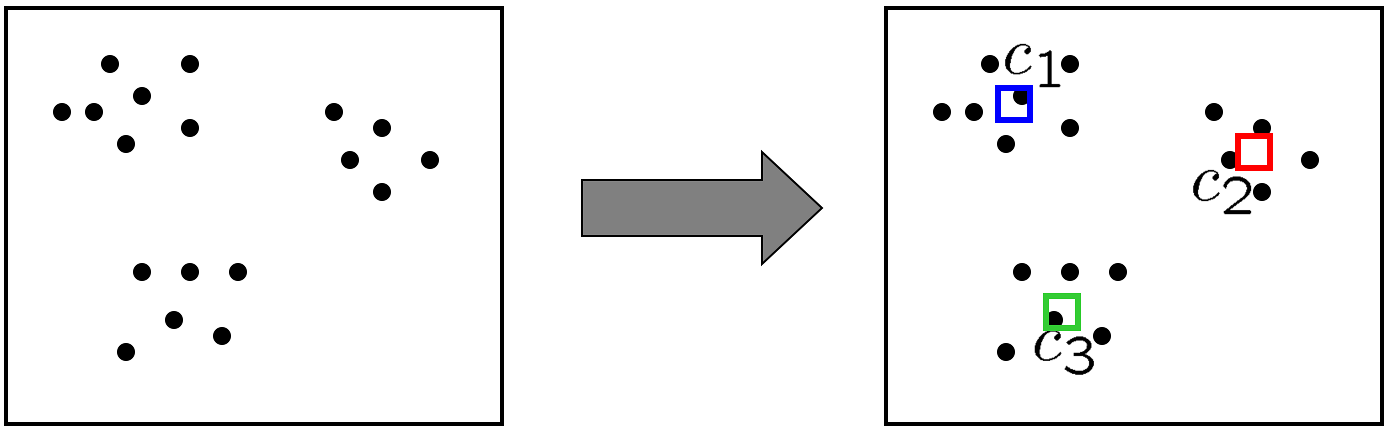
- Each point should be as close as possible to a cluster center
  - Minimize sum squared distance of each point to closest center

$$\sum_{\text{clusters } i} \quad \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

Break it down into subproblems

Suppose I tell you the cluster centers c_i

- Q: how to determine which points to associate with each c_i?
- A: for each point p, choose closest c_i



Suppose I tell you the points in each cluster

- Q: how to determine the cluster centers?
- A: choose c_i to be the mean of all points in the cluster

K-means clustering

K-means clustering algorithm

1. Randomly initialize the cluster centers, $c_1, ..., c_K$
2. Given cluster centers, determine points in each cluster
   - For each point p, find the closest c_i. Put p into cluster i
3. Given points in each cluster, solve for c_i
   - Set c_i to be the mean of points in cluster i
4. If c_i have changed, repeat Step 2

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

Properties

- Will always converge to *some* solution
- Can be a "local minimum"
  - does not always find the global minimum of objective function:

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

Probabilistic clustering

Basic questions

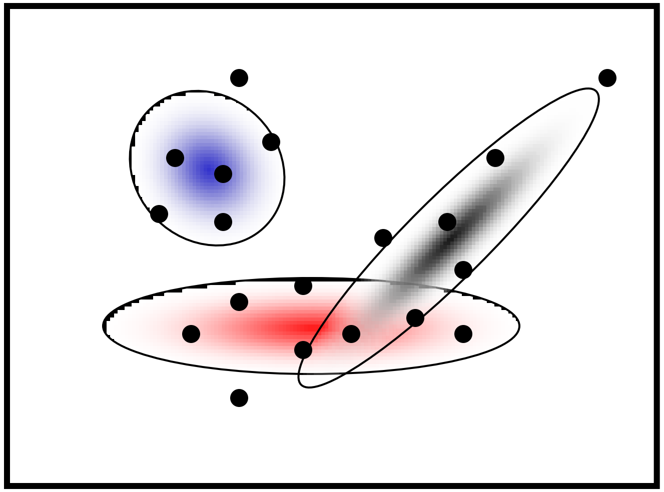
- what’s the probability that a point $\mathbf{x}$ is in cluster m?
- what’s the shape of each cluster?

