Image segmentation

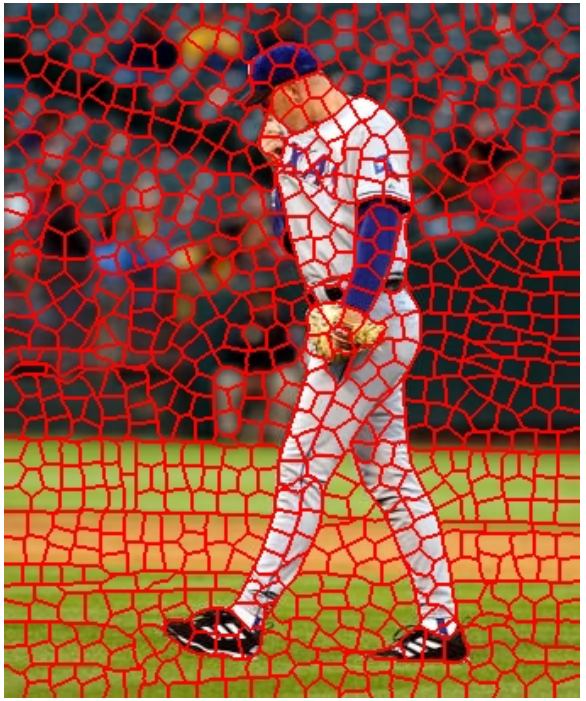
The goals of segmentation

- Obtain primitives for other tasks
  - e.g., feature support
- Perceptual organization, recognition
- Graphics, image manipulation

The goals of segmentation

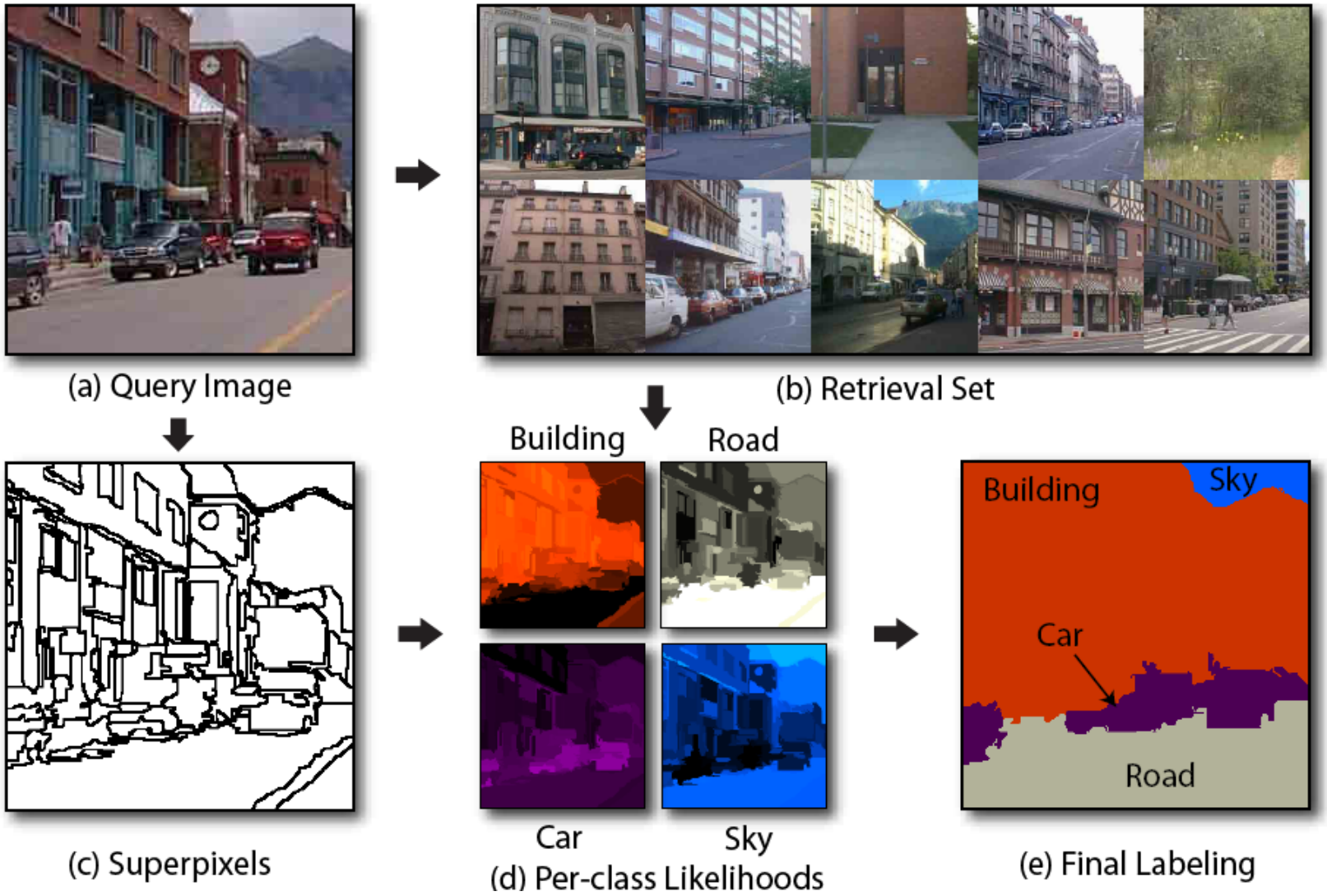
- Group together similar-looking pixels for efficiency of further processing
  - "Bottom-up" process
  - Unsupervised

"superpixels"



X. Ren and J. Malik. **Learning a classification model for segmentation.** ICCV 2003.

Segments as primitives for recognition

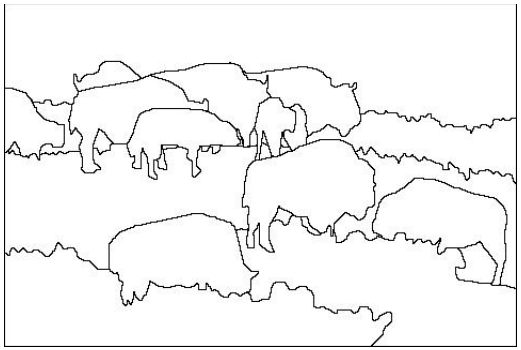


J. Tighe and S. Lazebnik, ECCV 2010

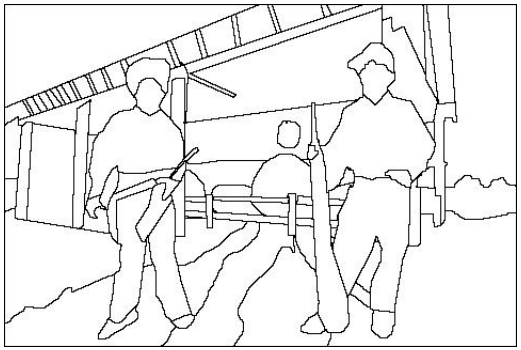
The goals of segmentation

- Separate image into coherent “objects”
  - “Bottom-up” or “top-down” process?
  - Supervised or unsupervised?

