

**BBM 413**

**Fundamentals of  
Image Processing**

Dec. 18, 2012

Erkut Erdem

Dept. of Computer Engineering

Hacettepe University

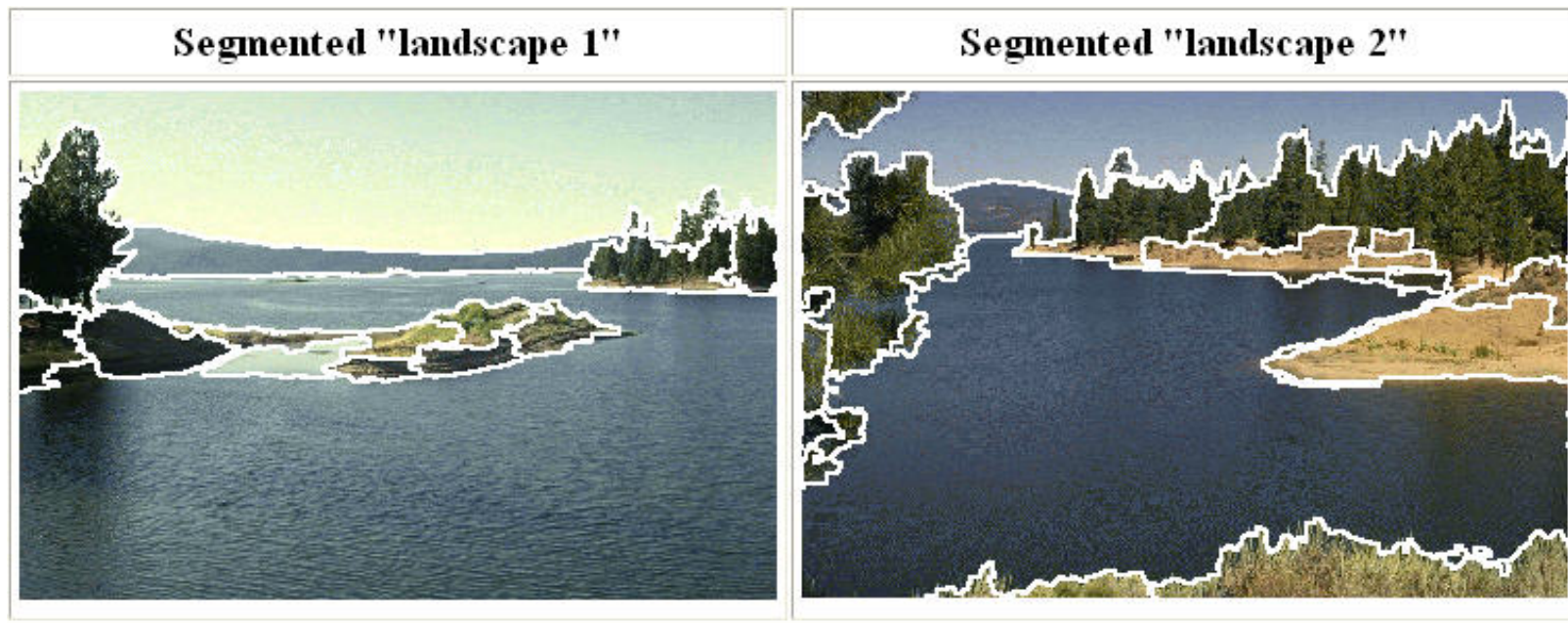
**Segmentation – Part 2**

# Segmentation methods

- Segment foreground from background
- Histogram-based segmentation
- Segmentation as clustering
  - K-means clustering
  - Mean-shift segmentation
- Graph-theoretic segmentation
  - Min cut
  - Normalized cuts
- Interactive segmentation

# Mean shift clustering and segmentation

- An advanced and versatile technique for clustering-based segmentation

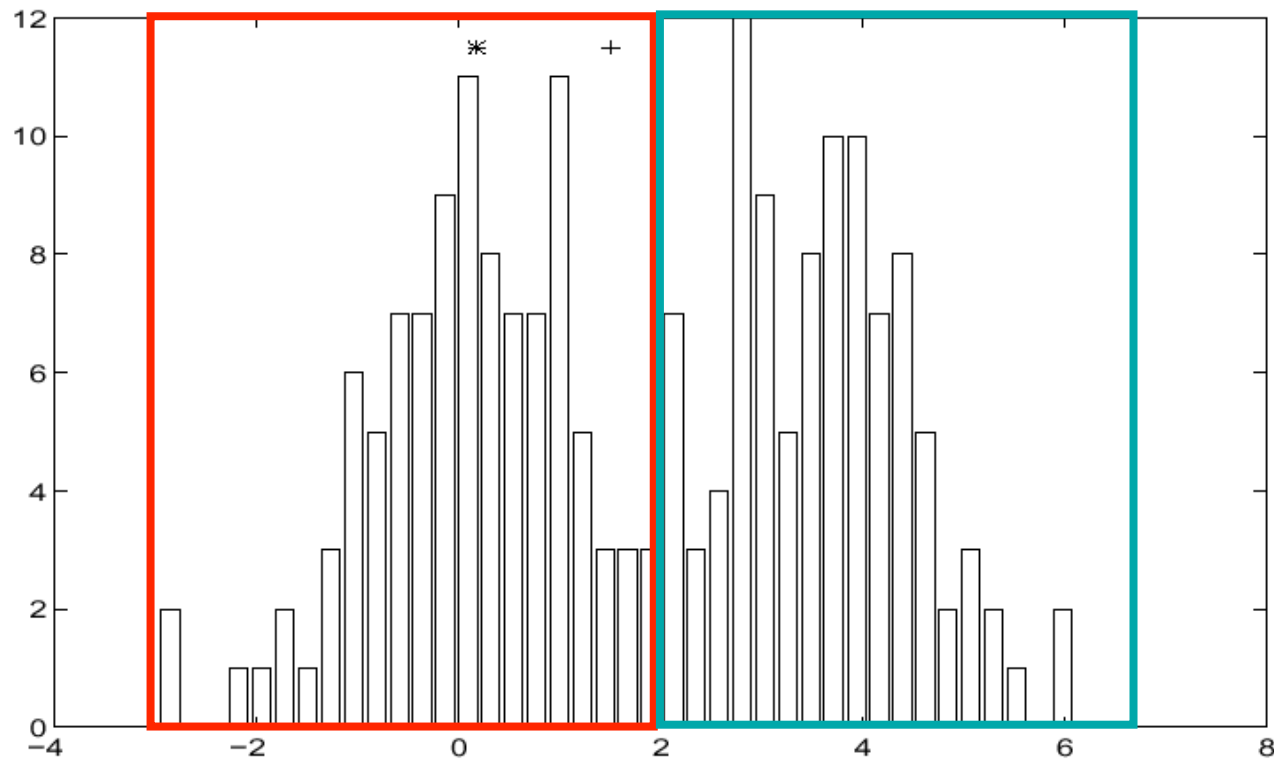


<http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html>

D. Comaniciu and P. Meer, [Mean Shift: A Robust Approach toward Feature Space Analysis](#), PAMI 2002.

Slide credit: S. Lazebnik

# Finding Modes in a Histogram



- How Many Modes Are There?
  - Easy to see, hard to compute

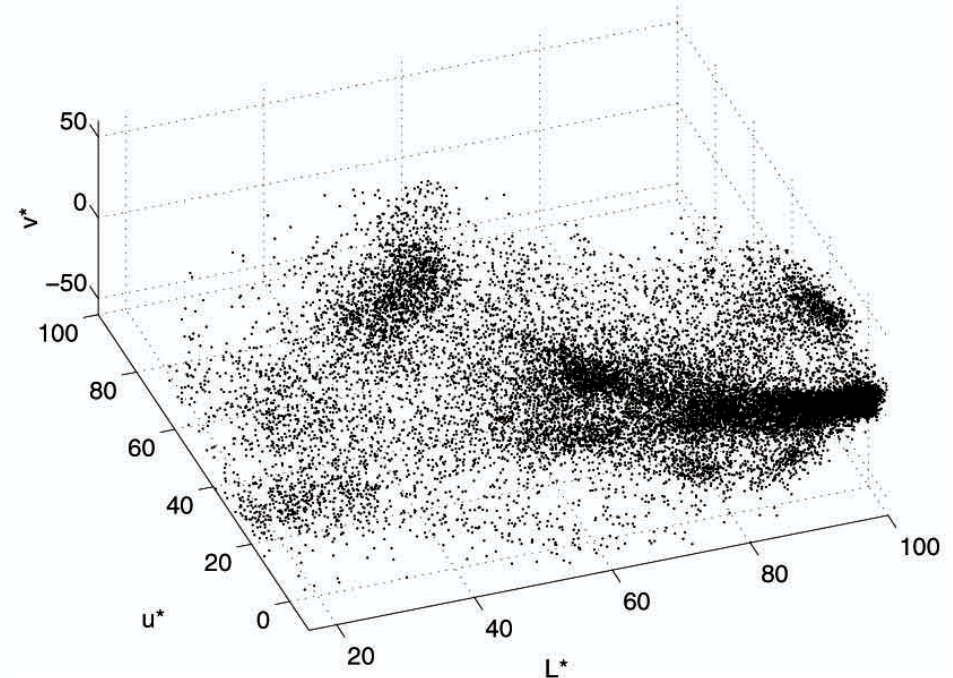
# Mean shift algorithm

- The mean shift algorithm seeks *modes* or local maxima of density in the feature space

image



Feature space  
( $L^*u^*v^*$  color values)

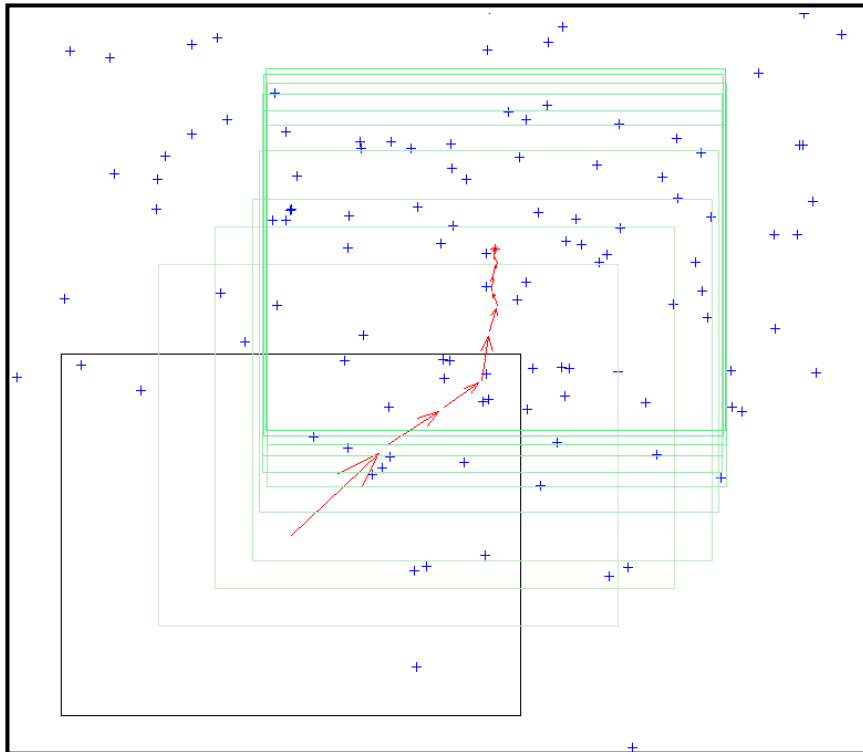


# Mean shift algorithm

## Mean Shift Algorithm

1. Choose a search window size.
2. Choose the initial location of the search window.
3. Compute the mean location (centroid of the data) in the search window.
4. Center the search window at the mean location computed in Step 3.
5. Repeat Steps 3 and 4 until convergence.

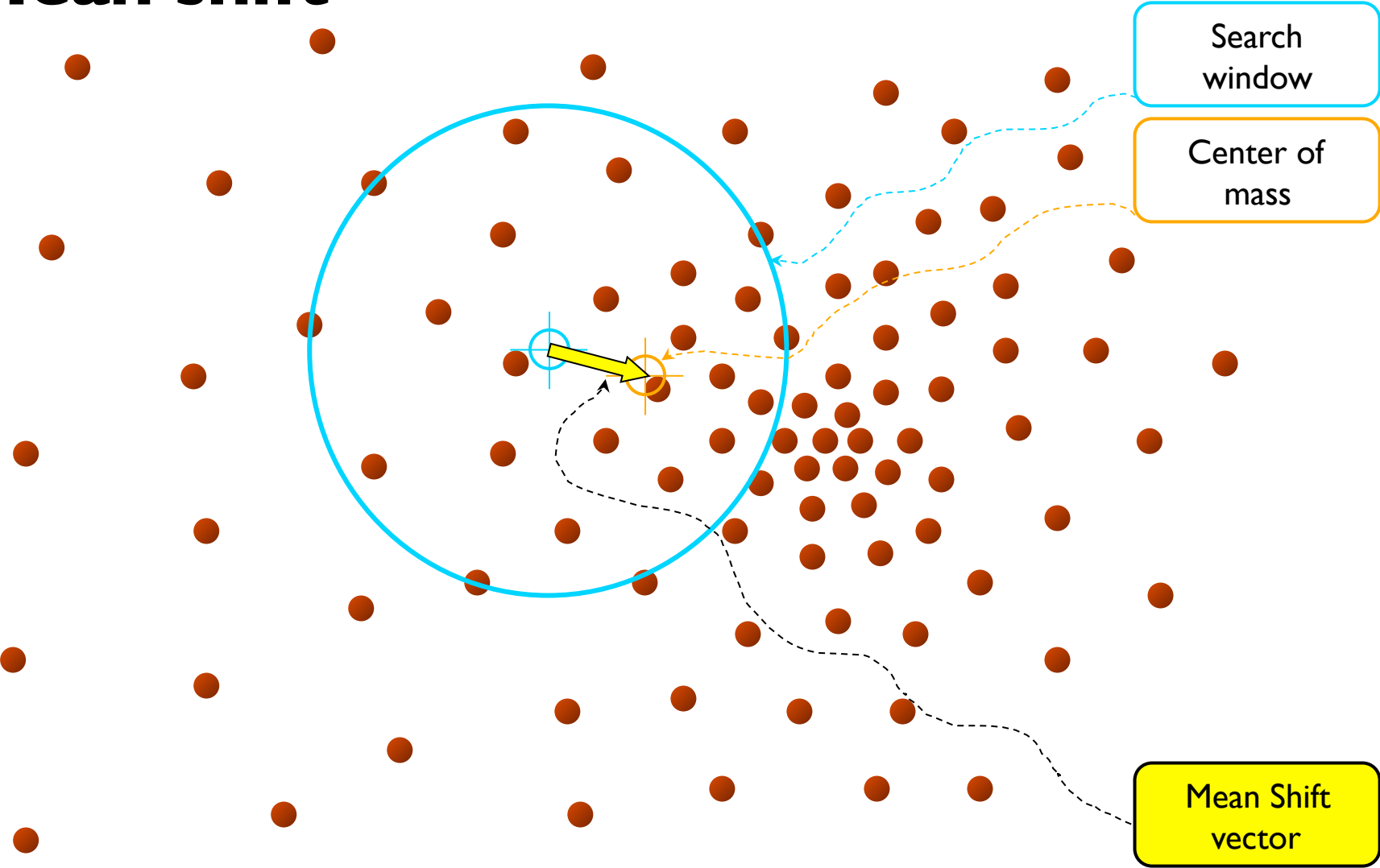
The mean shift algorithm seeks the “mode” or point of highest density of a data distribution:



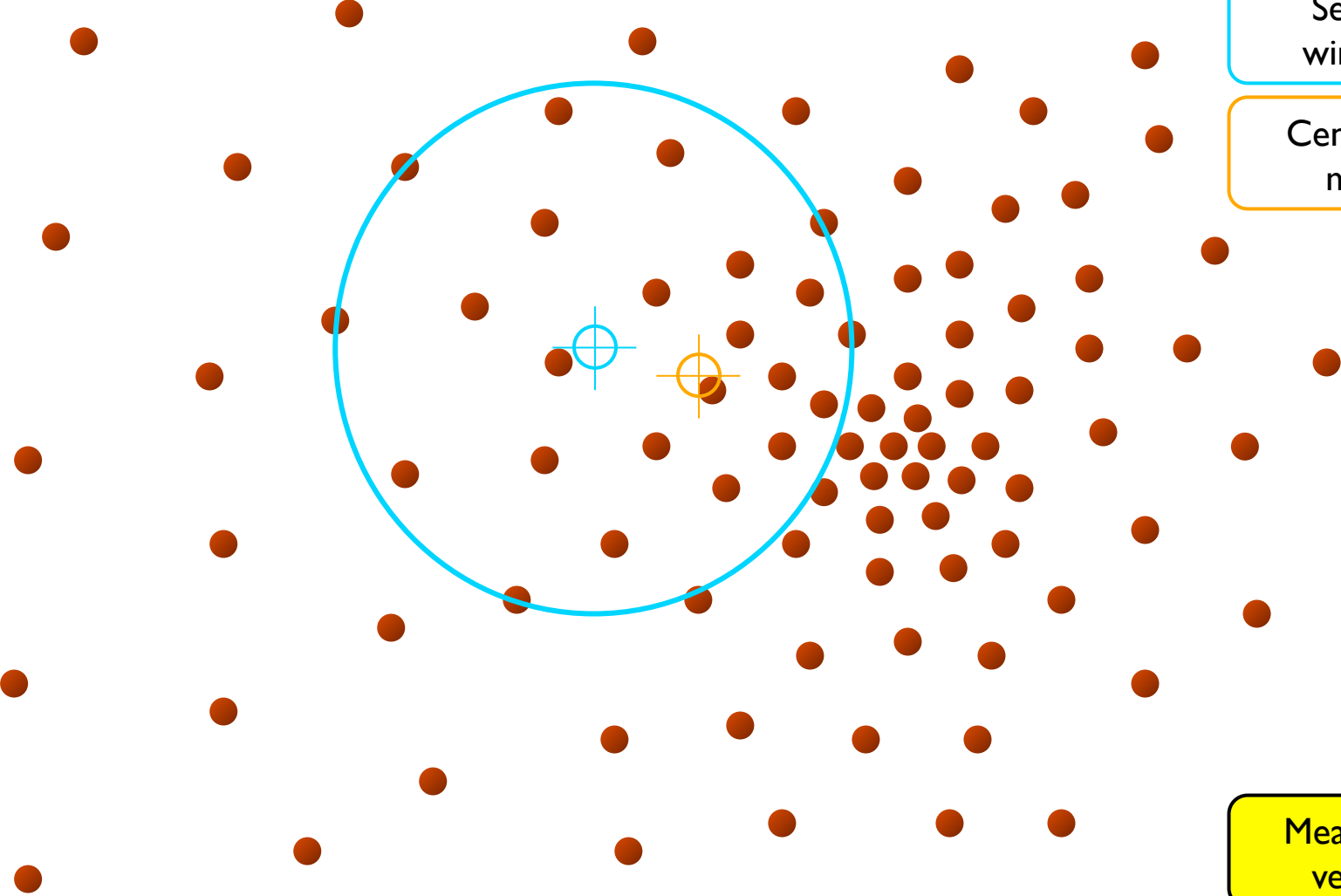
Two issues:

- (1) Kernel to interpolate density based on sample positions.
- (2) Gradient ascent to mode.