K-means doesn’t answer these questions

Basic idea

- instead of treating the data as a bunch of points, assume that they are all generated by sampling from a probability distribution
- This is called a **generative model**
  - defined by a vector of parameters $\boldsymbol{\theta}$

Mixture of Gaussians



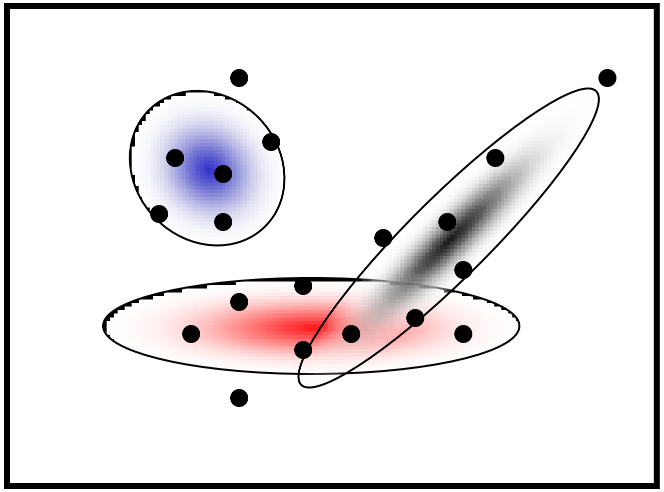
One generative model is a mixture of Gaussians (MOG)

- K Gaussian blobs with means $\boldsymbol{\mu}_b$ covariance matrices $\mathbf{V}_b$, dimension d
  - blob *b* defined by: $$P(x|\mu_b, V_b) = \frac{1}{\sqrt{(2\pi)^d |V_b|}} e^{-\frac{1}{2}(x-\mu_b)^T V_b^{-1}(x-\mu_b)}$$
- blob *b* is selected with probability α_b
- the likelihood of observing **x** is a weighted mixture of Gaussians

$$P(x|\theta) = \sum_{b=1}^{K} \alpha_b P(x|\theta_b)$$

- where $\theta = [\mu_1, \ldots, \mu_n, V_1, \ldots, V_n]$

Expectation maximization (EM)



Goal

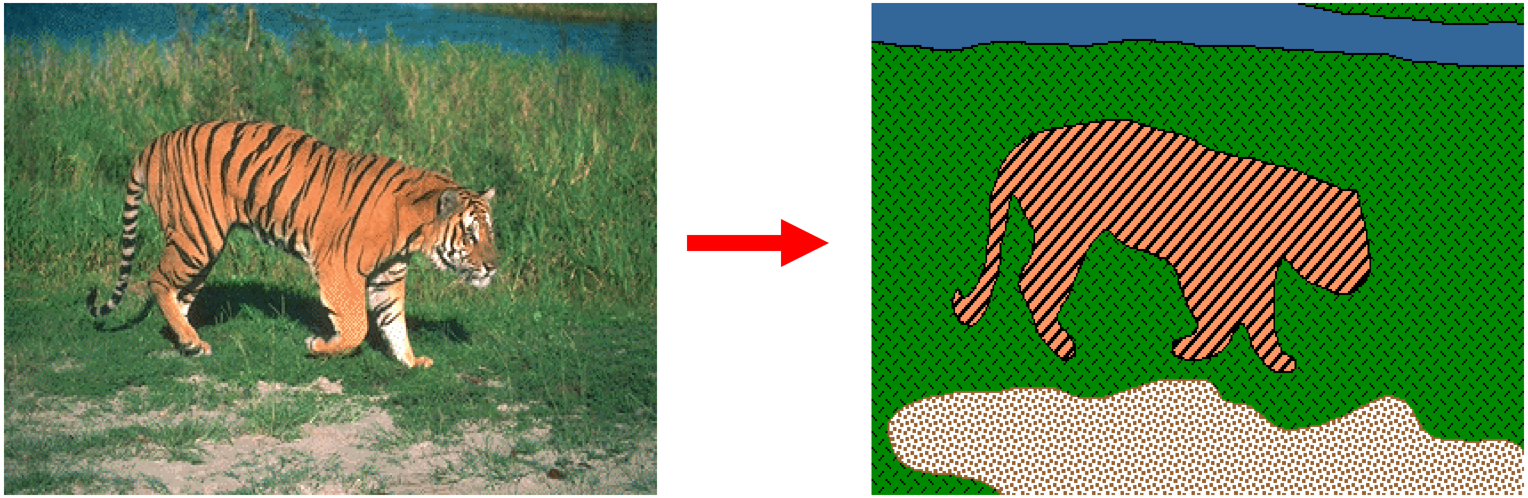
- find blob parameters θ that maximize the likelihood function:

$$P(data|\theta) = \prod_x P(x|\theta)$$

Approach:

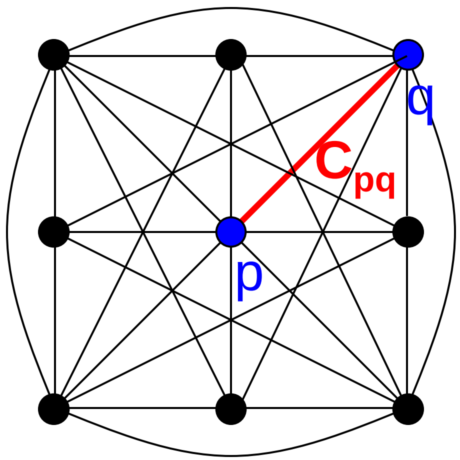
1. E step: given current guess of blobs, compute ownership of each point
2. M step: given ownership probabilities, update blobs to maximize likelihood function
3. repeat until convergence

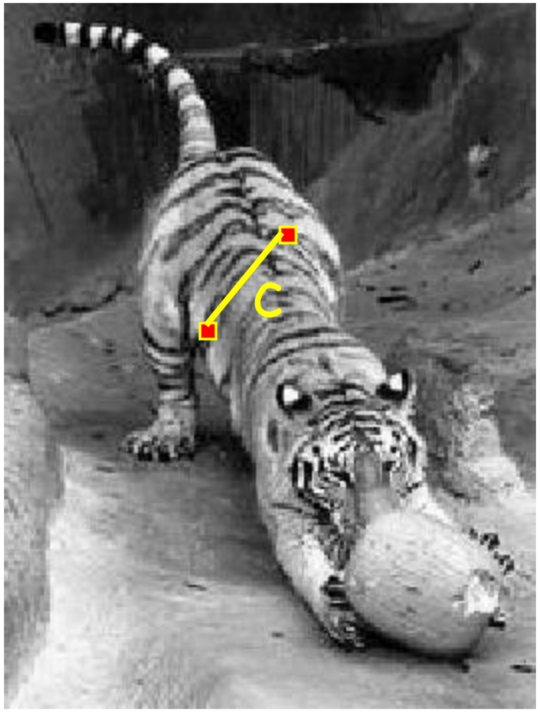
Graph-based segmentation



What if we look at relationships between pixels?

Images as graphs

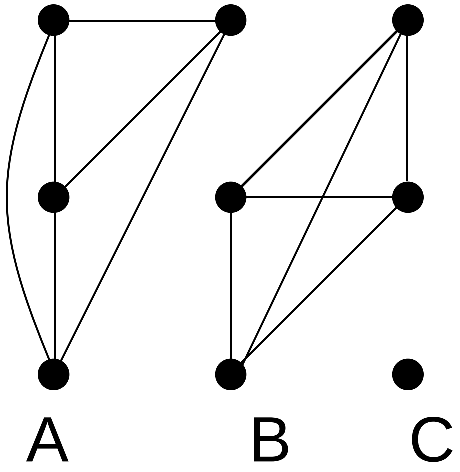




Fully-connected graph

- node for every pixel
- link between *every* pair of pixels, **p**,**q**
- cost $\mathbf{c_{pq}}$ for each link
  - $\mathbf{c_{pq}}$ measures *similarity*
    - » similarity is *inversely proportional* to difference in color and position

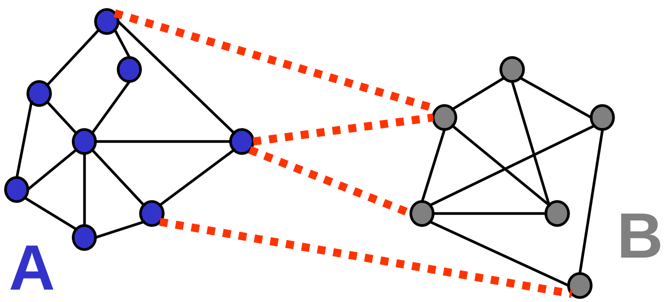
Segmentation by Graph Cuts



Break Graph into Segments

- Delete links that cross between segments
- Easiest to break links that have low cost (low similarity)
  - similar pixels should be in the same segments
  - dissimilar pixels should be in different segments