image



human segmentation



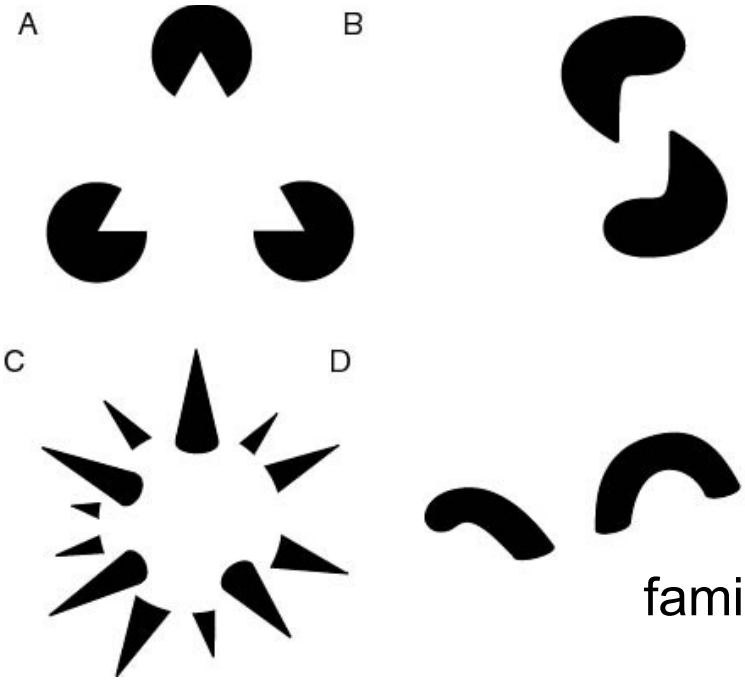
Berkeley segmentation database:

http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Inspiration from psychology

- The Gestalt school: Grouping is key to visual perception
  - "The whole is greater than the sum of its parts"

subjective contours



occlusion

familiar configuration

http://en.wikipedia.org/wiki/Gestalt_psychology

Emergence

http://en.wikipedia.org/wiki/Gestalt_psychology

The goals of segmentation

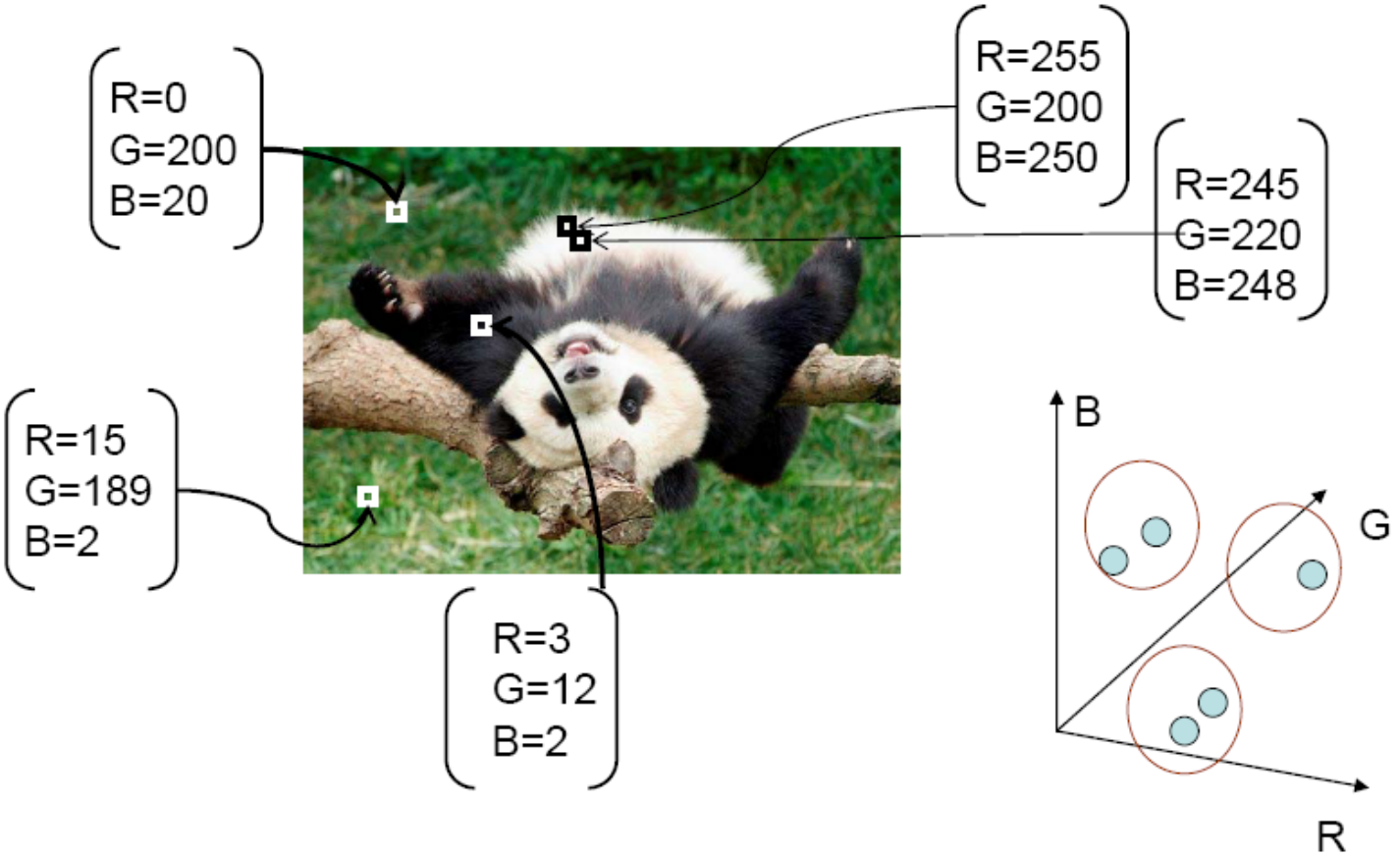
- Interactive segmentation for graphics

Approaches to segmentation

- Segmentation as clustering
- Segmentation as graph partitioning
- Segmentation as labeling

Segmentation as clustering

- Cluster similar pixels (features) together



Source: K. Grauman

Segmentation as clustering

- K-means clustering based on intensity or color is essentially vector quantization of the image attributes
  - Clusters don't have to be spatially coherent