# Mean shift



# Mean shift



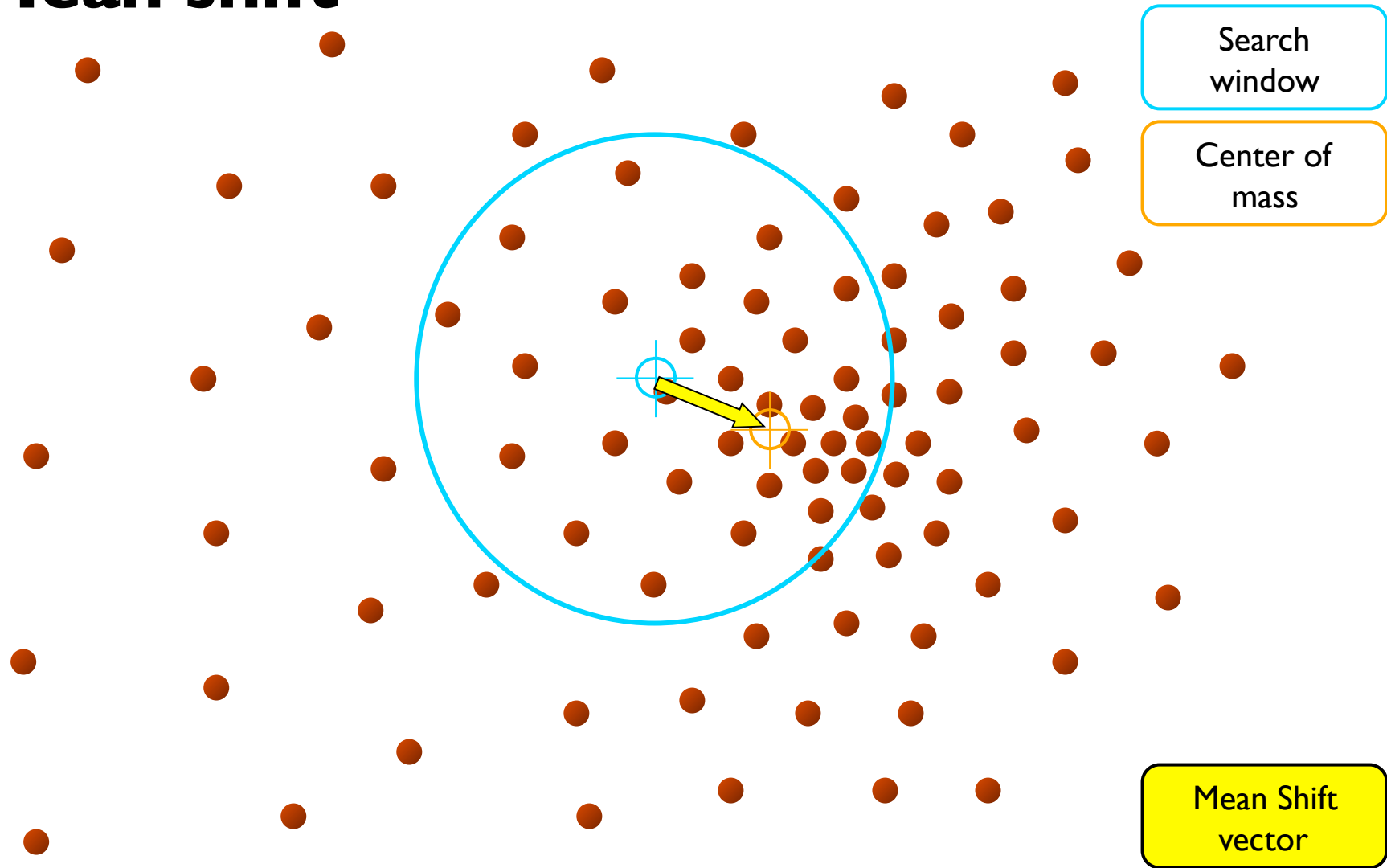
Search window

Center of mass

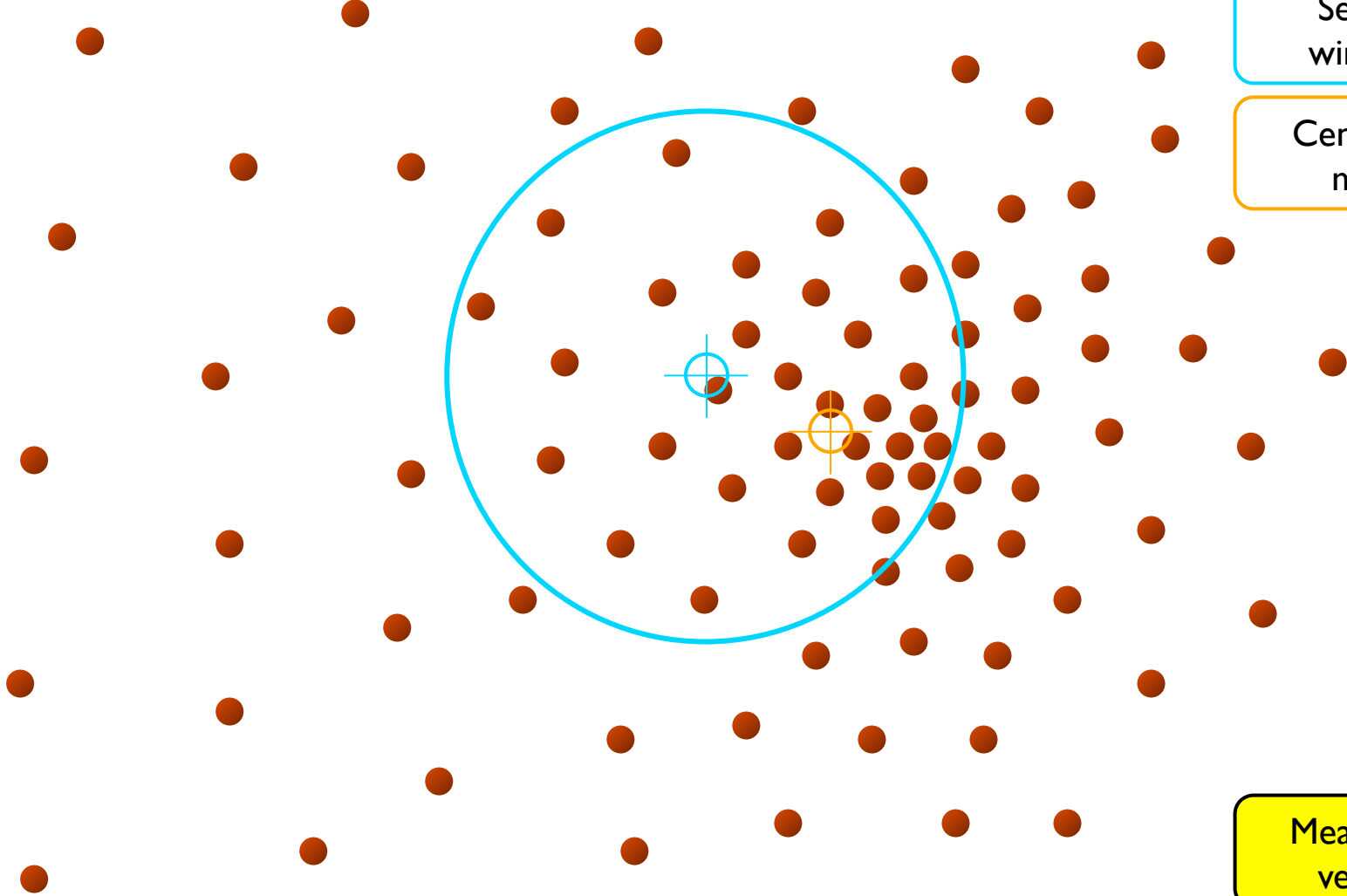
Mean Shift vector



# Mean shift



# Mean shift

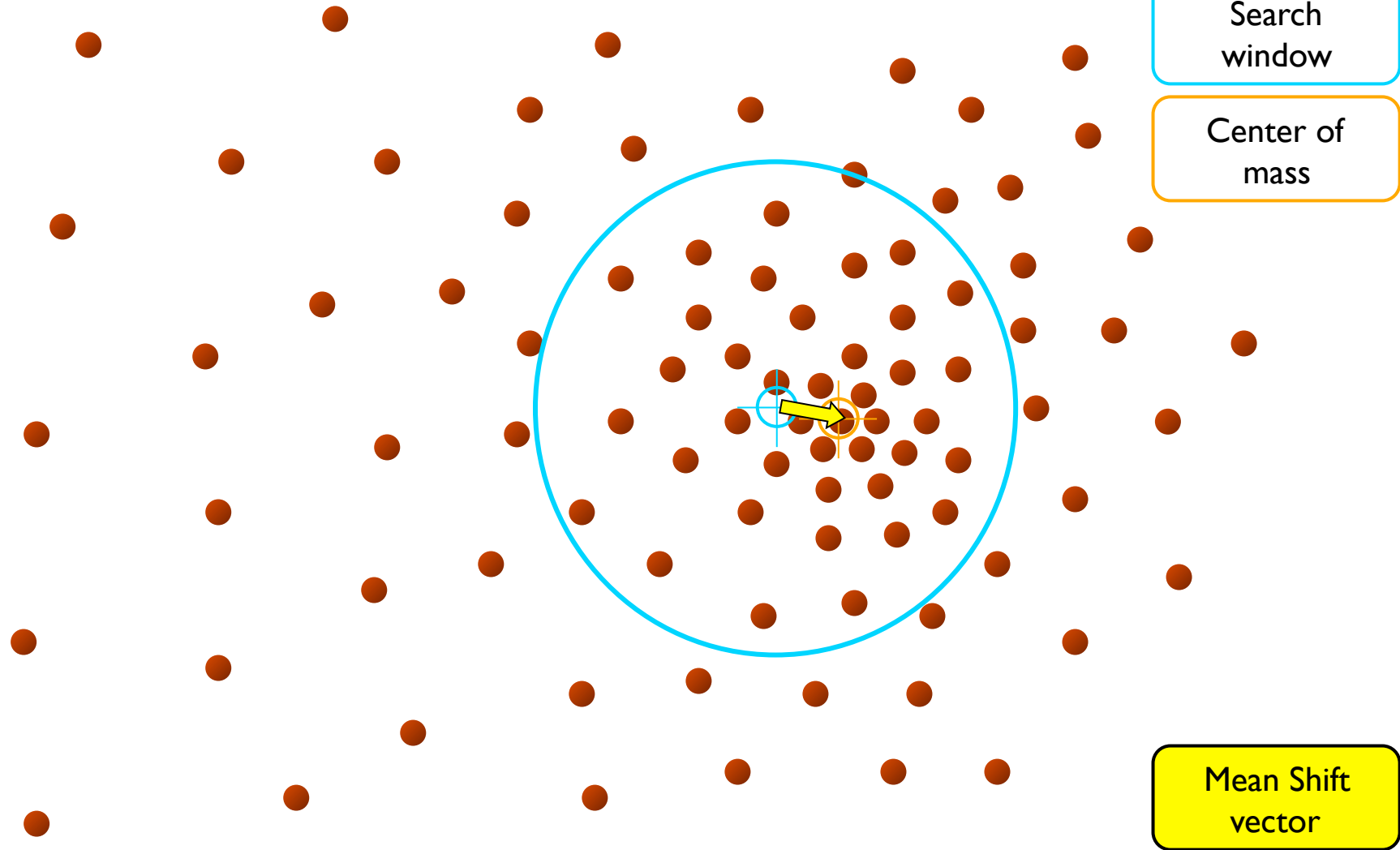


Search window

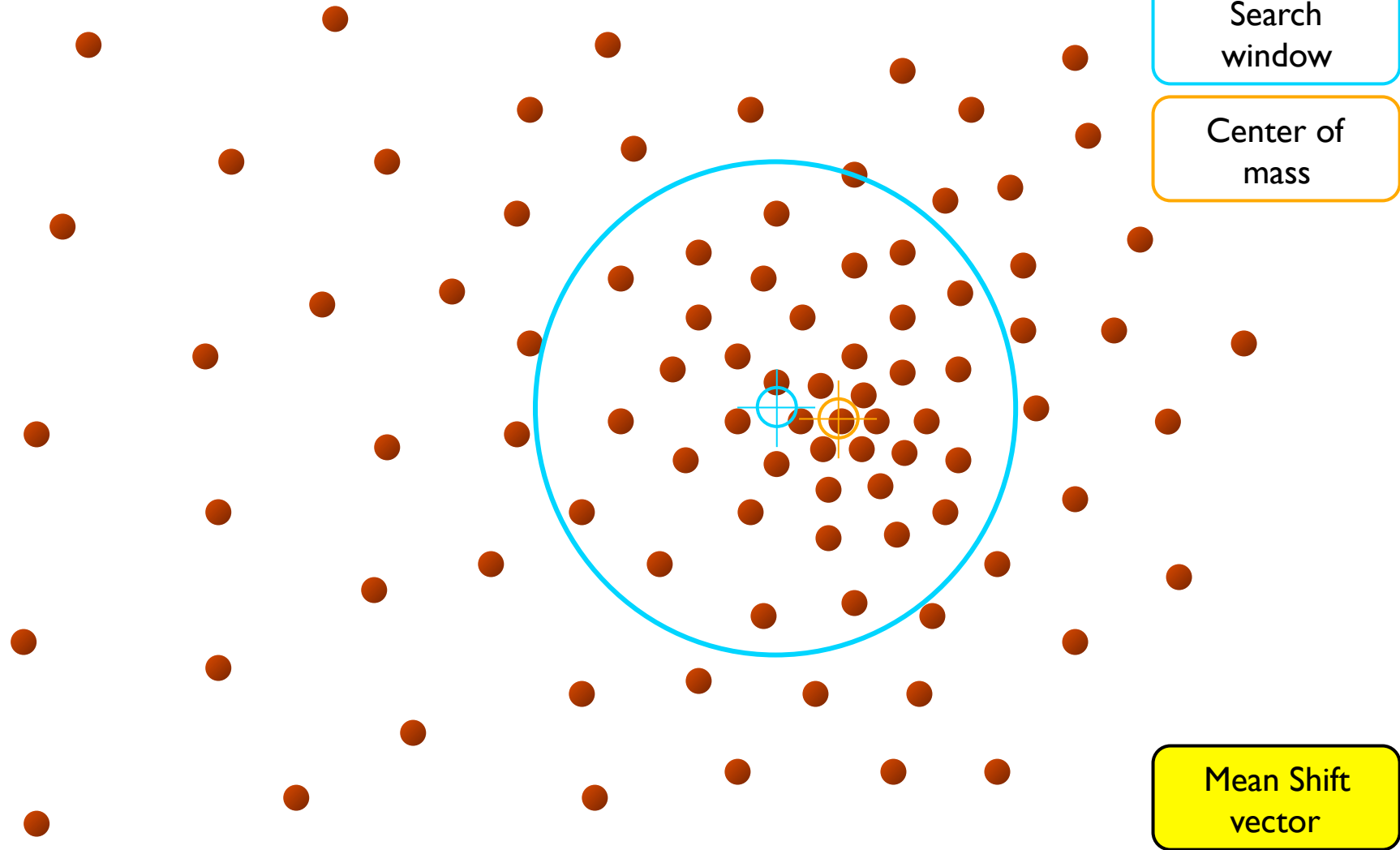
Center of mass

Mean Shift vector

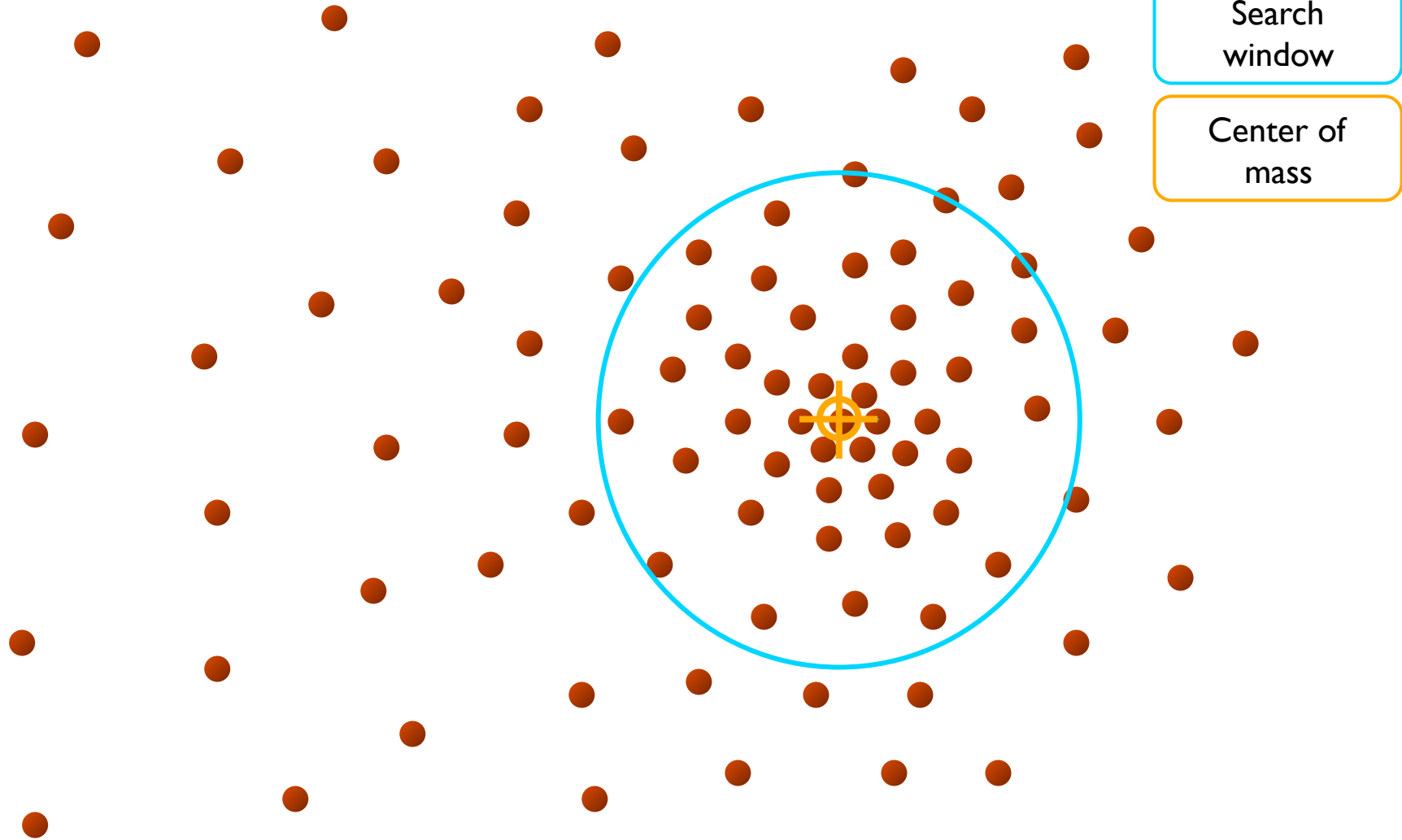
# Mean shift



# Mean shift

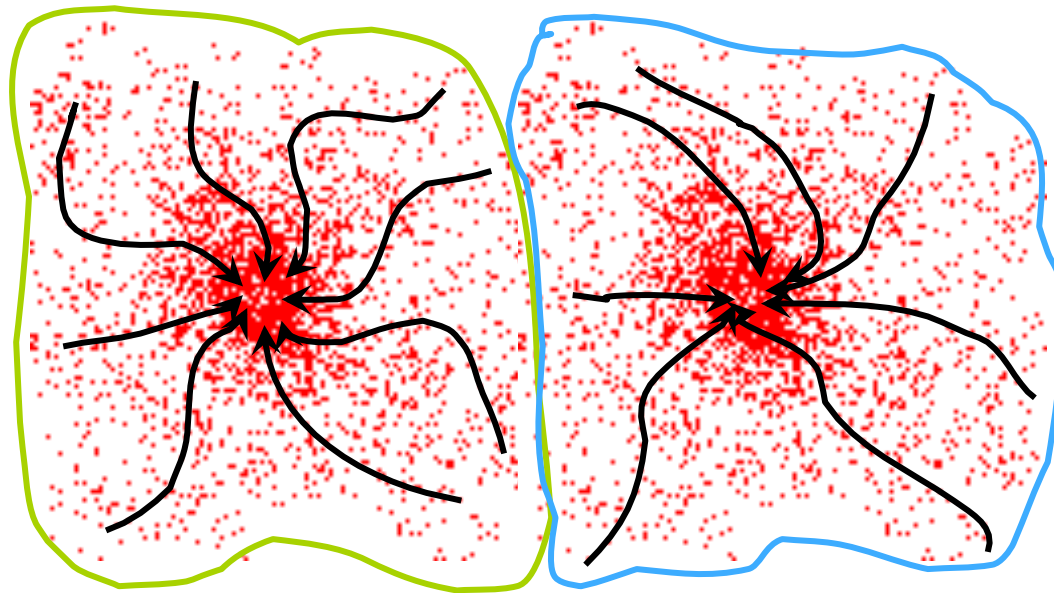


# Mean shift



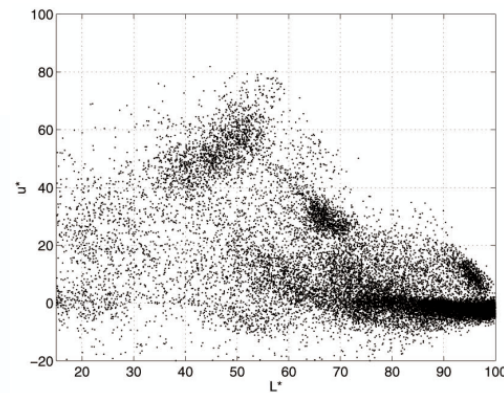
# Mean shift clustering

- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode

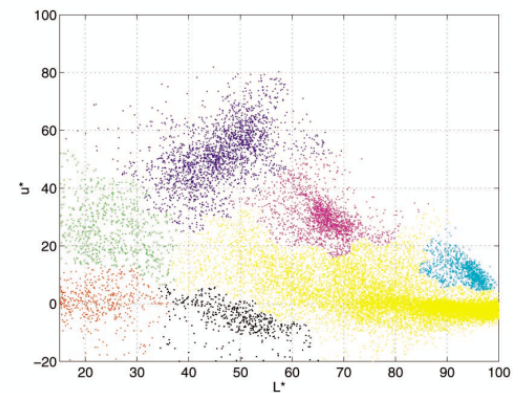


# Mean shift clustering/segmentation

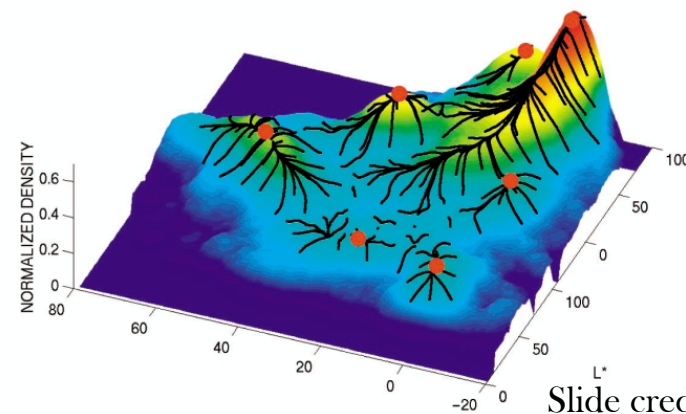
- Find features (color, gradients, texture, etc)
- Initialize windows at individual feature points
- Perform mean shift for each window until convergence
- Merge windows that end up near the same “peak” or mode



(a)



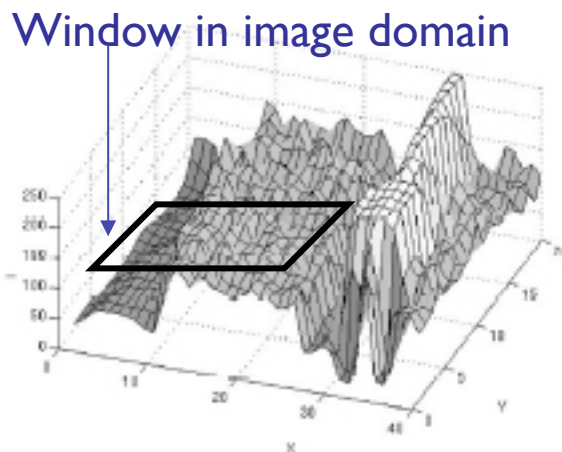
(b)