Cuts in a graph



Link Cut

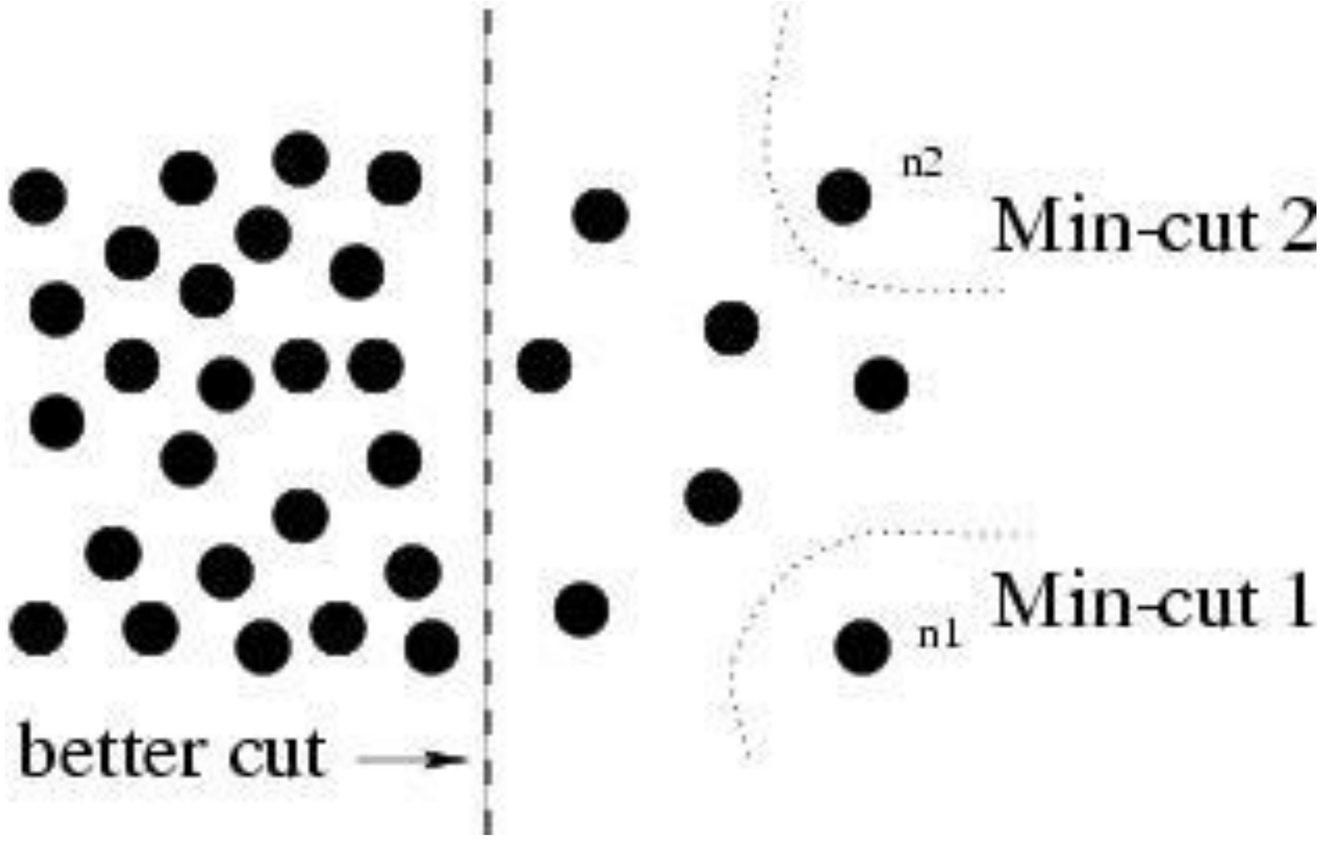
- set of links whose removal makes a graph disconnected
- cost of a cut:

$$cut(A,B) = \sum_{p \in A, q \in B} c_{p,q}$$

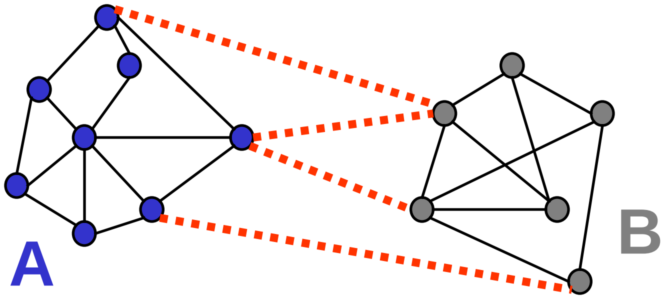
Find minimum cut

- gives you a segmentation
- fast algorithms exist for doing this

But min cut is not always the best cut...