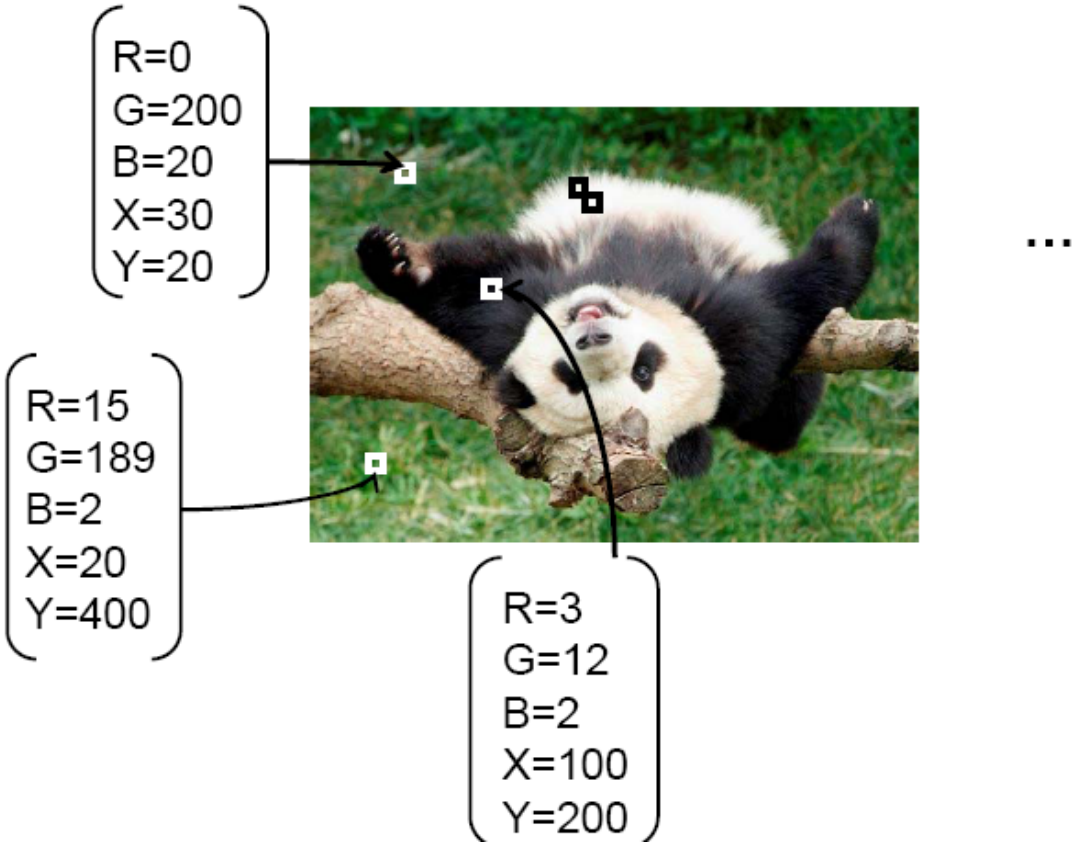
Image

Intensity-based clusters

Color-based clusters

Segmentation as clustering

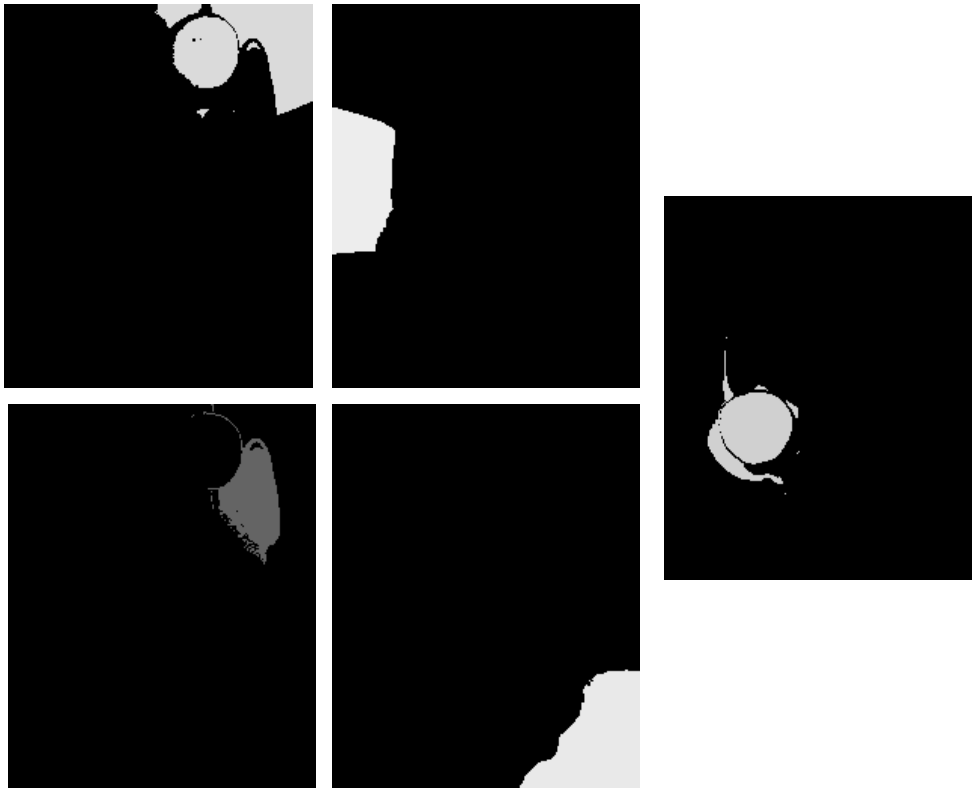
- Cluster similar pixels (features) together



Source: K. Grauman

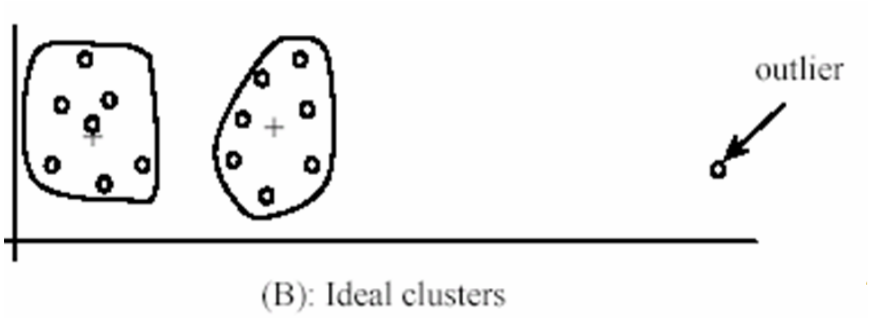
Segmentation as clustering

- Clustering based on (r,g,b,x,y) values enforces more spatial coherence

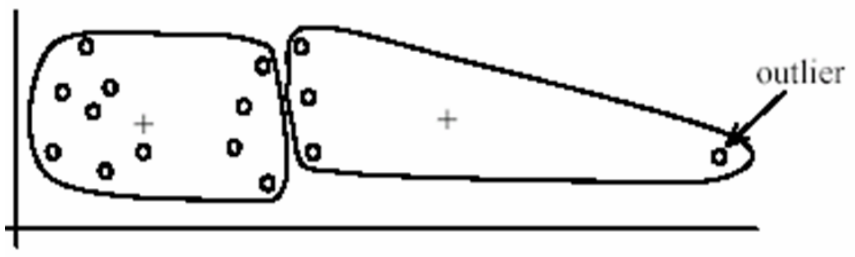


K-Means pros and cons

- Pros
  - Simple and fast
  - Easy to implement
- Cons
  - Need to choose K
  - Sensitive to outliers
- Usage
  - Rarely used for pixel segmentation