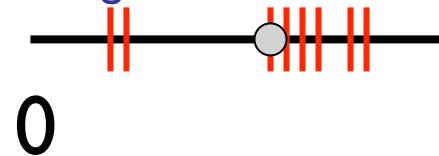
Slide credit: S. Lazebnik

# Apply mean shift jointly in the image (left col.) and range (right col.) domains

1

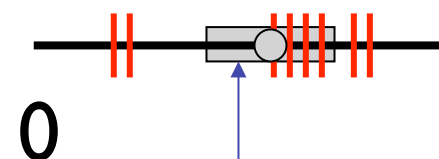
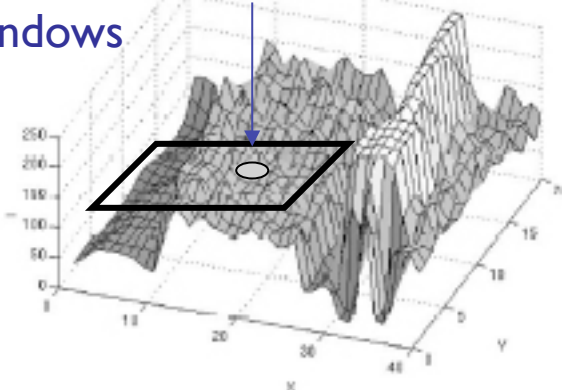


Intensities of pixels within image domain window



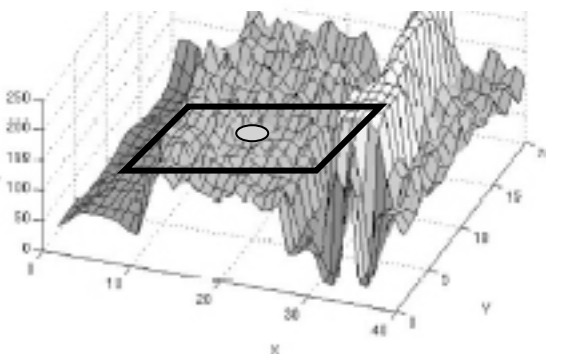
2

Center of mass of pixels within both image and range domain windows

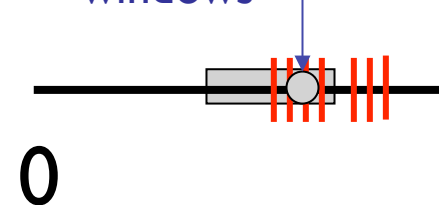


3

4

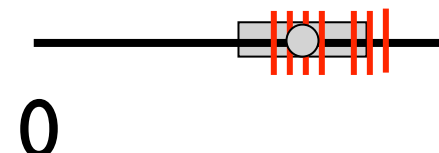
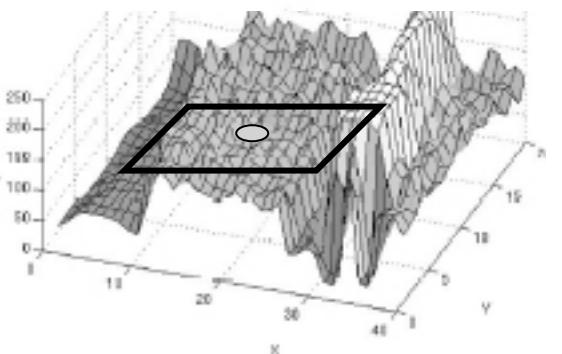


Center of mass of pixels within both image and range domain windows



6

5



7



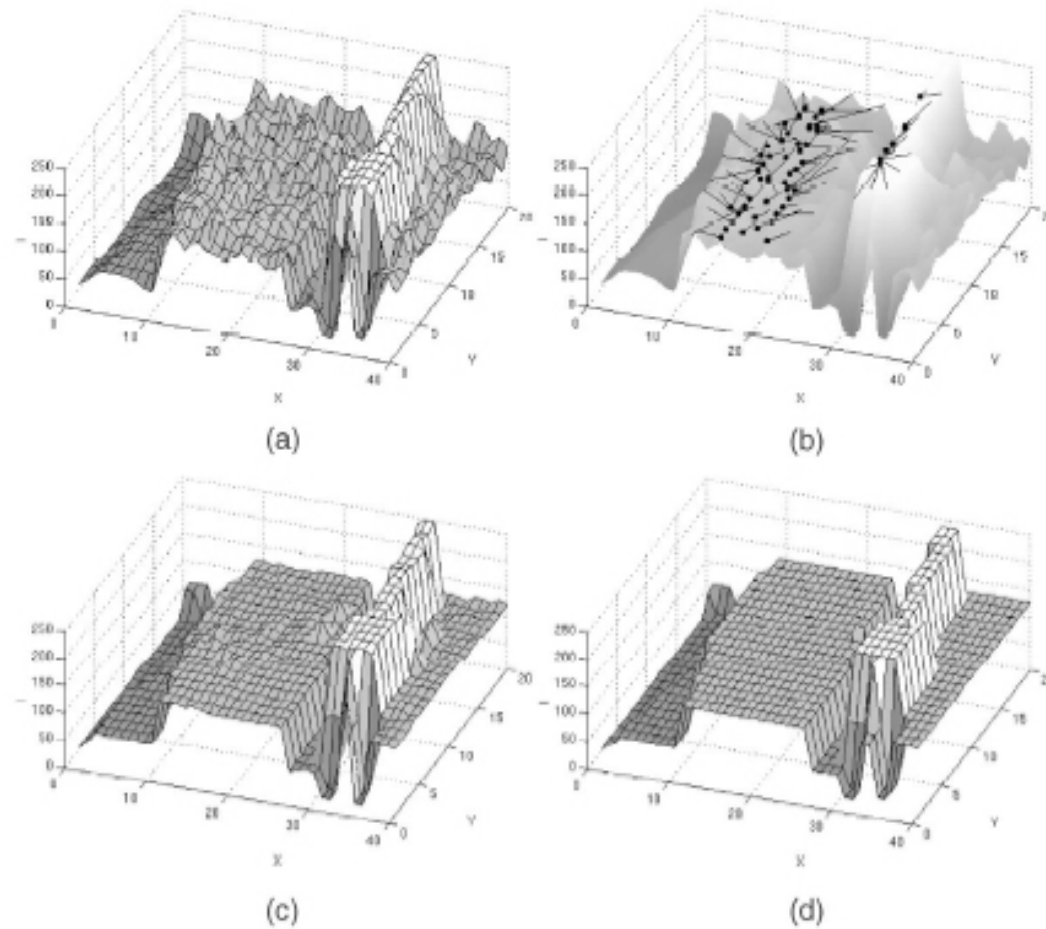
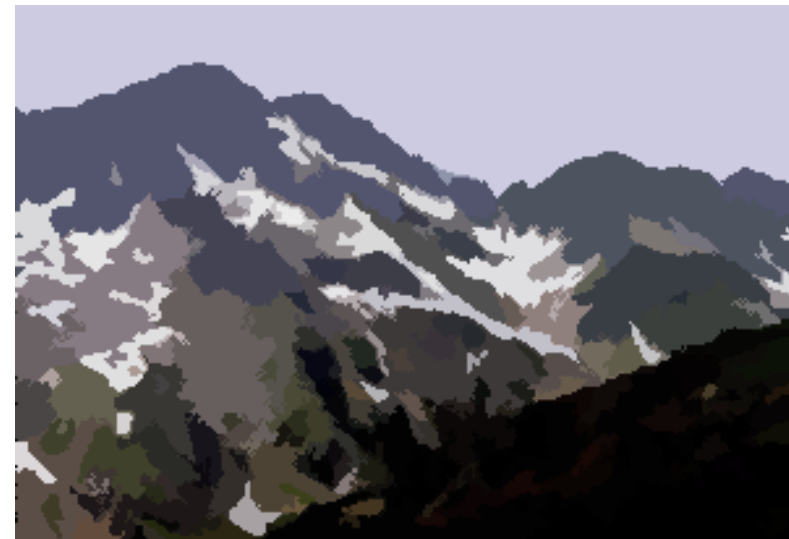


Fig. 4. Visualization of mean shift-based filtering and segmentation for gray-level data. (a) Input. (b) Mean shift paths for the pixels on the plateau and on the line. The black dots are the points of convergence. (c) Filtering result  $(h_s, h_r) = (8, 4)$ . (d) Segmentation result.