Cuts in a graph



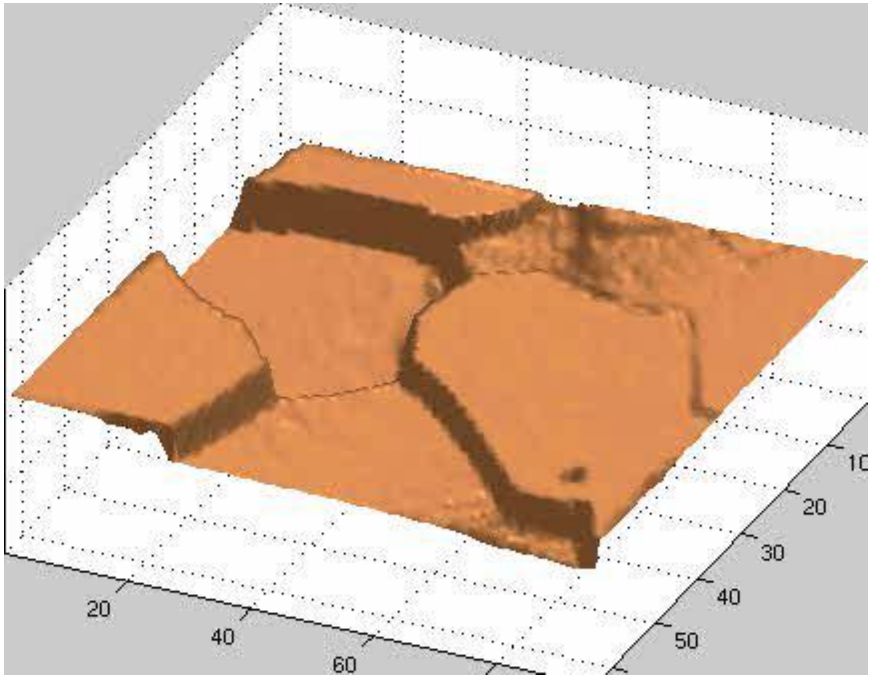
Normalized Cut

- 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

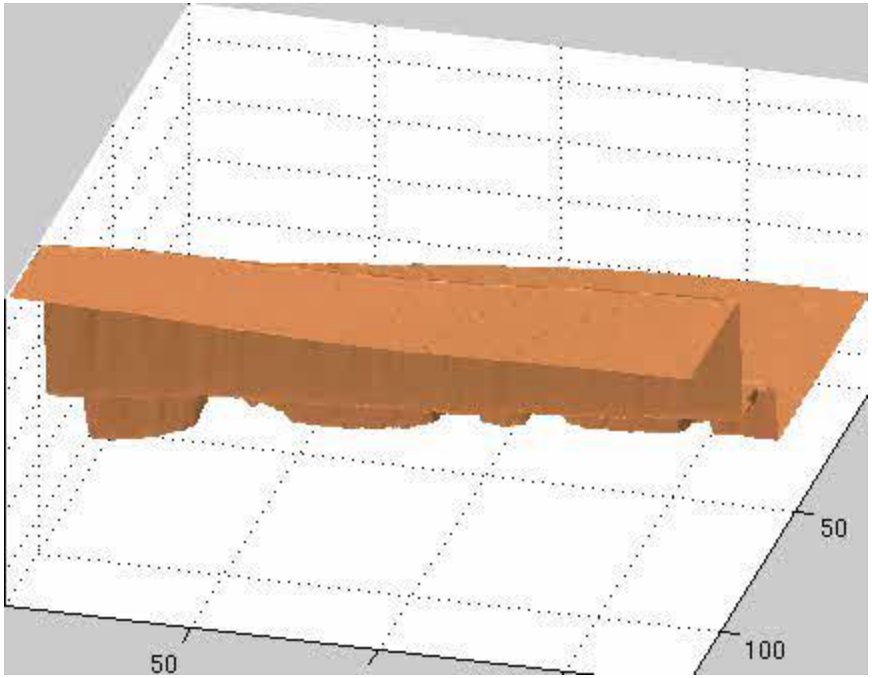
Interpretation as a Dynamical System



Treat the links as springs and shake the system

- elasticity proportional to cost
- vibration "modes" correspond to segments
  - can compute these by solving an eigenvector problem
  - for more details, see
    - » J. Shi and J. Malik, Normalized Cuts and Image Segmentation, CVPR, 1997