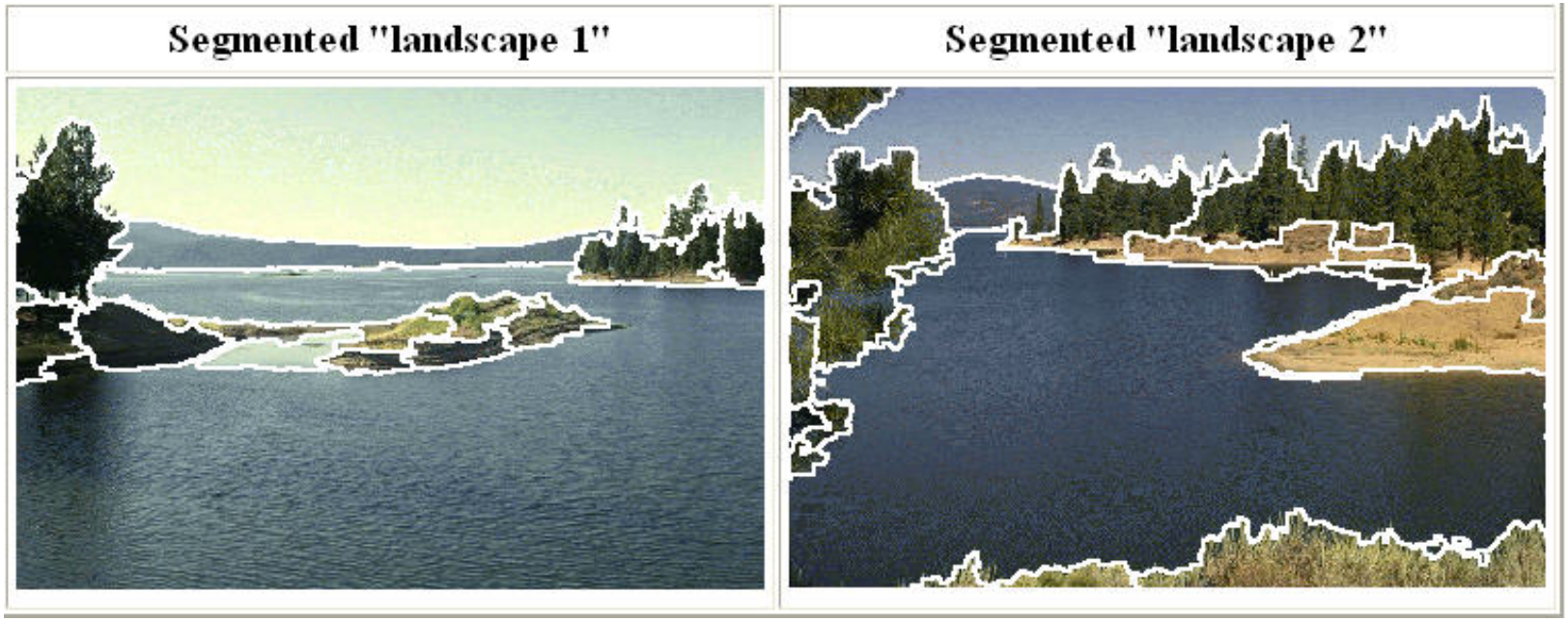
(B): Ideal clusters



Mean shift segmentation

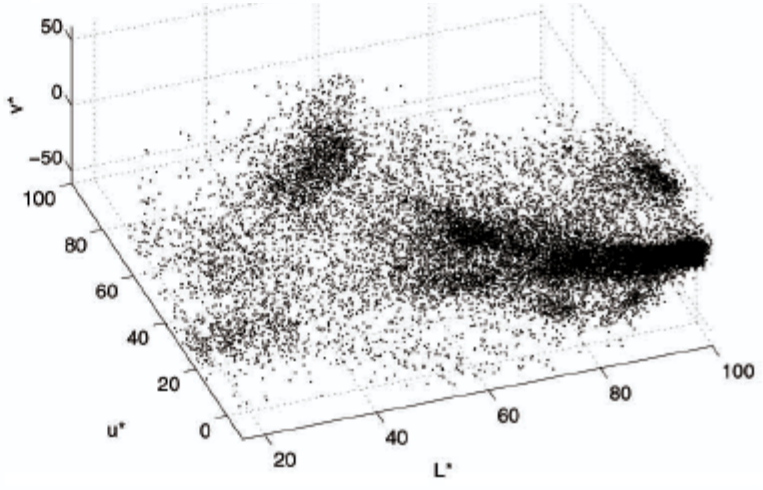
D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

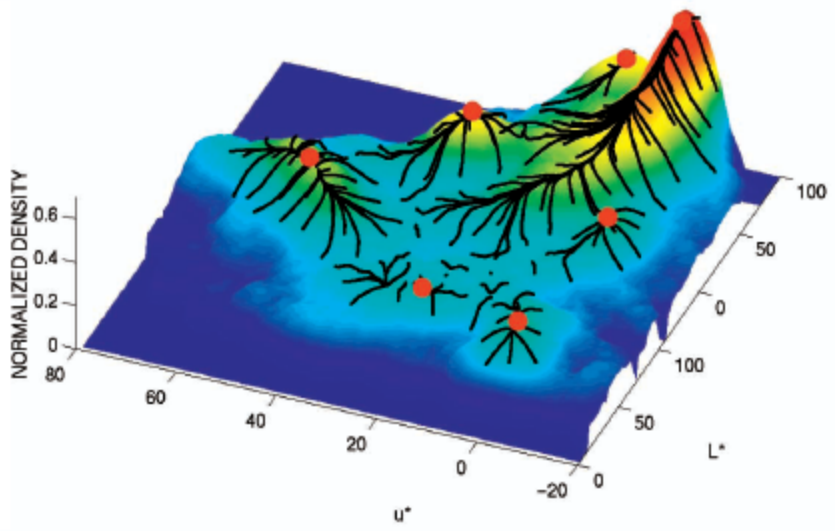
Versatile technique for clustering-based segmentation