# Mean shift segmentation results



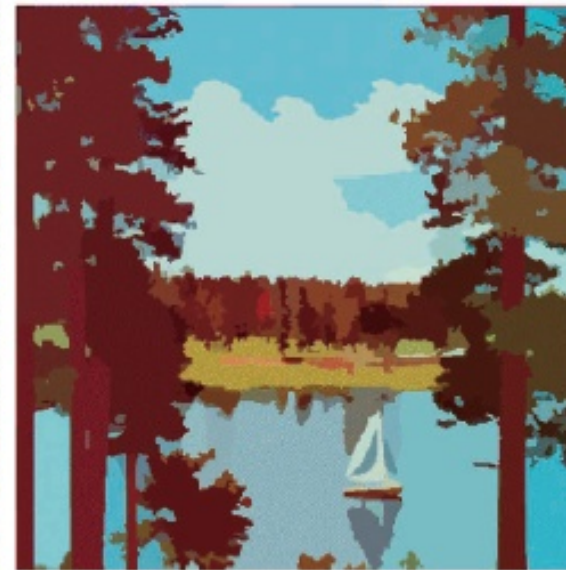
<http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html>

Slide credit: S. Lazebnik

# More results



# More results



# Mean shift pros and cons

- Pros
  - Does not assume spherical clusters
  - Just a single parameter (window size)
  - Finds variable number of modes
  - Robust to outliers
- Cons
  - Output depends on window size
  - Computationally expensive
  - Does not scale well with dimension of feature space

# Segmentation methods

- Segment foreground from background
- Histogram-based segmentation
- Segmentation as clustering
  - K-means clustering
  - Mean-shift segmentation
- **Graph-theoretic segmentation**
  - Min cut
  - Normalized cuts
- Interactive Segmentation

# Graph-Theoretic Image Segmentation

Build a weighted graph  $G=(V,E)$  from image



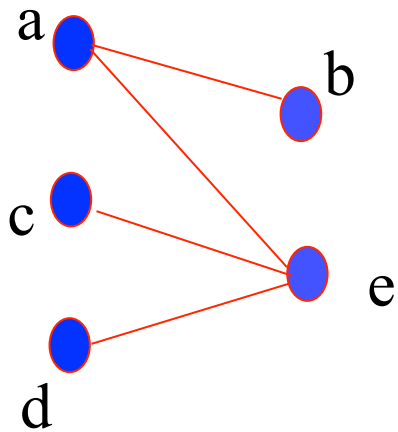
$V$ : image pixels

$E$ : connections between pairs of nearby pixels

$W_{ij}$ : probability that  $i$  &  $j$  belong to the same region

Segmentation = graph partition

# Graphs Representations

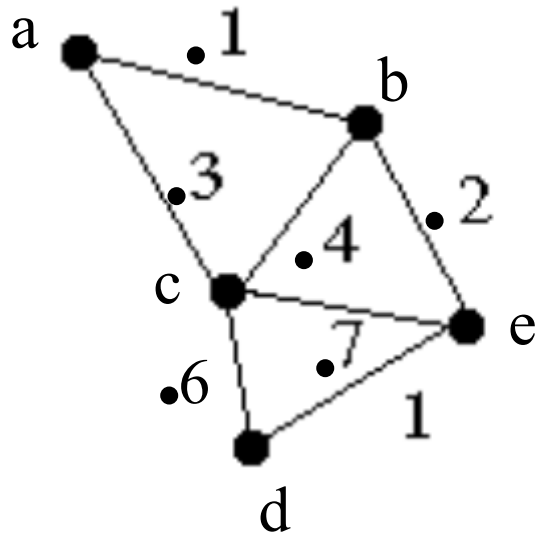


	a	b	c	d	e
a	0	1	0	0	1
b	1	0	0	0	0
c	0	0	0	0	1
d	0	0	0	0	1
e	1	0	1	1	0

Adjacency Matrix



# A Weighted Graph and its Representation

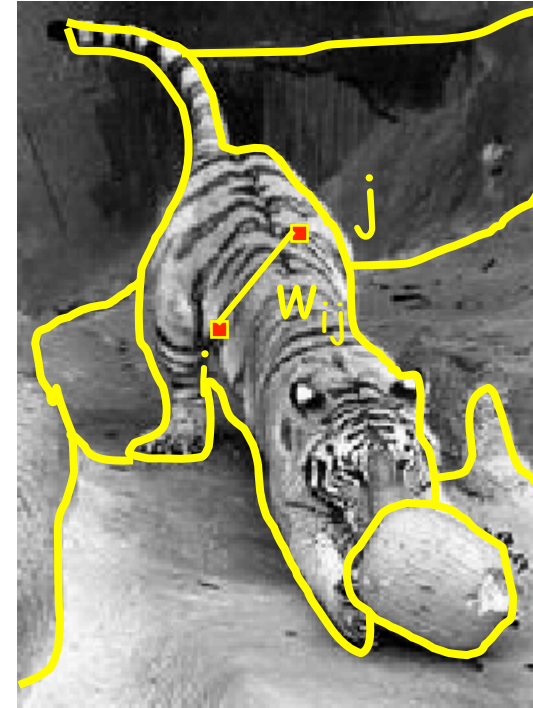
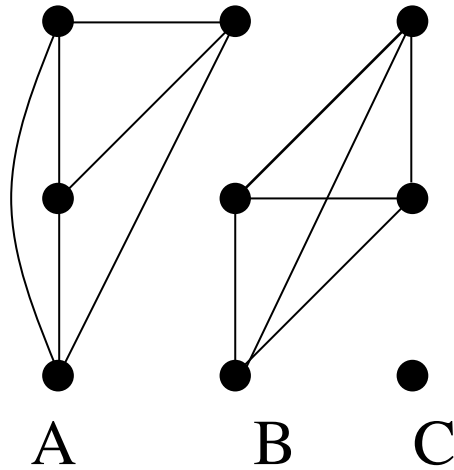


Affinity Matrix

$$W = \begin{bmatrix} 1 & .1 & .3 & 0 & 0 \\ .1 & 1 & .4 & 0 & .2 \\ .3 & .4 & 1 & .6 & .7 \\ 0 & 0 & .6 & 1 & 1 \\ 0 & .2 & .7 & 1 & 1 \end{bmatrix}$$

$W_{ij}$  : probability that i & j  
belong to the same  
region

# Segmentation by graph partitioning



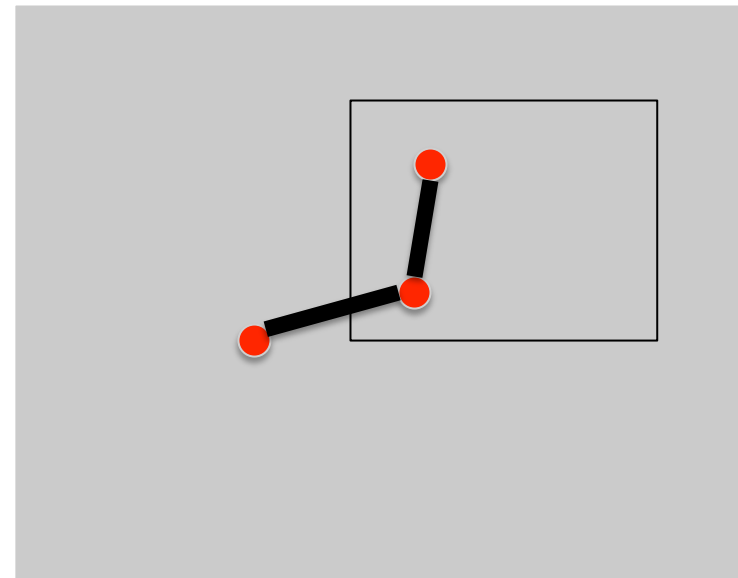
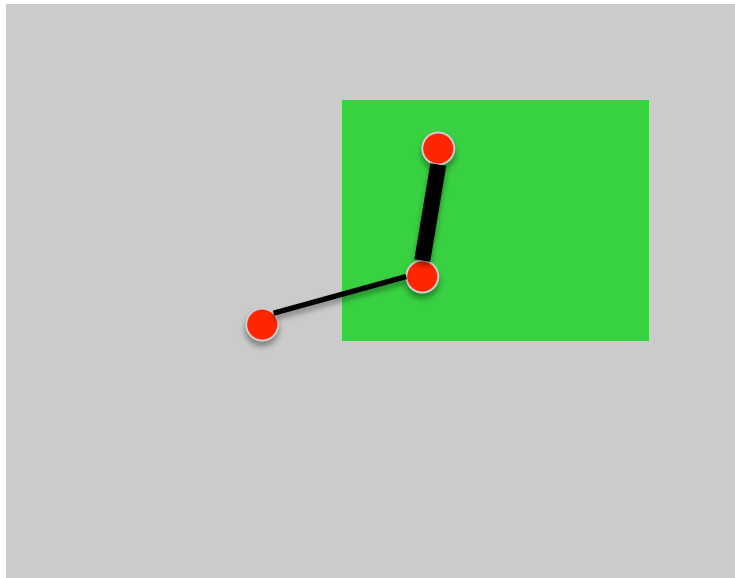
- Break graph into segments
  - Delete links that cross between segments
  - Easiest to break links that have low affinity
    - similar pixels should be in the same segments
    - dissimilar pixels should be in different segments

# Affinity between pixels

Similarities among pixel descriptors

$$W_{ij} = \exp(-\|z_i - z_j\|^2 / s^2)$$

←  $s$  = Scale factor...  
it will hunt us later



# Affinity between pixels

Similarities among pixel descriptors

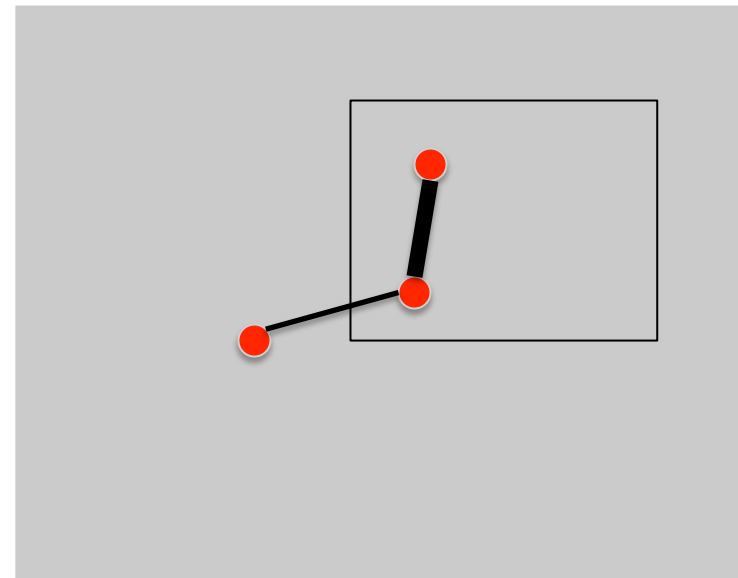
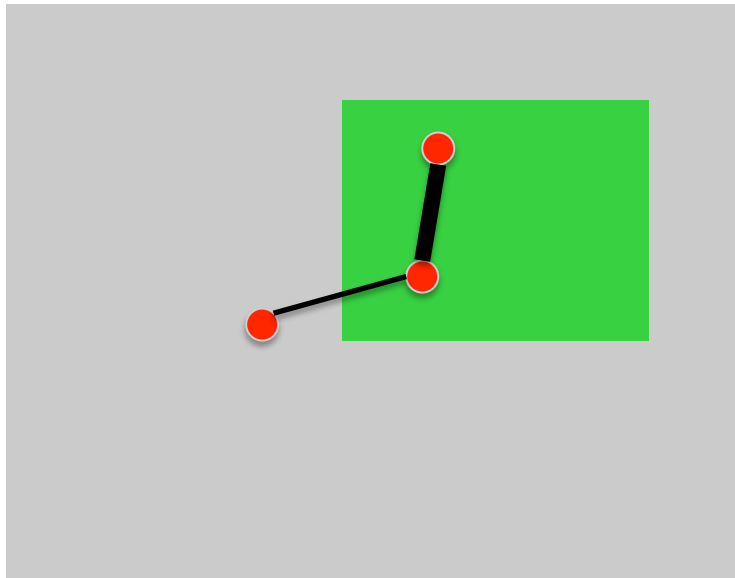
$$W_{ij} = \exp(-\|z_i - z_j\|^2 / s^2)$$

← s = Scale factor...  
it will hunt us later

Interleaving edges

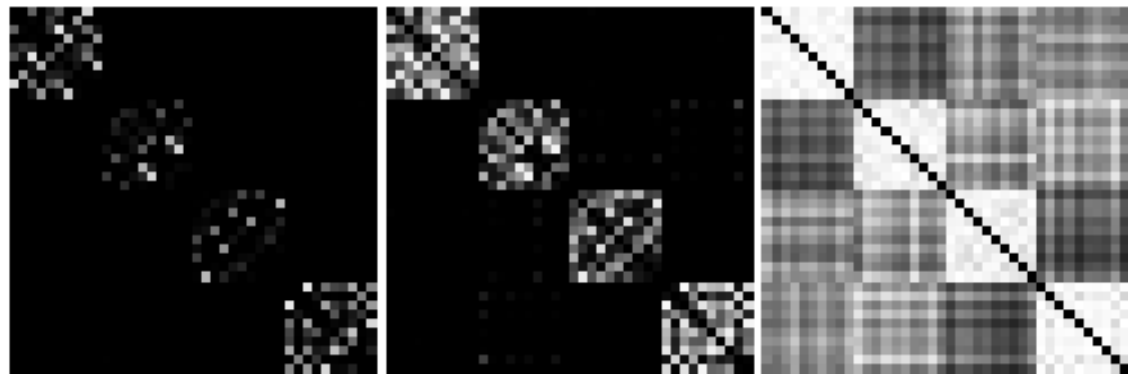
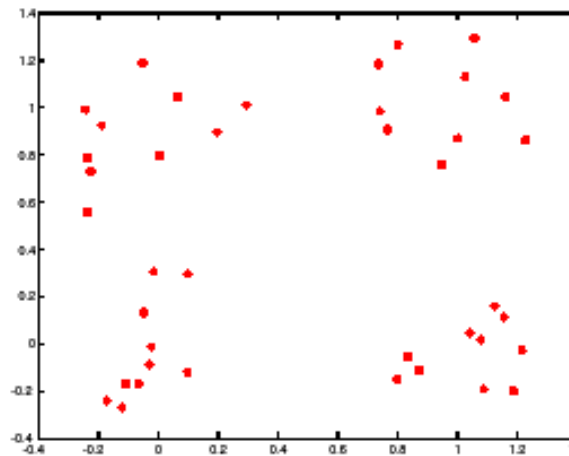
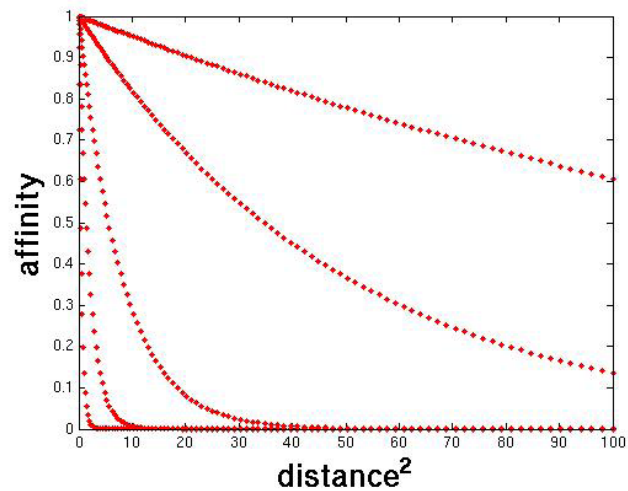
$$W_{ij} = 1 - \max_{\text{Line between } i \text{ and } j} P_b$$

With  $P_b$  = probability of boundary



# Scale affects affinity

- Small  $\sigma$ : group only nearby points
- Large  $\sigma$ : group far-away points

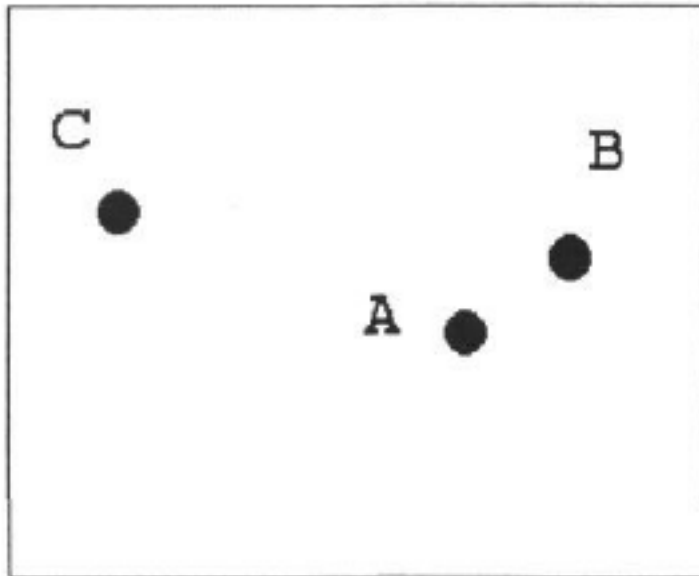


# Feature grouping by “relocalisation” of eigenvectors of the proximity matrix

British Machine Vision Conference, pp. 103-108, 1990

Guy L. Scott  
Robotics Research Group  
Department of Engineering Science  
University of Oxford