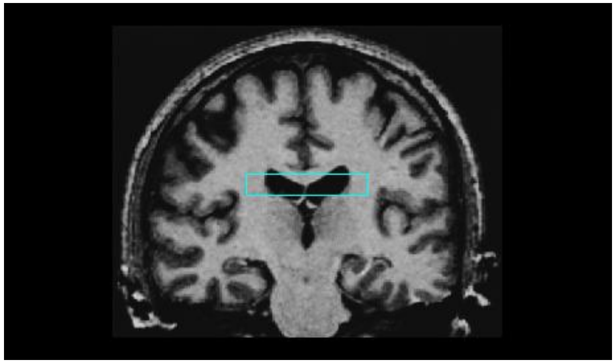
Interpretation as a Dynamical System



More on Segmentation

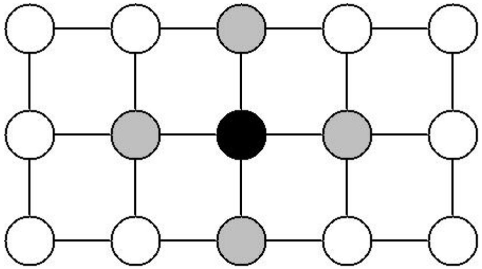
Active Contour Models

[Kass et al. 1988](#)



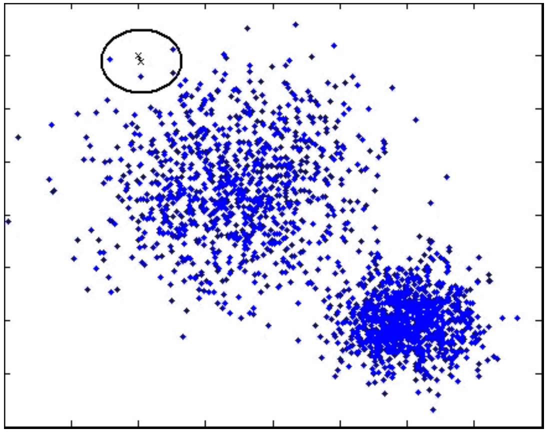
Markov Random Fields

[Boykov et al. 2001](#)



Mean Shift

[Comaniciu and Meer 2002](#)



Grabcut [Rother et al., SIGGRAPH 2004]

Intelligent Scissors (demo)