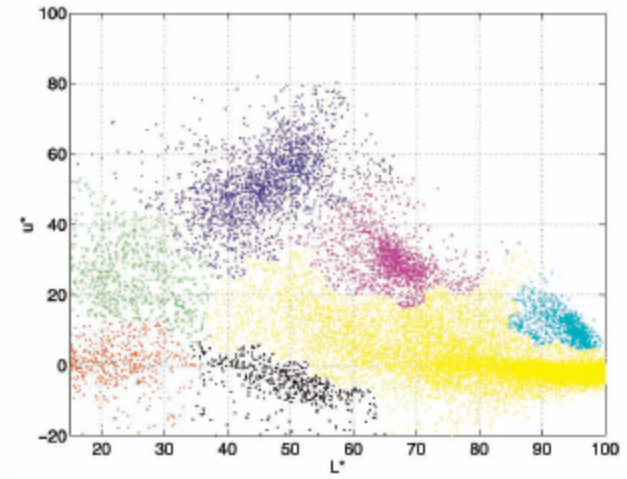


Mean shift algorithm

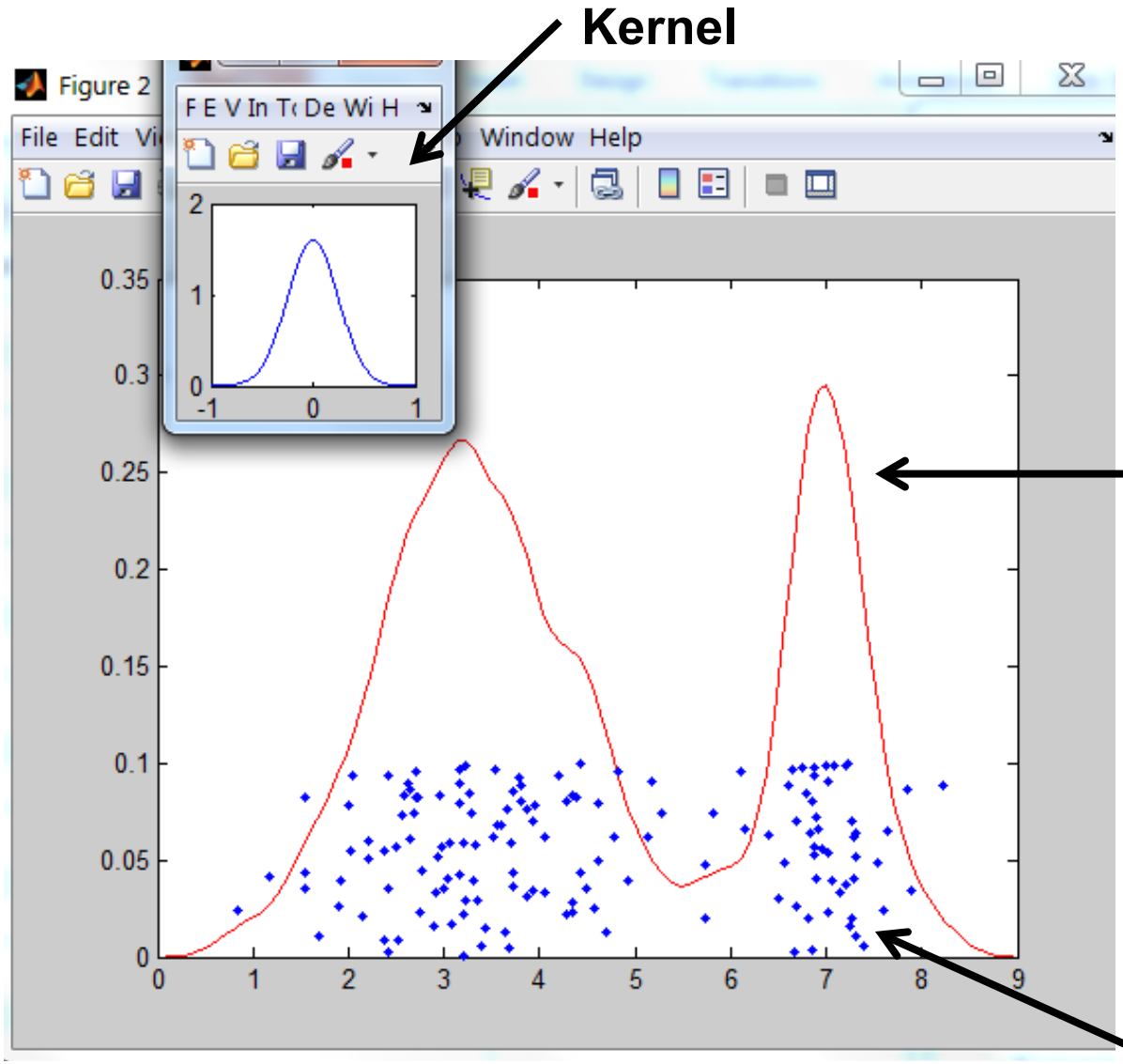
Try to find *modes* of this non-parametric density