H. Christopher Longuet-Higgins  
University of Sussex  
Falmer  
Brighton



Three points in feature space

$$W_{ij} = \exp(-\|z_i - z_j\|^2 / \sigma^2)$$

With an appropriate  $\sigma$

W=

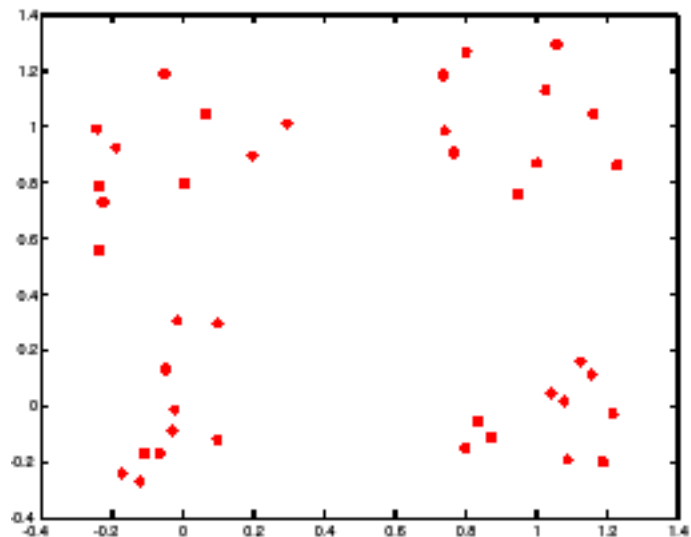
	A	B	C
A	1.00	0.63	0.03
B	0.63	1.00	0.0
C	0.03	0.0	1.00

The eigenvectors of W are:

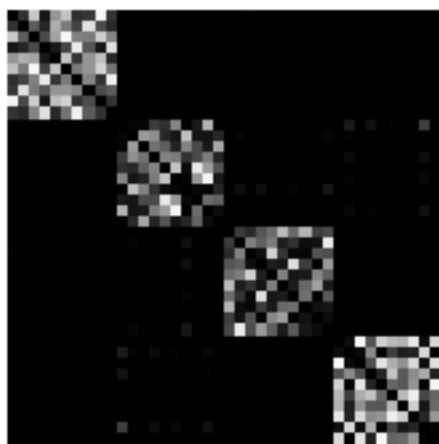
	$E_1$	$E_2$	$E_3$
Eigenvalues	1.63	1.00	0.37
A	-0.71	-0.01	0.71
B	-0.71	-0.05	-0.71
C	-0.04	1.00	-0.03

The first 2 eigenvectors group the points as desired...

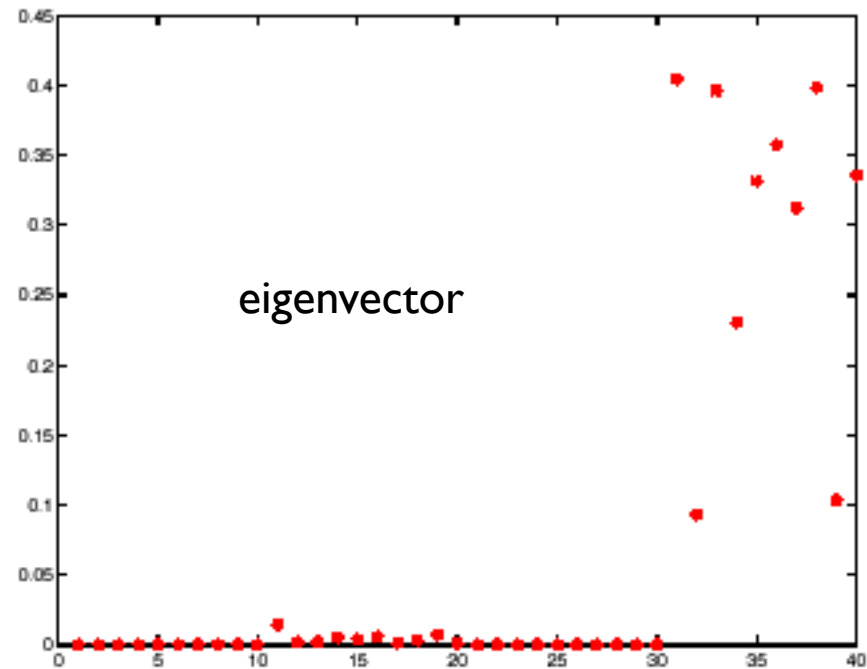
# Example eigenvector



points

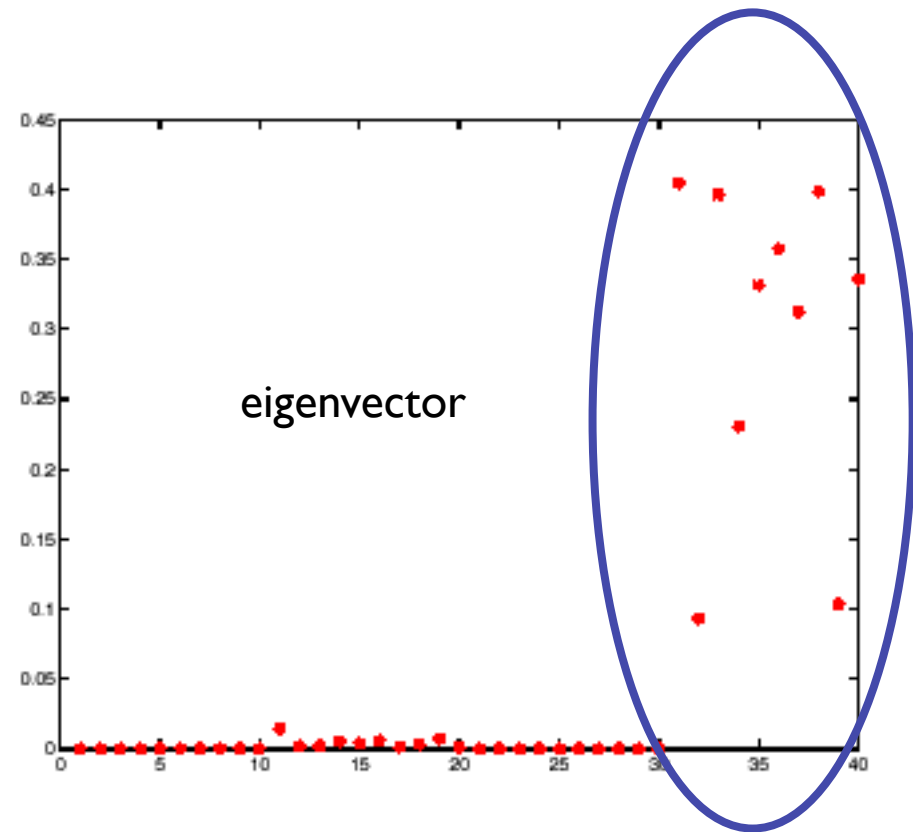
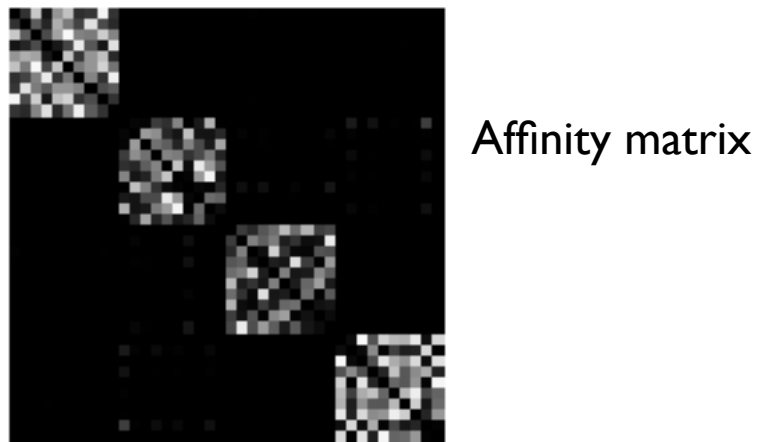
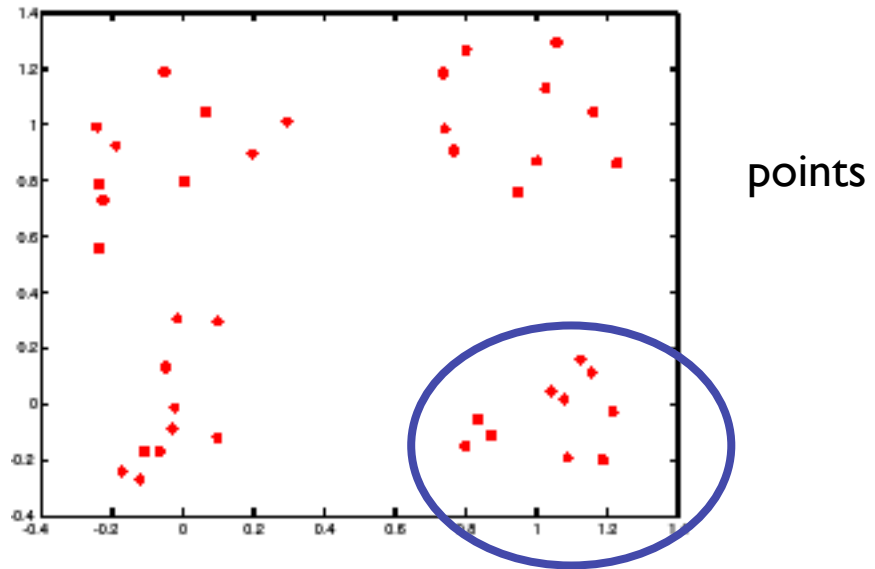


Affinity matrix



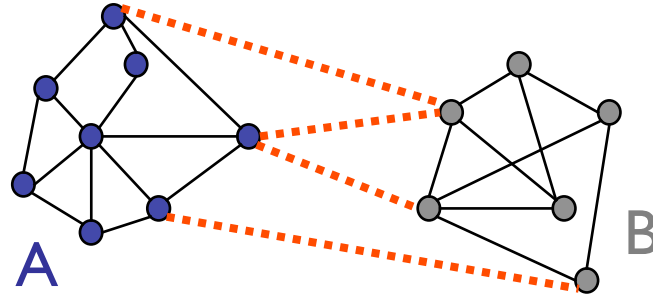
eigenvector

# Example eigenvector





# Graph cut



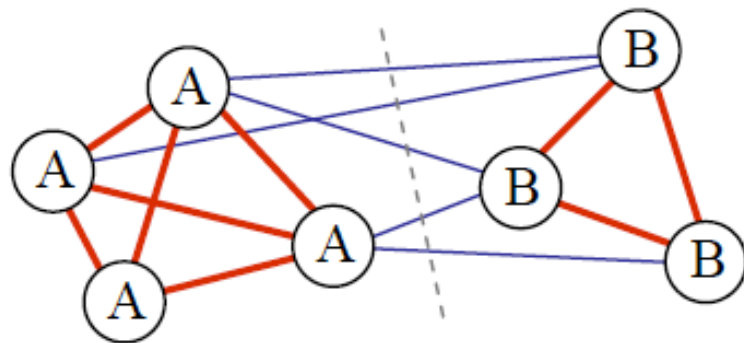
- Set of edges whose removal makes a graph disconnected
- Cost of a cut: sum of weights of cut edges
- A graph cut gives us a segmentation
  - What is a “good” graph cut and how do we find one?

# Segmentation methods

- Segment foreground from background
- Histogram-based segmentation
- Segmentation as clustering
  - K-means clustering
  - Mean-shift segmentation
- **Graph-theoretic segmentation**
  - Min cut
  - Normalized cuts
- Interactive segmentation

# Minimum cut

A cut of a graph  $G$  is the set of edges  $S$  such that removal of  $S$  from  $G$  disconnects  $G$ .



**Cut:** sum of the weight of the cut edges:

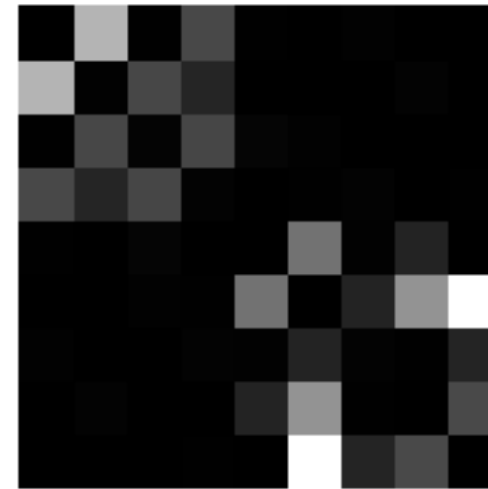
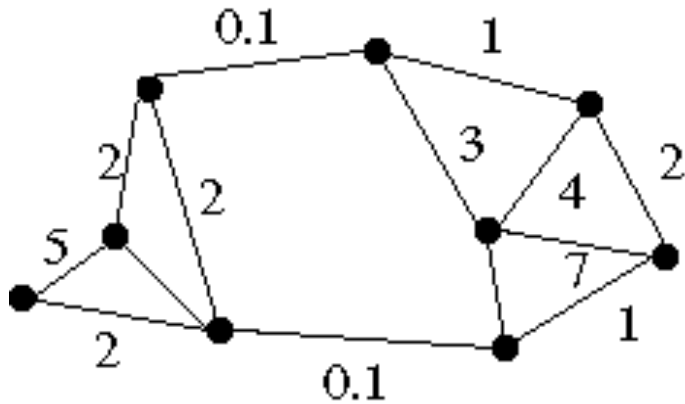
$$cut(A,B) = \sum_{u \in A, v \in B} W(u,v),$$

with  $A \cap B = \emptyset$

# Minimum cut

- We can do segmentation by finding the *minimum cut* in a graph
  - Efficient algorithms exist for doing this

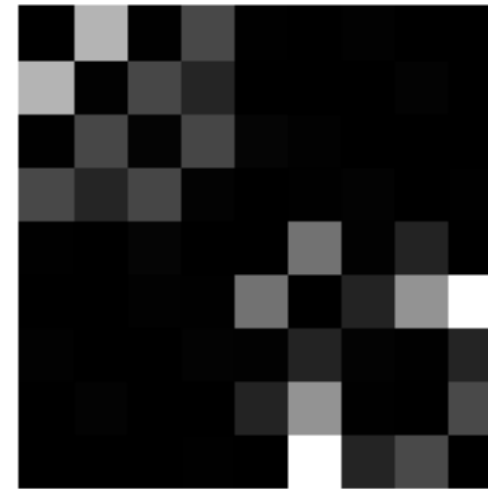
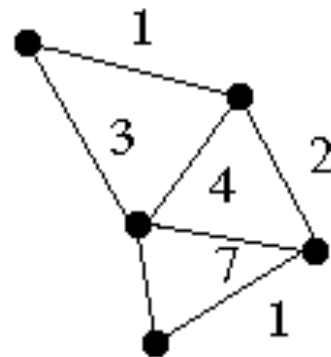
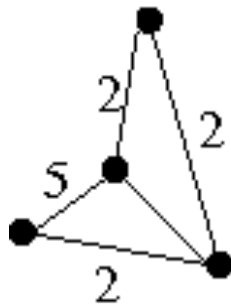
Minimum cut example



# Minimum cut

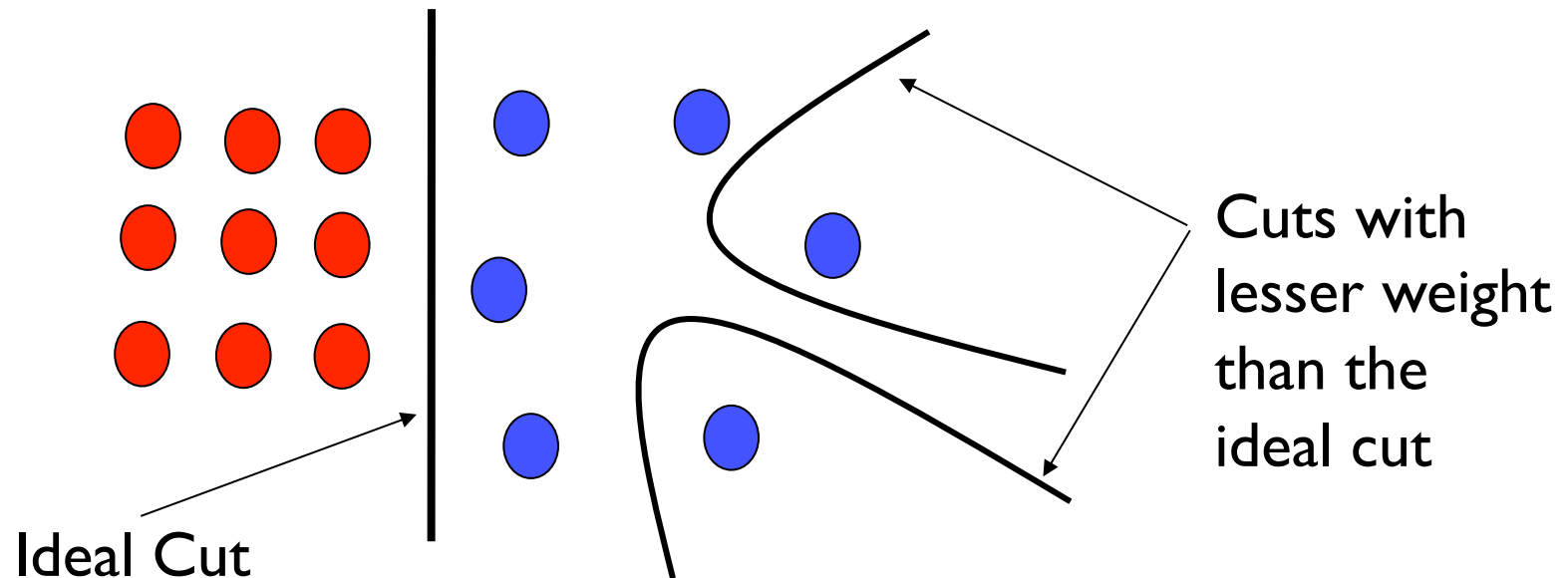
- We can do segmentation by finding the *minimum cut* in a graph
  - Efficient algorithms exist for doing this

Minimum cut example



# Drawbacks of Minimum cut

- Weight of cut is directly proportional to the number of edges in the cut.