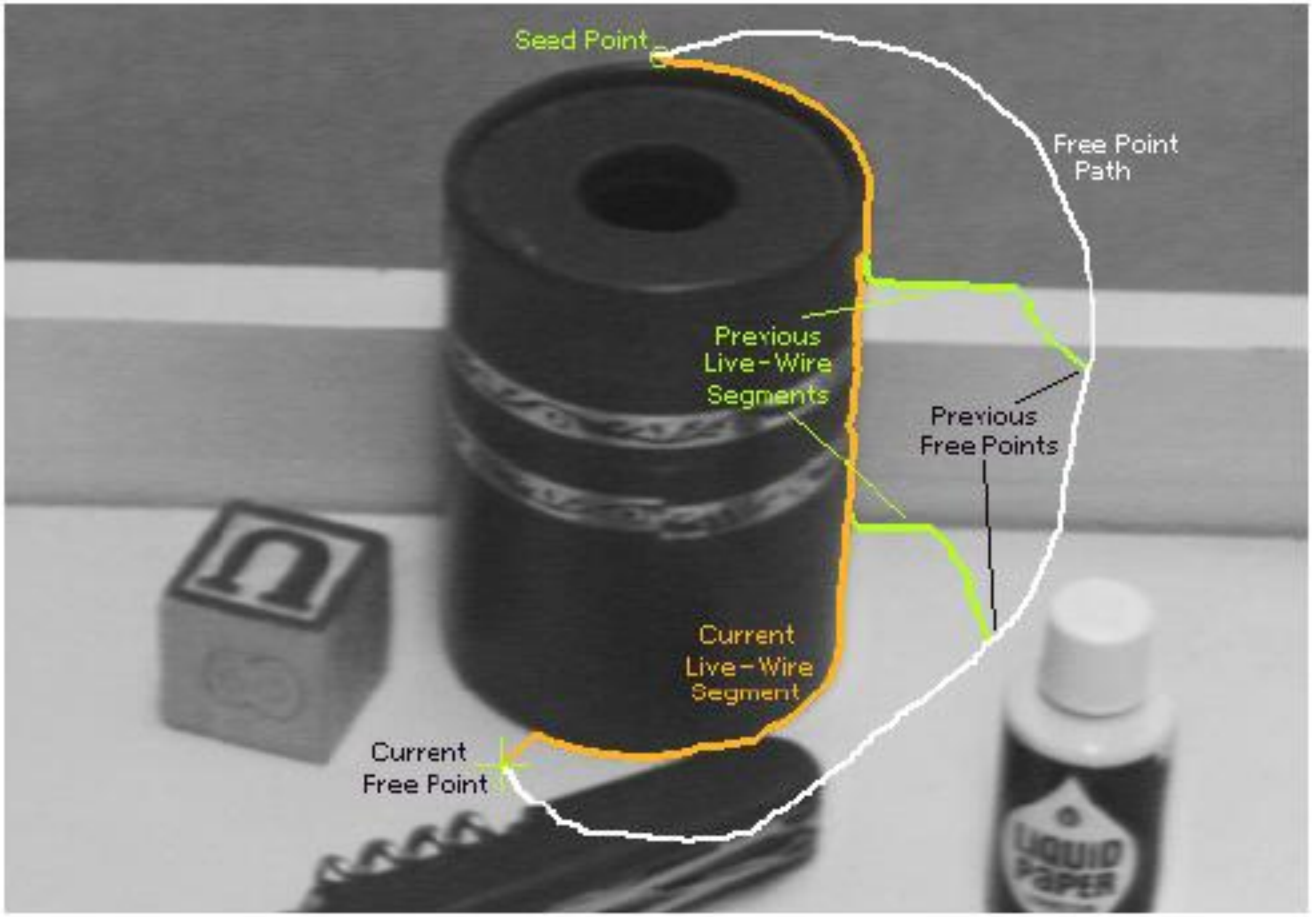
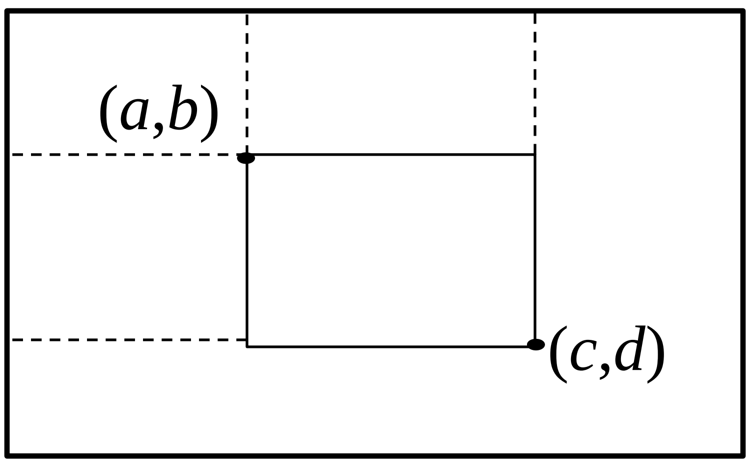


Figure 2: *Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions (t_0, t_1, and t_2) are shown in green.*

Mean Filtering with Integral Images

- Integral Image (or summed-area table)
  - Precomputation: $O(n)$
  - Sum over rectangle: $O(1)$

$$S[x, y] = \sum_{i=1}^{x} \sum_{j=1}^{y} A[i, j]$$



$$\sum_{i=a}^{c} \sum_{j=b}^{d} A[i, j] = S[c, d] - S[c, b] - S[a, d] + S[a, b]$$

Matlab code

```
function B = meanfilt(A,k)

[m n] = size(A);
p1 = ceil(k/2);
p2 = floor(k/2);

P = zeros(m+k-1,n+k-1);
P(p1:end-p2,p1:end-p2) = A / (k*k);

S = cumsum(P,1);
S = cumsum(S,2);

B =    S(k:end,k:end)
     - S(1:end-k+1,k:end)
     - S(k:end,1:end-k+1)
     + S(1:end-k+1,1:end-k+1);
```