Kernel density estimation



Kernel density estimation

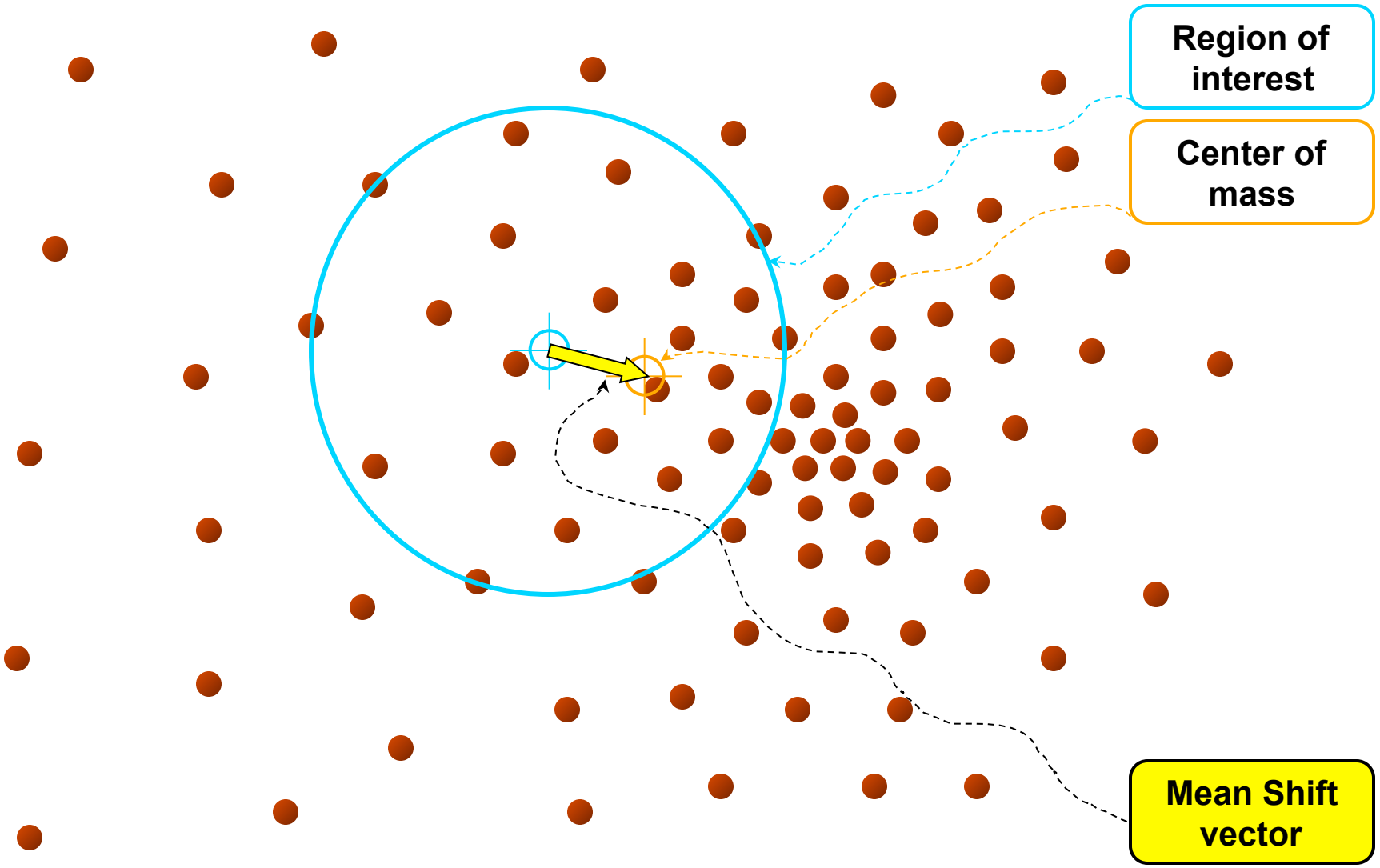
Kernel density estimation function

$$\widehat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

Gaussian kernel

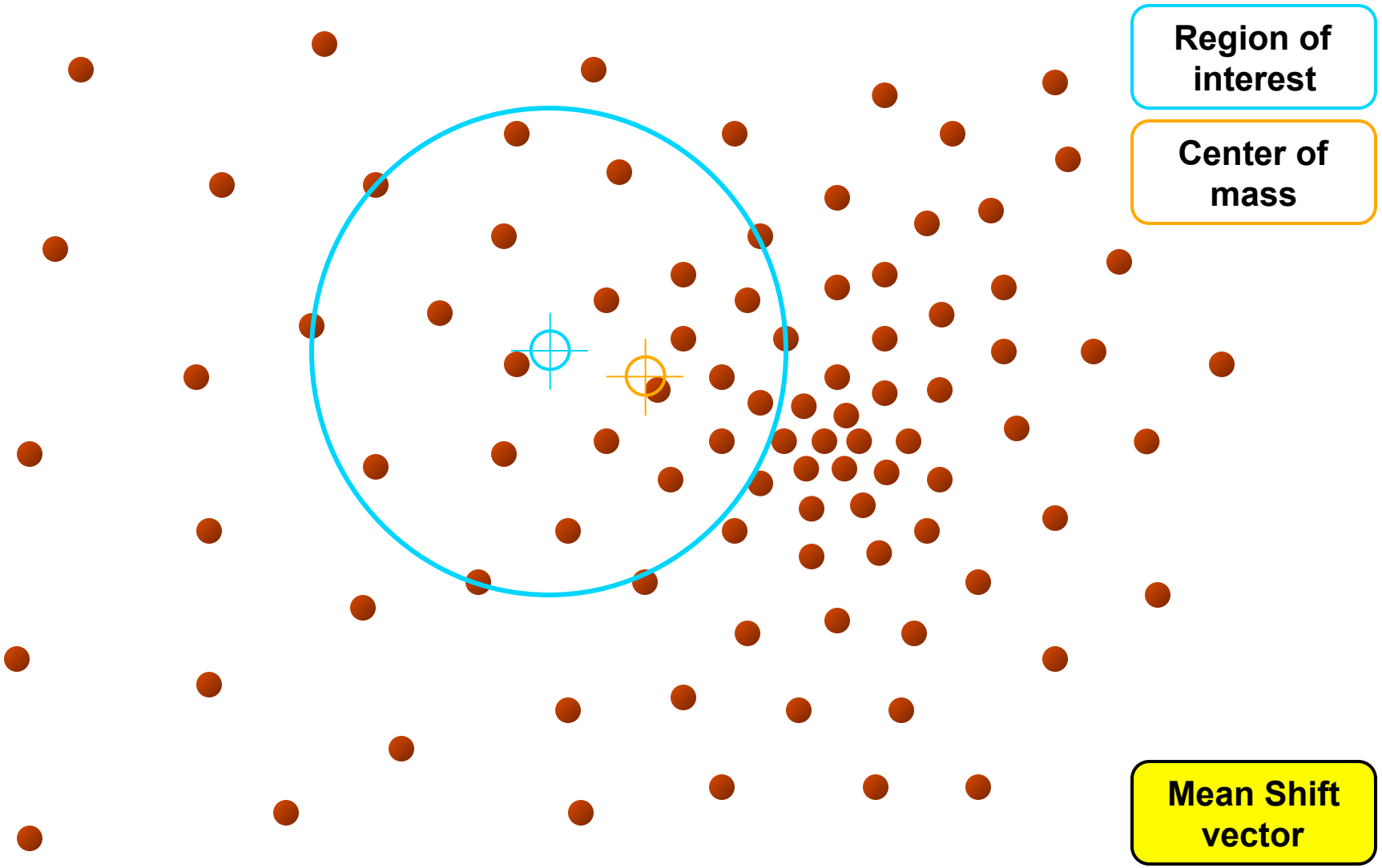
$$K\left(\frac{x - x_i}{h}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-x_i)^2}{2h^2}}.$$