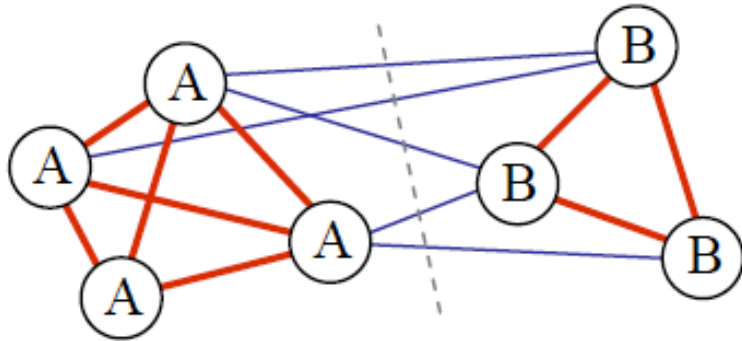


# Segmentation methods

- Segment foreground from background
- Histogram-based segmentation
- Segmentation as clustering
  - K-means clustering
  - Mean-shift segmentation
- **Graph-theoretic segmentation**
  - Min cut
  - Normalized cuts
- Interactive segmentation

# Normalized cuts

Write graph as  $V$ , one cluster as  $A$  and the other as  $B$



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

$cut(A,B)$  is sum of weights with one end in  $A$  and one end in  $B$

$$cut(A,B) = \sum_{u \in A, v \in B} W(u,v),$$

with  $A \cap B = \emptyset$

$assoc(A,V)$  is sum of all edges with one end in  $A$ .

$$assoc(A,B) = \sum_{u \in A, v \in B} W(u,v)$$

$A$  and  $B$  not necessarily disjoint



# Normalized cut

- Let  $W$  be the adjacency matrix of the graph
- Let  $D$  be the diagonal matrix with diagonal entries  $D(i, i) = \sum_j W(i, j)$
- Then the normalized cut cost can be written as

$$\frac{y^T (D - W)y}{y^T Dy}$$

where  $y$  is an indicator vector whose value should be 1 in the  $i$ th position if the  $i$ th feature point belongs to  $A$  and a negative constant otherwise

# Normalized cut

- Finding the exact minimum of the normalized cut cost is NP-complete, but if we *relax*  $y$  to take on arbitrary values, then we can minimize the relaxed cost by solving the *generalized eigenvalue problem*  $(D - W)y = \lambda Dy$
- The solution  $y$  is given by the generalized eigenvector corresponding to the second smallest eigenvalue
- Intuitively, the  $i$ th entry of  $y$  can be viewed as a “soft” indication of the component membership of the  $i$ th feature
  - Can use 0 or median value of the entries as the splitting point (threshold), or find threshold that minimizes the Ncut cost

# Normalized cut algorithm

1. Given an image or image sequence, set up a weighted graph  $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ , and set the weight on the edge connecting two nodes being a measure of the similarity between the two nodes.
2. Solve  $(\mathbf{D} - \mathbf{W})\mathbf{x} = \lambda\mathbf{D}\mathbf{x}$  for eigenvectors with the smallest eigenvalues.
3. Use the eigenvector with second smallest eigenvalue to bipartition the graph.
4. Decide if the current partition should be sub-divided, and recursively repartition the segmented parts if necessary.

# Global optimization

- In this formulation, the segmentation becomes a global process.
- Decisions about what is a boundary are not local (as in Canny edge detector)

# Boundaries of image regions defined by a number of attributes

- Brightness/color
- Texture
- Motion
- Stereoscopic depth
- Familiar configuration



[Malik]

Slide credit: B. Freeman and A. Torralba

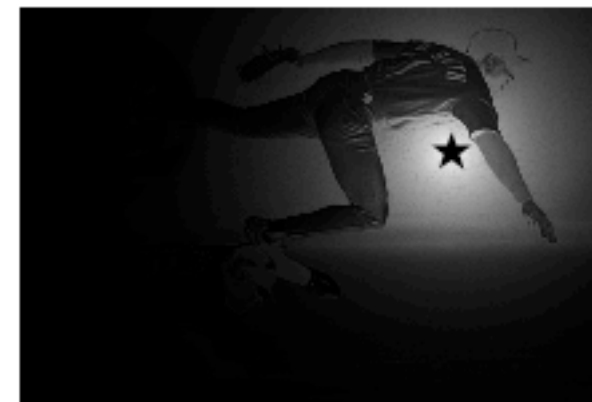
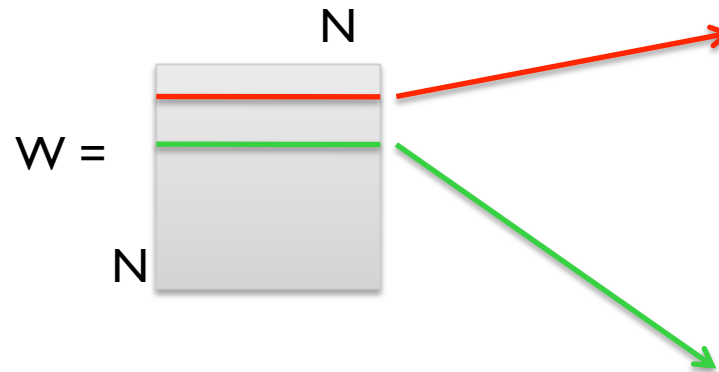
# Example

Affinity:

$$w_{ij} = \underbrace{e^{\frac{-\|F^{(i)} - F^{(j)}\|_2^2}{\sigma_I}}}_{\text{brightness}} * \underbrace{\begin{cases} e^{\frac{-\|X^{(i)} - X^{(j)}\|_2^2}{\sigma_X}} & \text{if } \|X^{(i)} - X^{(j)}\|_2 < r \\ 0 & \text{otherwise} \end{cases}}_{\text{Location}}$$



N pixels = ncols \* nrows



Slide credit: B. Freeman and A. Torralba

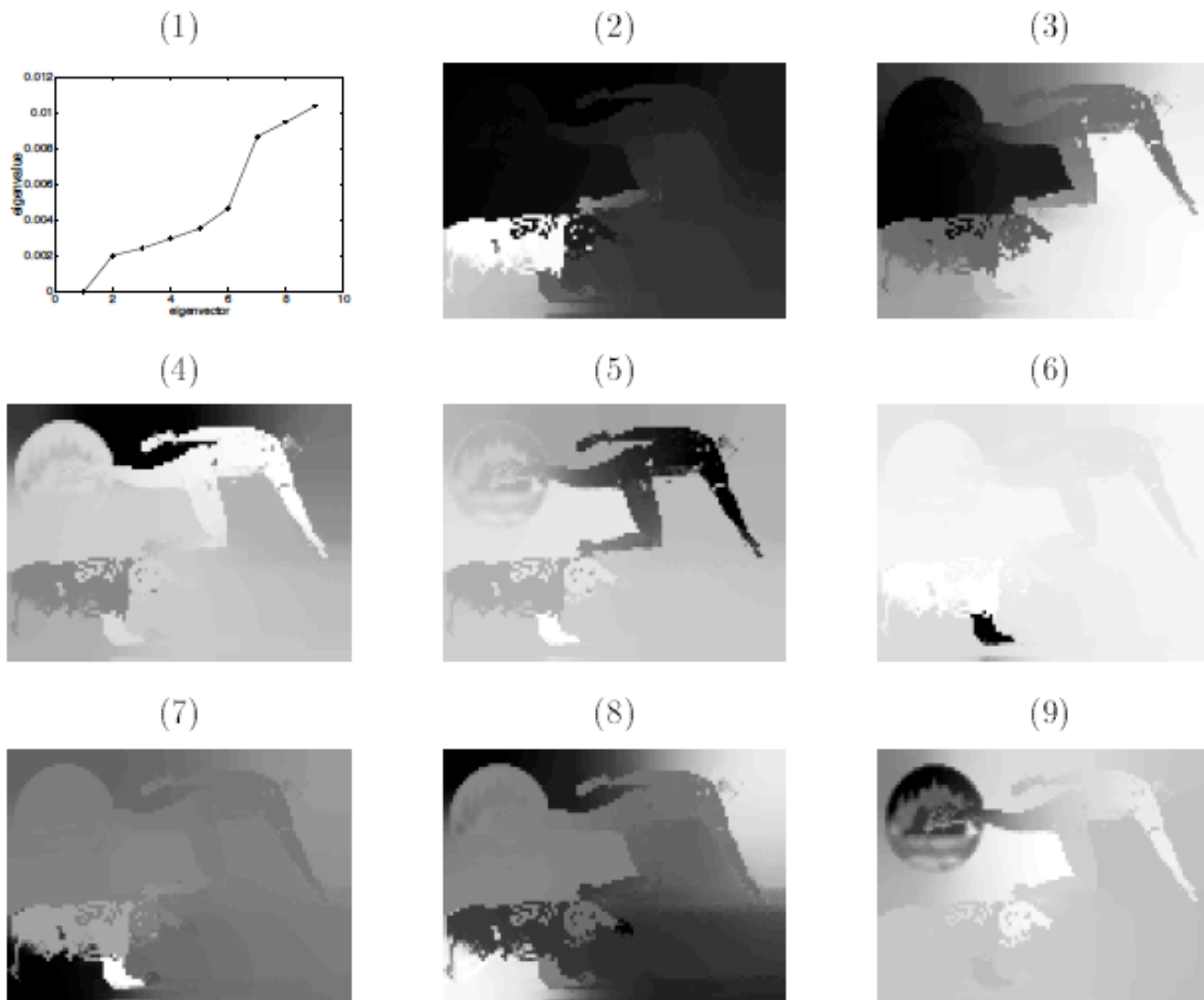


Figure 12: Subplot (1) plots the smallest eigenvectors of the generalized eigenvalue system (11). Subplot (2) - (9) shows the eigenvectors corresponding the 2nd smallest to the 9th smallest eigenvalues of the system. The eigenvectors are reshaped to be the size of the image.

# Brightness Image Segmentation

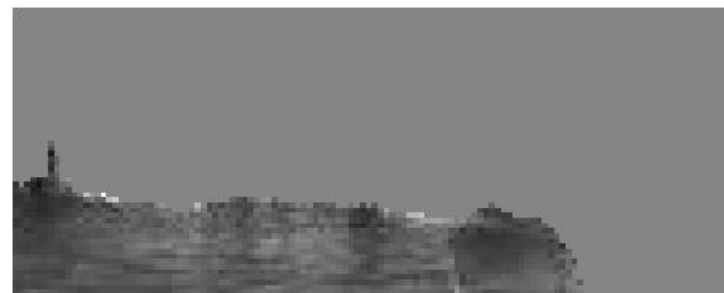


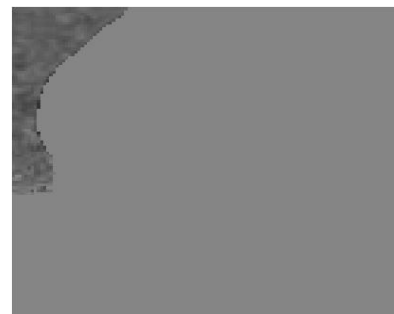
converge. On the  $100 \times 120$  test images shown here, the normalized cut algorithm takes about 2 minutes on Intel Pentium 200MHz machines.

A multiresolution implementation can be used to reduce this running time further on larger images. In our current experiments, with this implementation, the running time on a  $300 \times 400$  image can be reduced to about 20 seconds on Intel Pentium 300MHz machines. Furthermore, the bottleneck of the computation, a sparse matrix-vector



# Brightness Image Segmentation





<http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf>

Slide credit: B. Freeman and A. Torralba

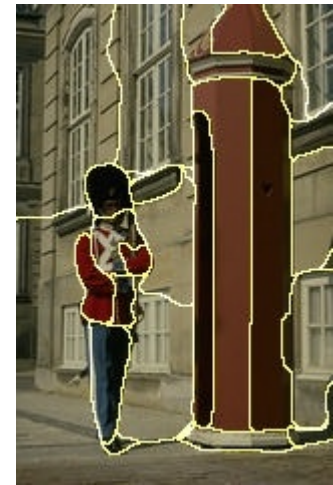
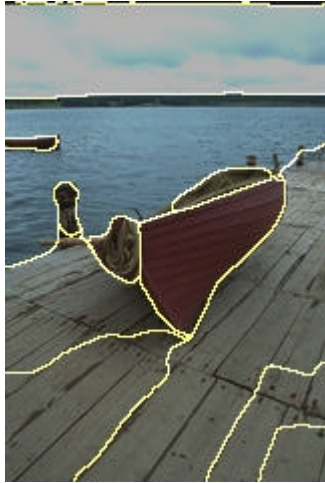
# Results on color segmentation



# Example results



# Results: Berkeley Segmentation Engine



<http://www.cs.berkeley.edu/~fowlkes/BSE/>

# Normalized cuts: Pro and con

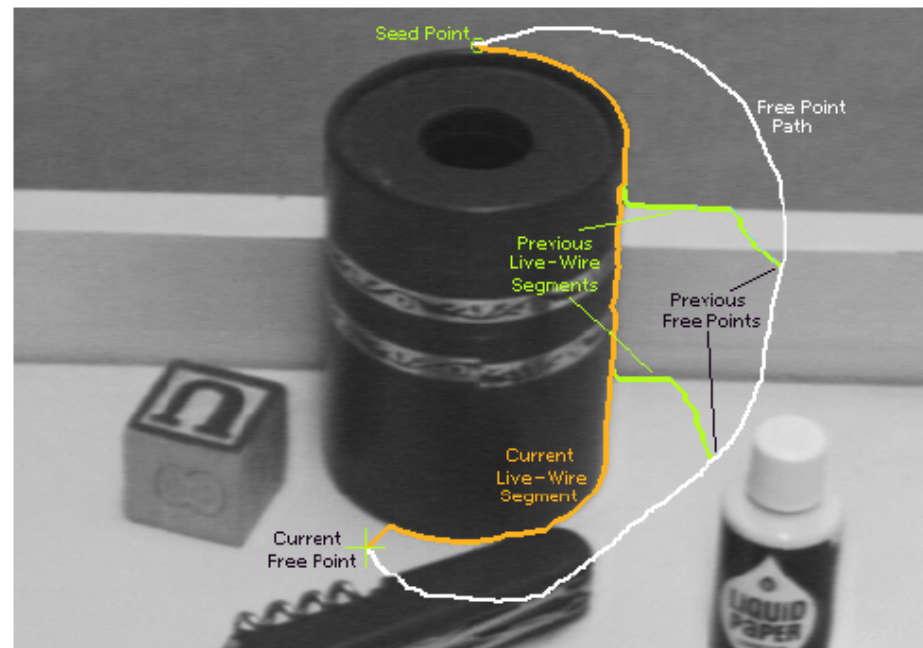
- Pros
  - Generic framework, can be used with many different features and affinity formulations
- Cons
  - High storage requirement and time complexity
  - Bias towards partitioning into equal segments

# Segmentation methods

- Segment foreground from background
- Histogram-based segmentation
- Segmentation as clustering
  - K-means clustering
  - Mean-shift segmentation
- Graph-theoretic segmentation
  - Min cut
  - Normalized cuts
- **Interactive segmentation**

# Intelligent Scissors [Mortensen 95]

- Approach answers a basic question
  - Q: how to find a path from seed to mouse that follows object boundary as closely as possible?