Mean shift



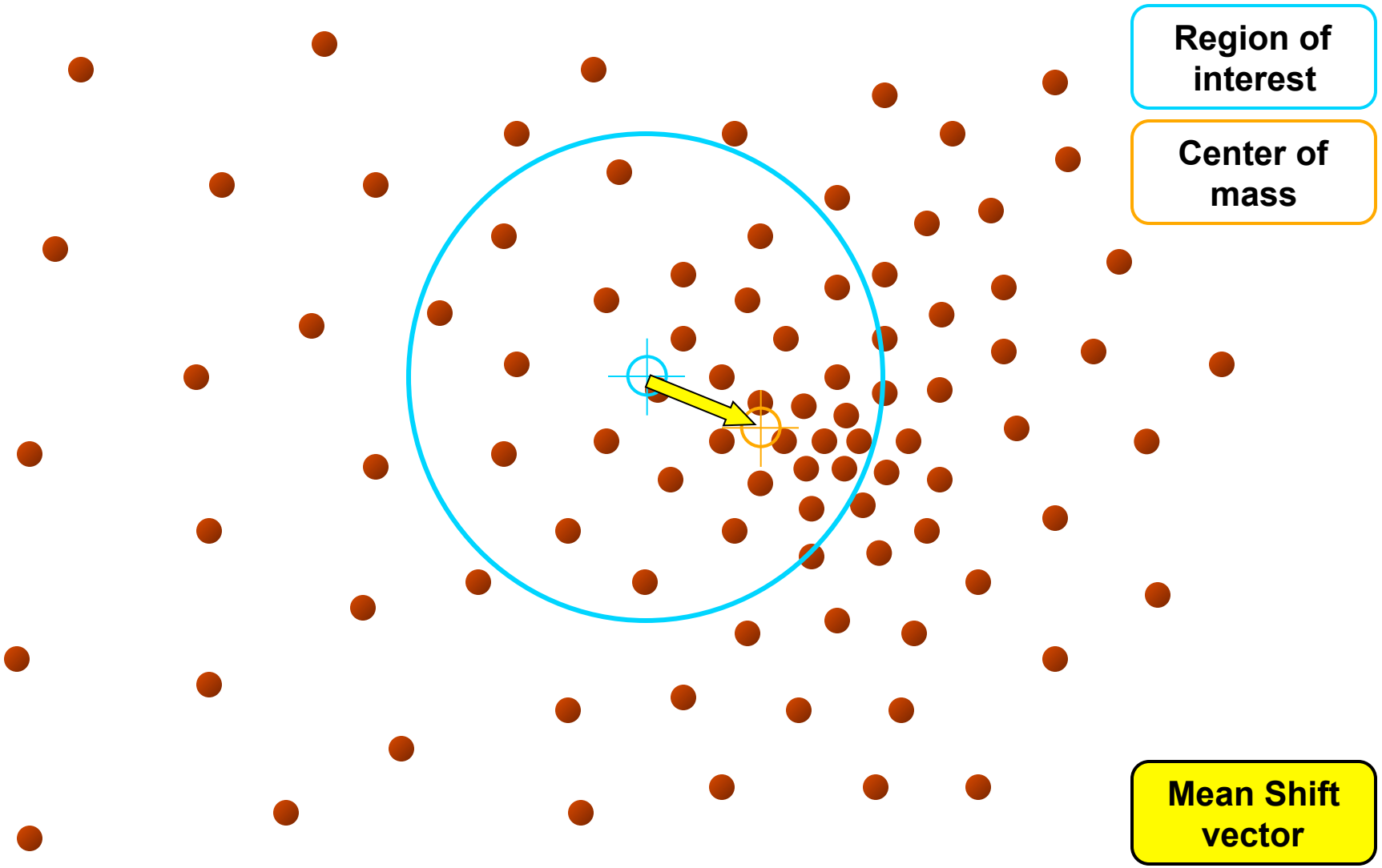
Slide by Y. Ukrainitz & B. Sarel

Mean shift



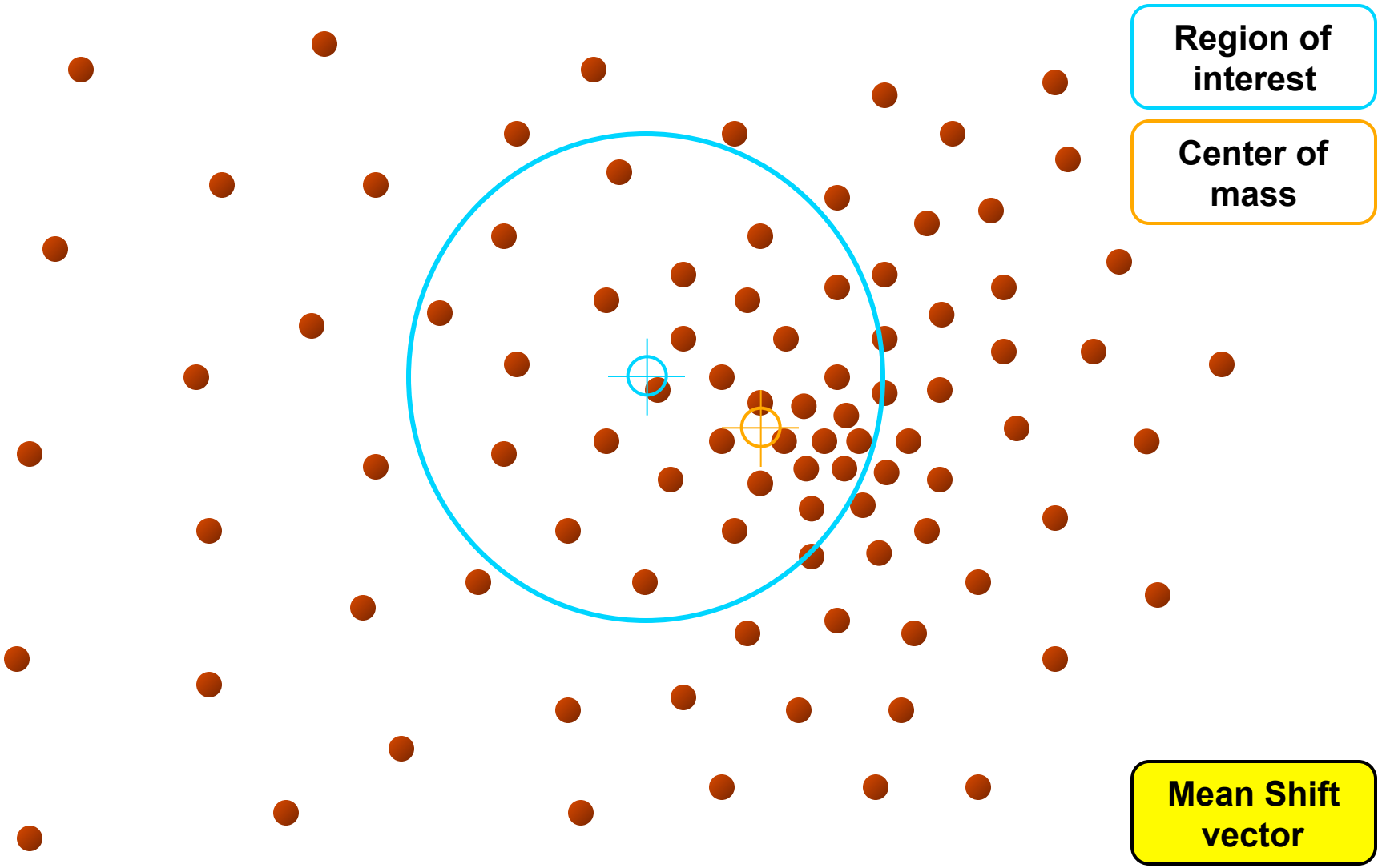
Slide by Y. Ukrainitz & B. Sarel

Mean shift



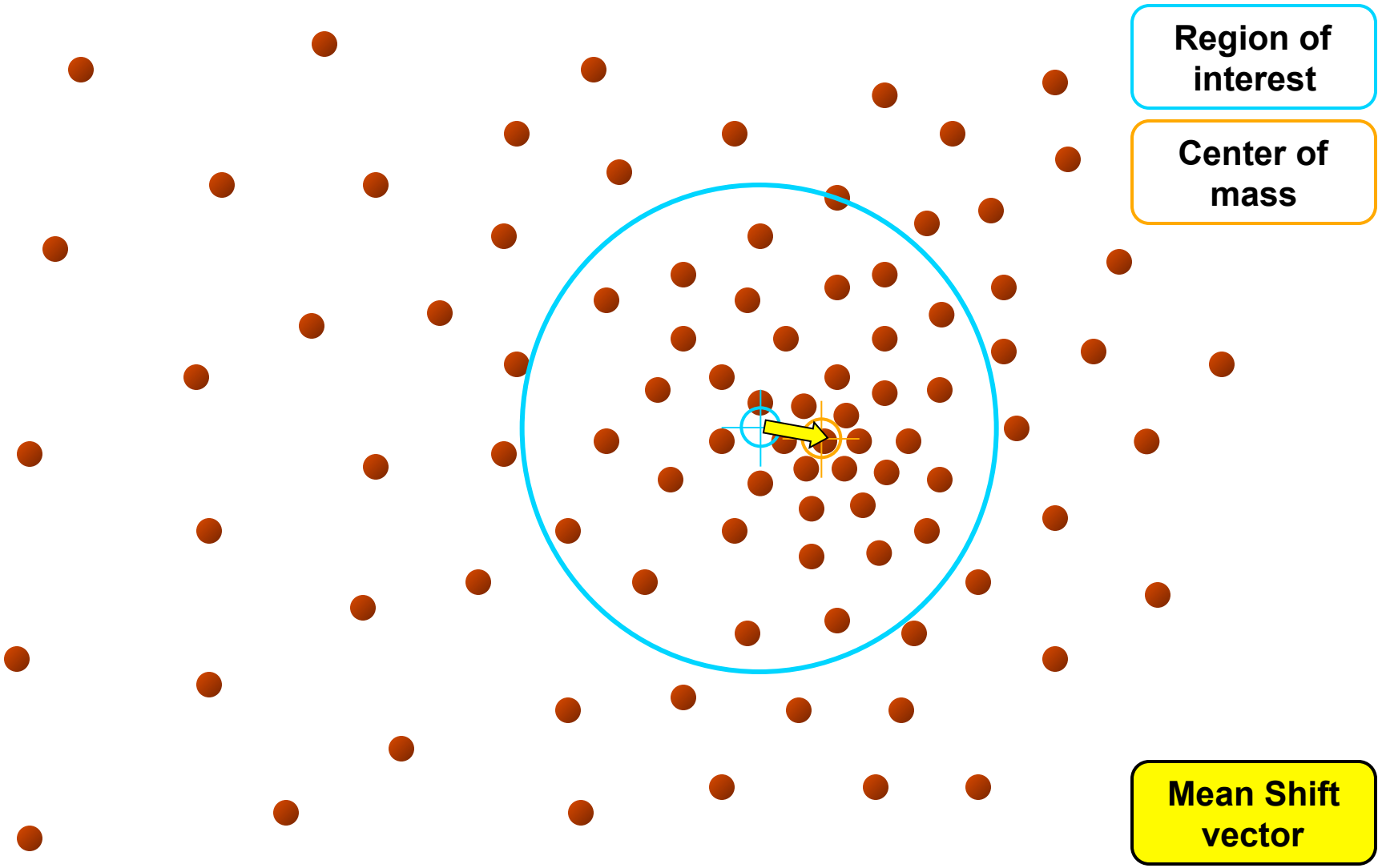
Slide by Y. Ukrainitz & B. Sarel