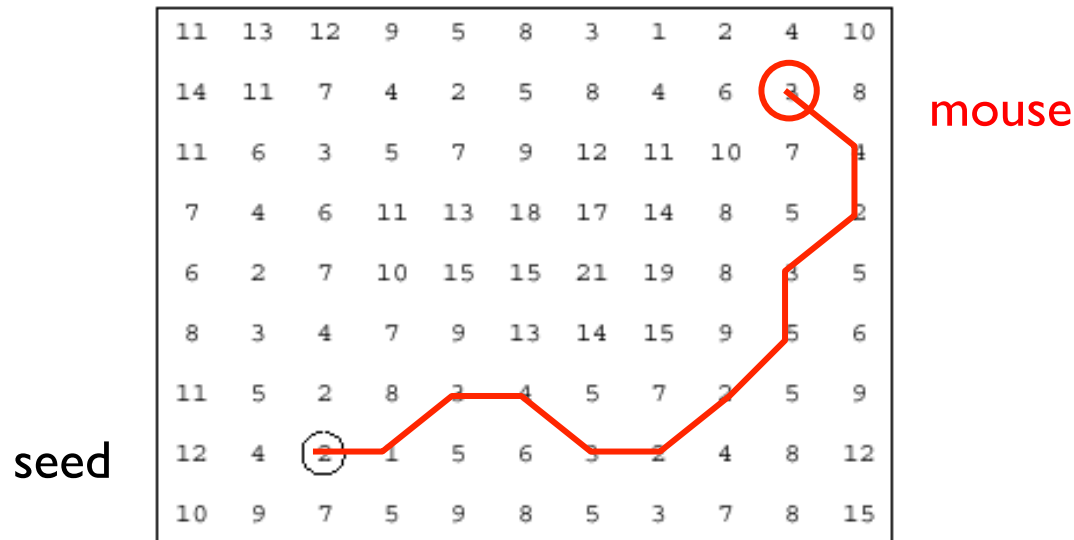
**Figure 2:** Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions ( $t_0$ ,  $t_1$ , and  $t_2$ ) are shown in green.

Mortensen and Barrett, Intelligent Scissors for Image Composition, Proc. 22nd annual conference on Computer graphics and interactive techniques, 1995



# Intelligent Scissors

- Basic Idea
  - Define edge score for each pixel
    - edge pixels have low cost
  - Find lowest cost path from seed to mouse



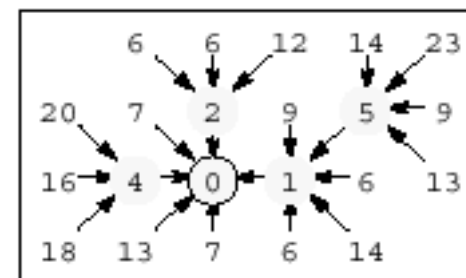
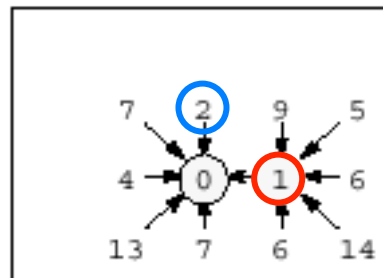
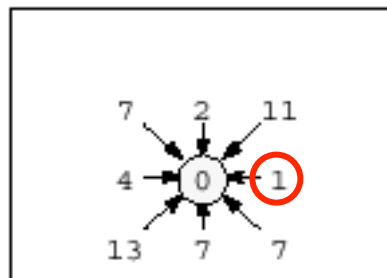
## Questions

- How to define costs?
- How to find the path?

# Path Search (basic idea)

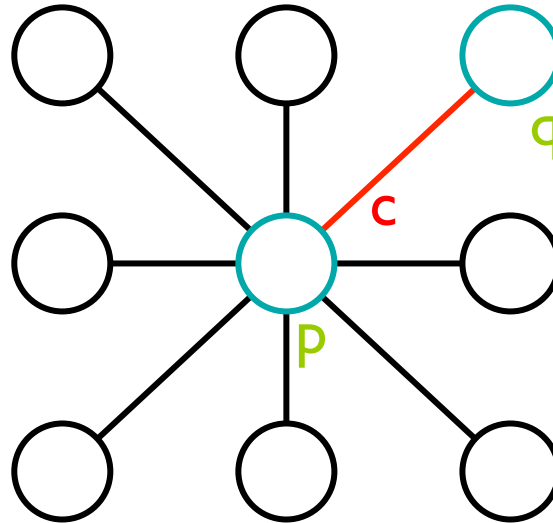
- Graph Search Algorithm
  - Computes minimum cost path from seed to *all other pixels*

11	13	12	9	5	8	3	1	2	4	10
14	11	7	4	2	5	8	4	6	3	8
11	6	3	5	7	9	12	11	10	7	4
7	4	6	11	13	18	17	14	8	5	2
6	2	7	10	15	15	21	19	8	3	5
8	3	4	7	9	13	14	15	9	5	6
11	5	2	8	3	4	5	7	2	5	9
12	4	2	1	5	6	3	2	4	8	12
10	9	7	5	9	8	5	3	7	8	15



# How does this really work?

- Treat the image as a graph



## Graph

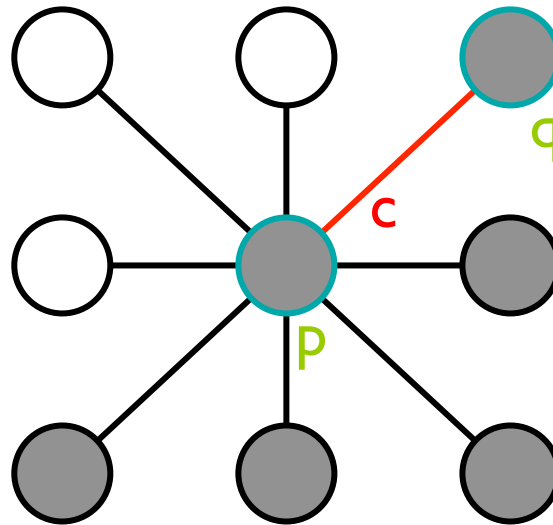
- node for every pixel **p**
- link between every adjacent pair of pixels, **p,q**
- cost **c** for each link

Note: each *link* has a cost

- this is a little different than the figure before where each pixel had a cost

# Defining the costs

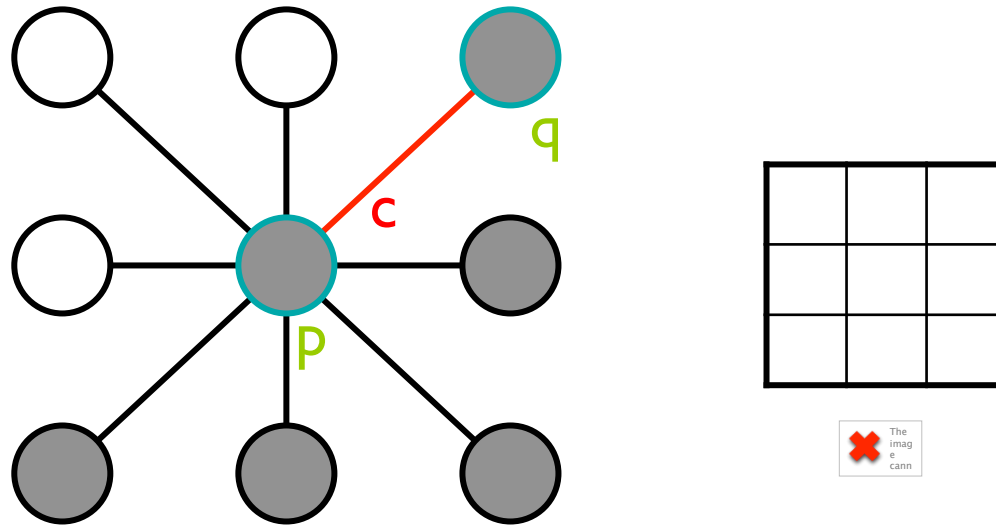
- Treat the image as a graph



Want to hug image edges: how to define cost of a link?

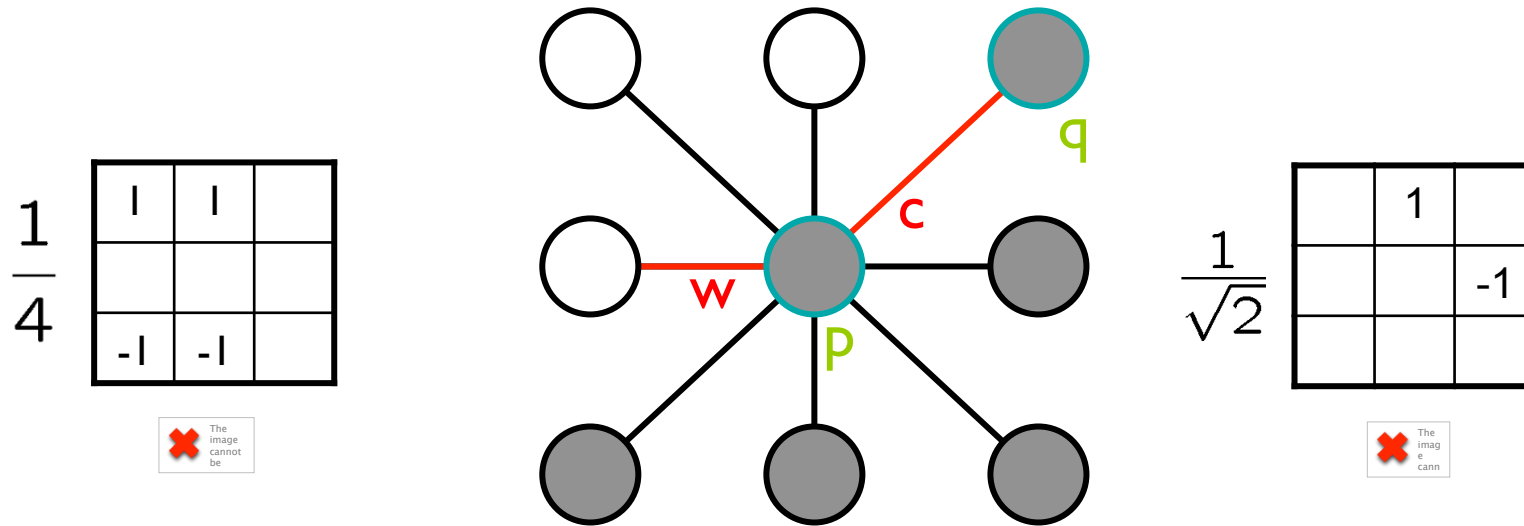
- the link should follow the intensity edge
  - want intensity to change rapidly  $\perp$  to the link
- $c \approx - |\text{difference of intensity } \perp \text{ to link}|$

# Defining the costs



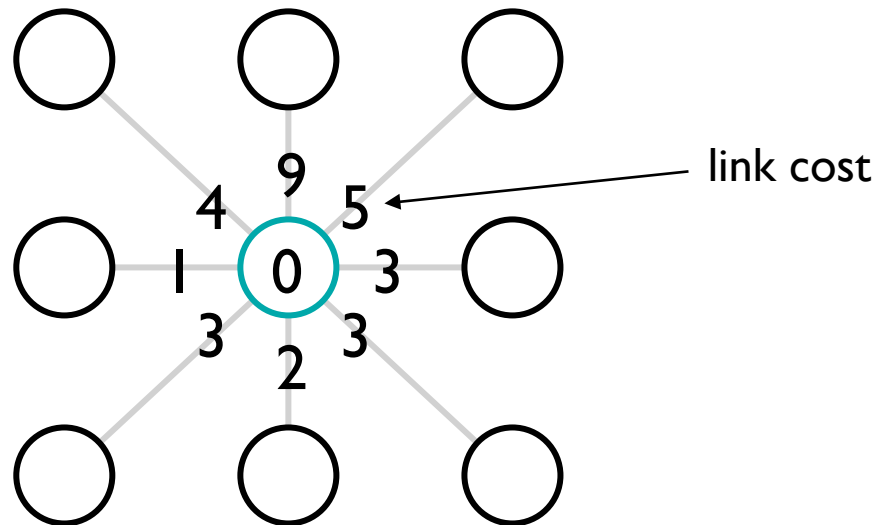
- **c** can be computed using a cross-correlation filter
  - assume it is centered at **p**
- Also typically scale **c** by its length
  - set  $c = (\max - |\text{filter response}|)$ 
    - where  $\max = \text{maximum } |\text{filter response}|$  over all pixels in the image

# Defining the costs



- $c$  can be computed using a cross-correlation filter
  - assume it is centered at  $p$
- Also typically scale  $c$  by its length
  - set  $c = (\max\text{-}|\text{filter response}|)$ 
    - where max = maximum  $|\text{filter response}|$  over all pixels in the image

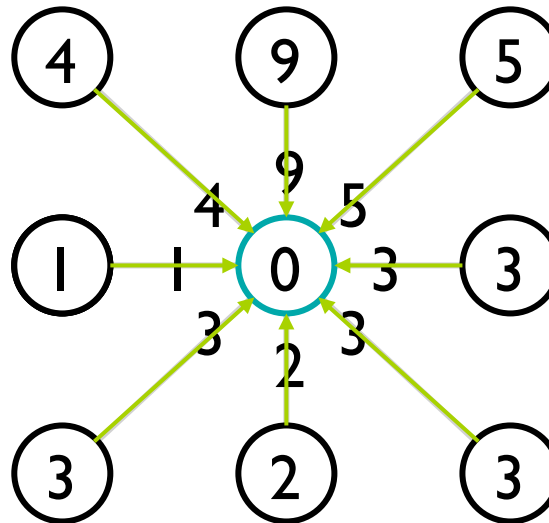
# Dijkstra's shortest path algorithm



## Algorithm

1. init node costs to  $\infty$ , set  $p$  = seed point,  $\text{cost}(p) = 0$
2. expand  $p$  as follows:
  - for each of  $p$ 's neighbors  $q$  that are not expanded
    - » set  $\text{cost}(q) = \min(\text{cost}(p) + c_{pq}, \text{cost}(q))$

# Dijkstra's shortest path algorithm

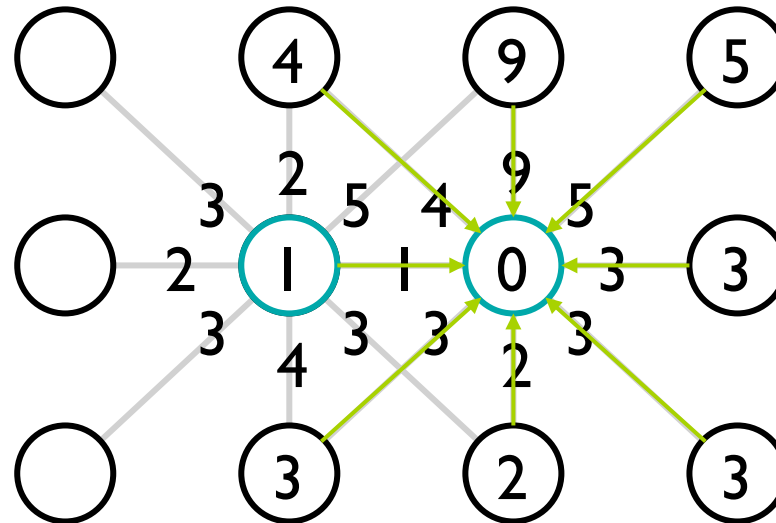


## Algorithm

1. init node costs to  $\infty$ , set  $p$  = seed point,  $\text{cost}(p) = 0$
2. expand  $p$  as follows:
  - for each of  $p$ 's neighbors  $q$  that are not expanded
    - » set  $\text{cost}(q) = \min(\text{cost}(p) + c_{pq}, \text{cost}(q))$ 
      - » if  $q$ 's cost changed, make  $q$  point back to  $p$
    - » put  $q$  on the ACTIVE list (if not already there)



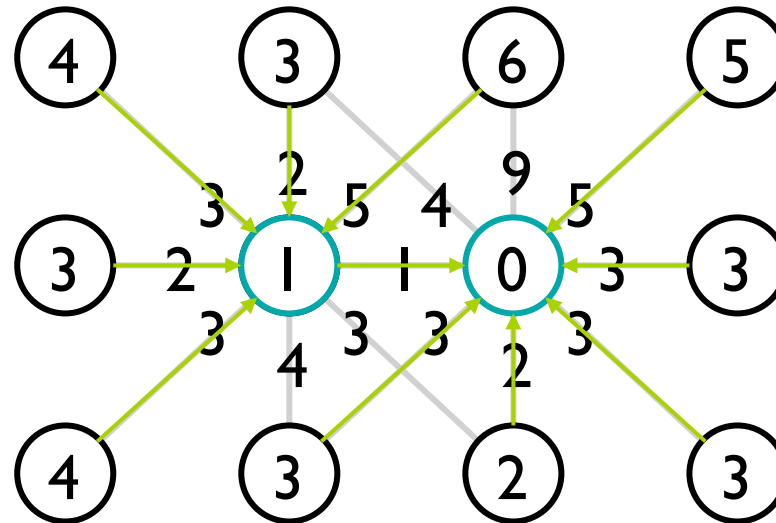
# Dijkstra's shortest path algorithm



## Algorithm

1. init node costs to  $\infty$ , set  $p$  = seed point,  $\text{cost}(p) = 0$
2. expand  $p$  as follows:
  - for each of  $p$ 's neighbors  $q$  that are not expanded
    - » set  $\text{cost}(q) = \min(\text{cost}(p) + c_{pq}, \text{cost}(q))$ 
      - » if  $q$ 's cost changed, make  $q$  point back to  $p$
    - » put  $q$  on the ACTIVE list (if not already there)
3. set  $r$  = node with minimum cost on the ACTIVE list
4. repeat Step 2 for  $p = r$

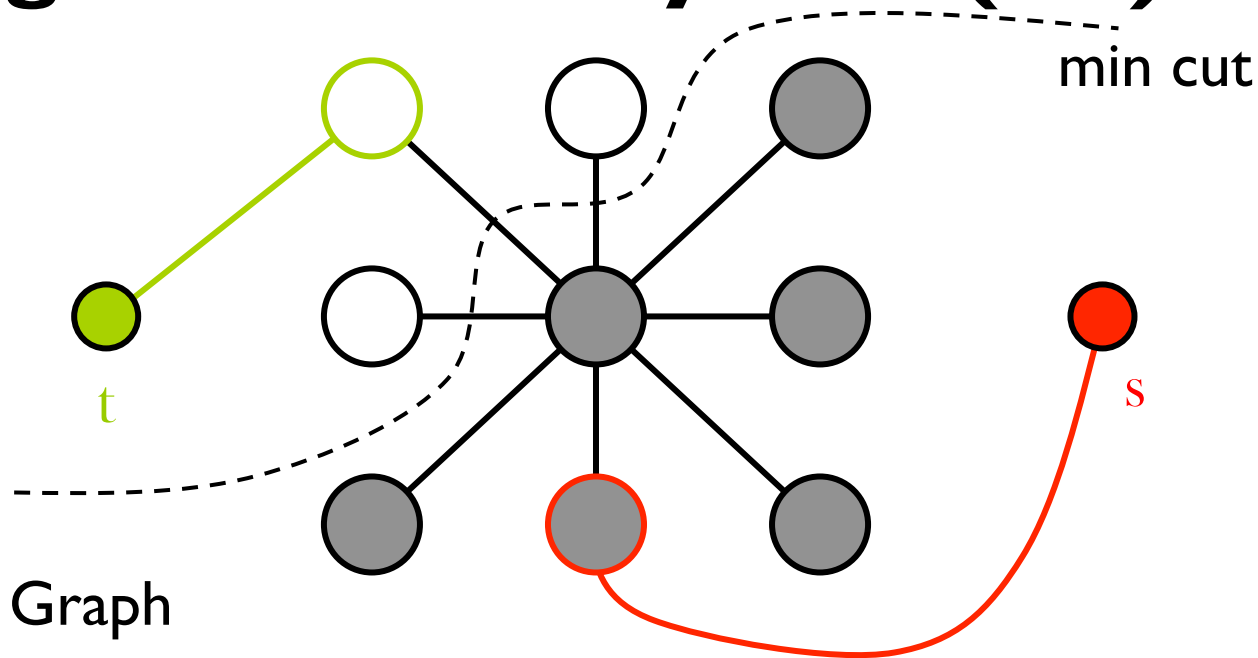
# Dijkstra's shortest path algorithm



## Algorithm

1. init node costs to  $\infty$ , set  $p$  = seed point,  $\text{cost}(p) = 0$
2. expand  $p$  as follows:
  - for each of  $p$ 's neighbors  $q$  that are not expanded
    - » set  $\text{cost}(q) = \min(\text{cost}(p) + c_{pq}, \text{cost}(q))$ 
      - » if  $q$ 's cost changed, make  $q$  point back to  $p$
    - » put  $q$  on the ACTIVE list (if not already there)
3. set  $r$  = node with minimum cost on the ACTIVE list
4. repeat Step 2 for  $p = r$

# Segmentation by min (s-t) cut



- Graph

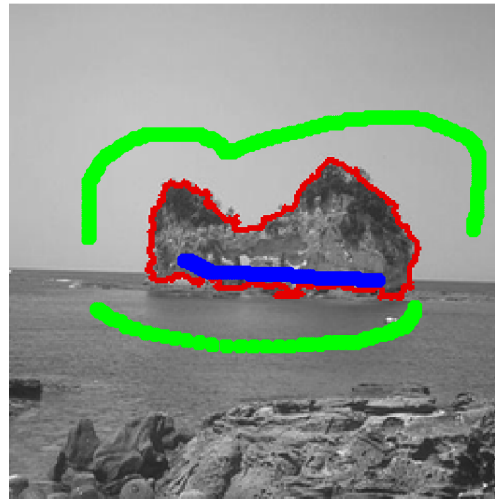
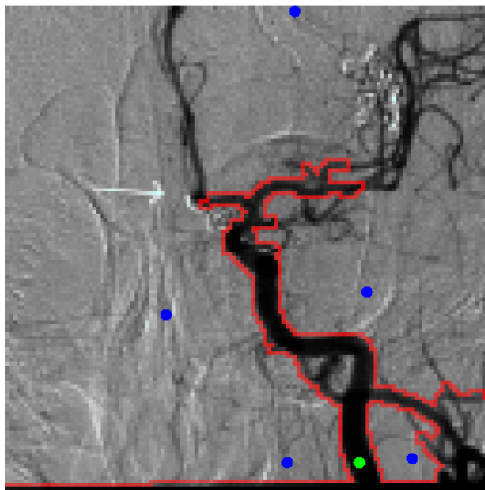
- node for each pixel, link between pixels
- specify a few pixels as foreground and background
  - create an infinite cost link from each bg pixel to the “t” node
  - create an infinite cost link from each fg pixel to the “s” node
- compute min cut that separates s from t
- how to define link cost between neighboring pixels?

Y. Boykov and M-P Jolly, Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D images, ICCV, 2001.

Slide credit: S. Seitz

# Random Walker

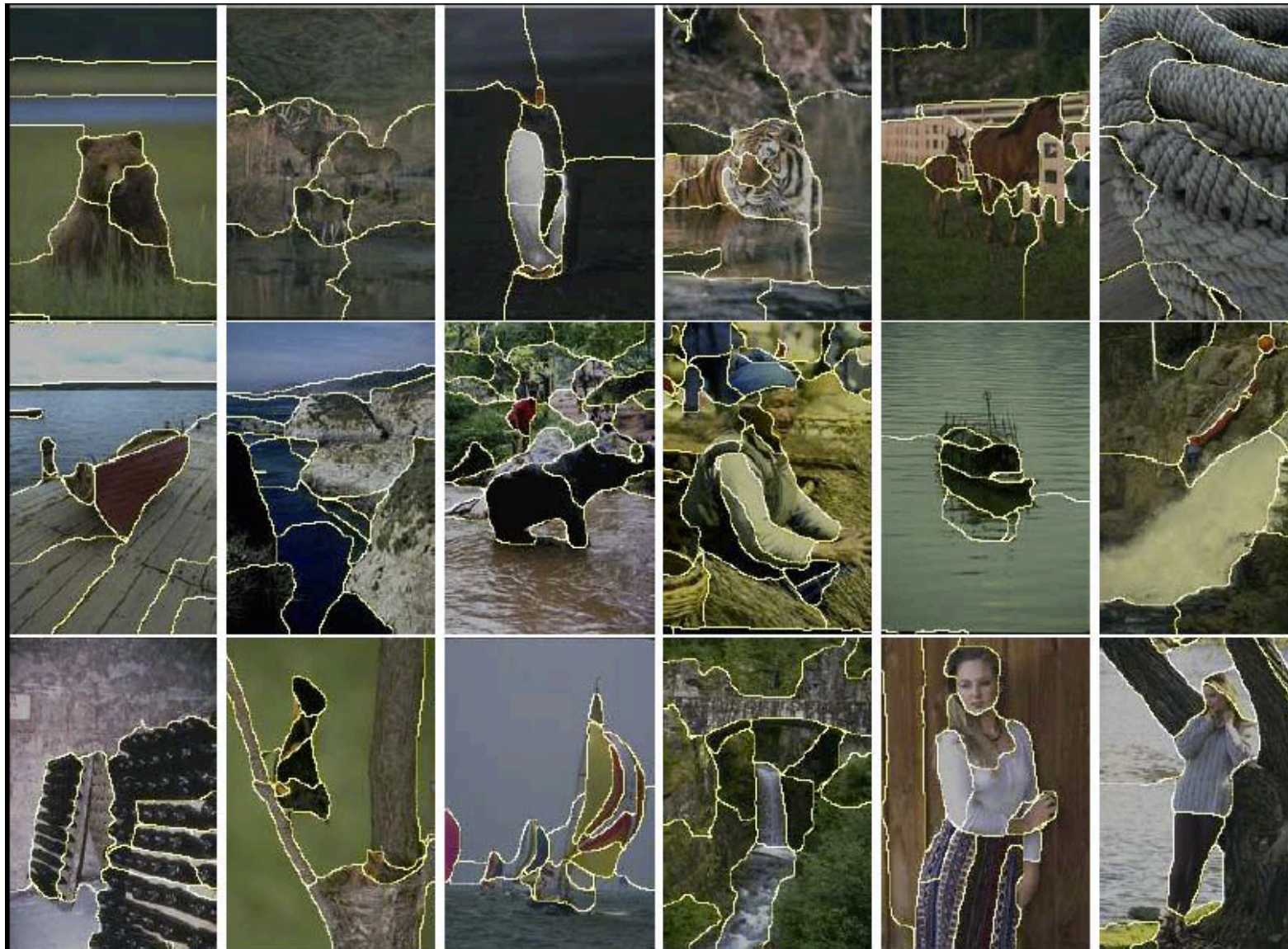
- Compute probability that a random walker arrives at seed



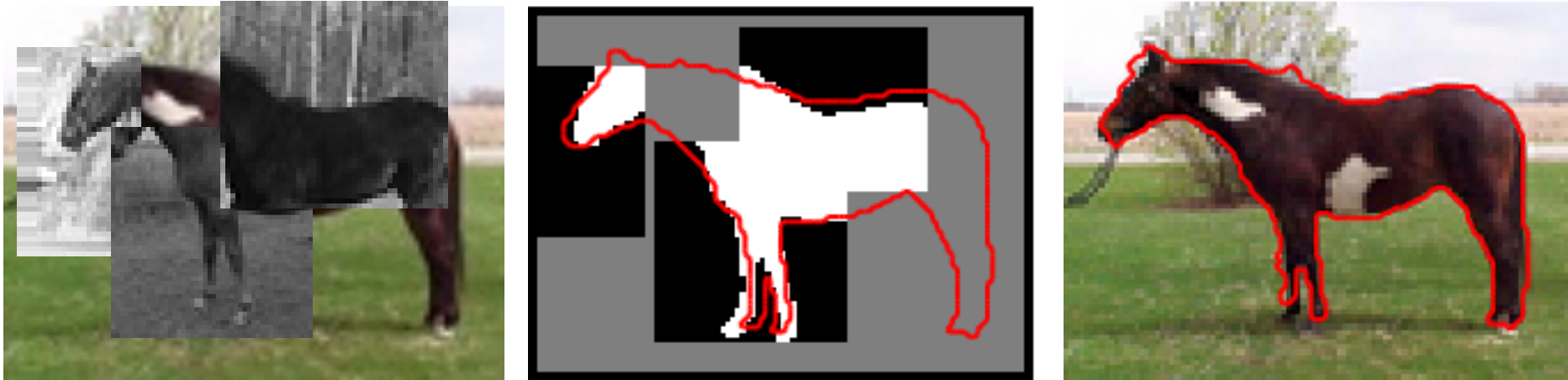
L. Grady, [Random Walks for Image Segmentation](http://cns.bu.edu/~lgrady/Random_Walker_Image_Segmentation.html), IEEE T-PAMI, 2006

[http://cns.bu.edu/~lgrady/Random\\_Walker\\_Image\\_Segmentation.html](http://cns.bu.edu/~lgrady/Random_Walker_Image_Segmentation.html)

# Do we need recognition to take the next step in performance?

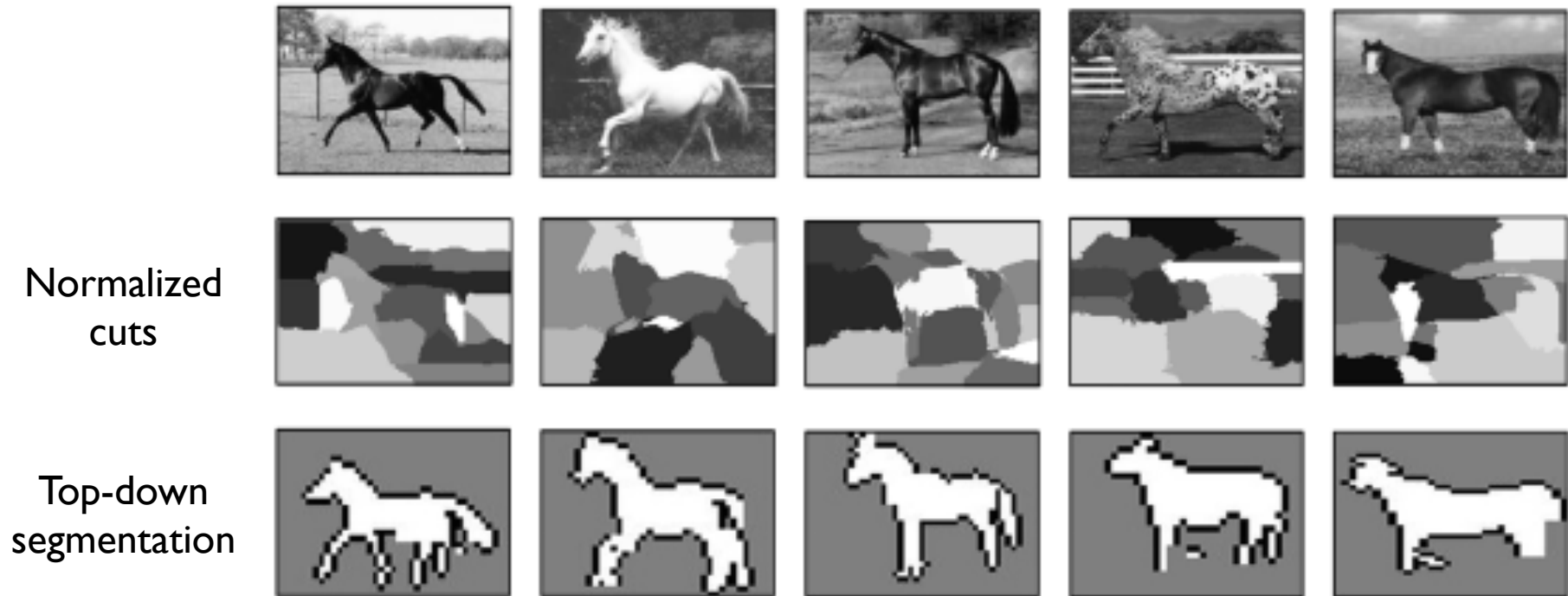


# Top-down segmentation



- E. Borenstein and S. Ullman, [Class-specific, top-down segmentation](#), ECCV 2002
- A. Levin and Y. Weiss, [Learning to Combine Bottom-Up and Top-Down Segmentation](#), ECCV 2006.

# Top-down segmentation



- E. Borenstein and S. Ullman, [Class-specific, top-down segmentation](#), ECCV 2002
- A. Levin and Y. Weiss, [Learning to Combine Bottom-Up and Top-Down Segmentation](#), ECCV 2006.

# Motion segmentation



Input sequence

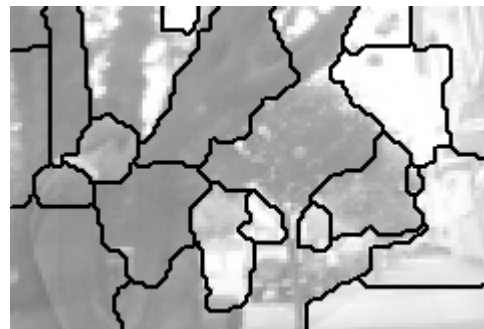
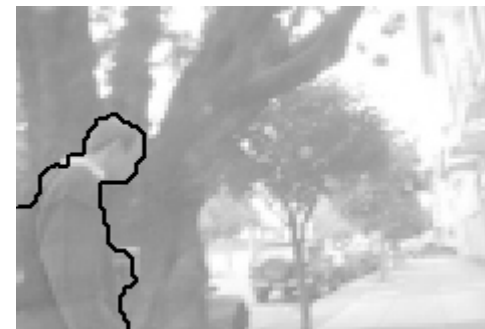


Image Segmentation



Motion Segmentation



Input sequence

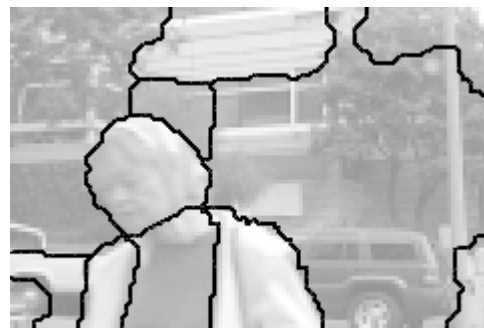


Image Segmentation



Motion Segmentation

A. Barbu, S.C. Zhu. [Generalizing Swendsen-Wang to sampling arbitrary posterior probabilities](#), IEEE TPAMI, 2005.

Slide credit: K. Grauman