Mean shift



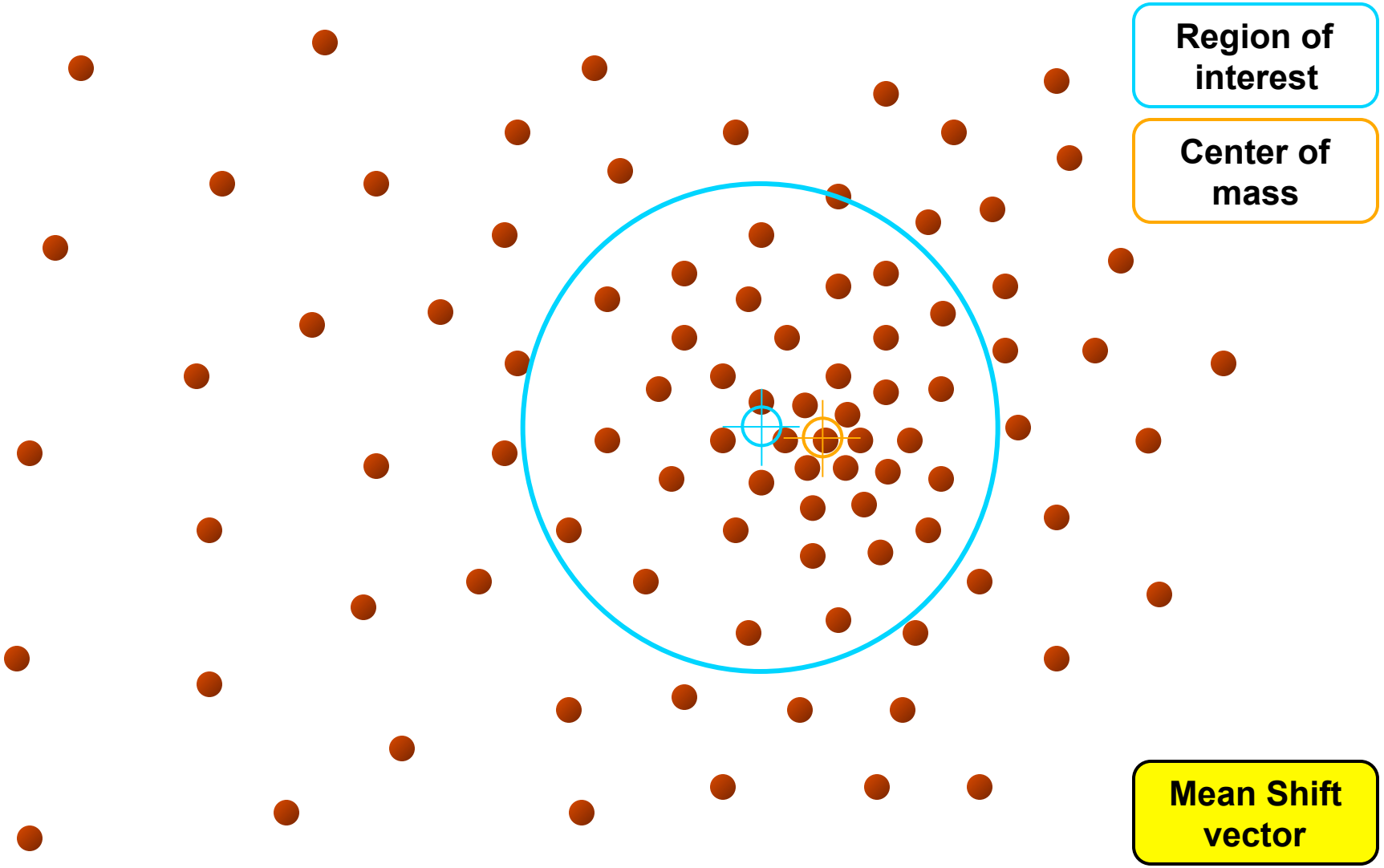
Slide by Y. Ukrainitz & B. Sarel

Mean shift



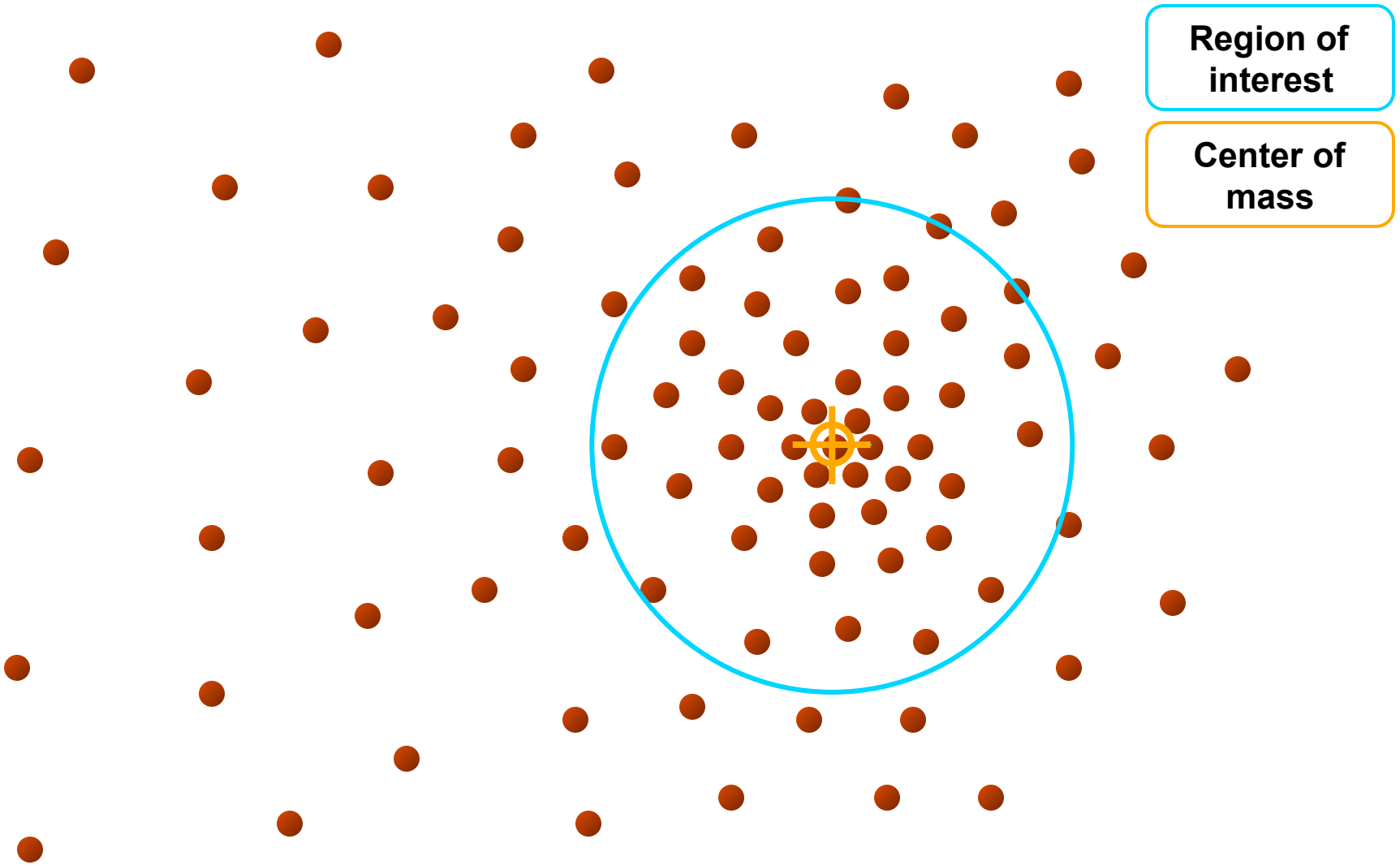
Slide by Y. Ukrainitz & B. Sarel

Mean shift



Slide by Y. Ukrainitz & B. Sarel

Mean shift



Slide by Y. Ukrainitz & B. Sarel

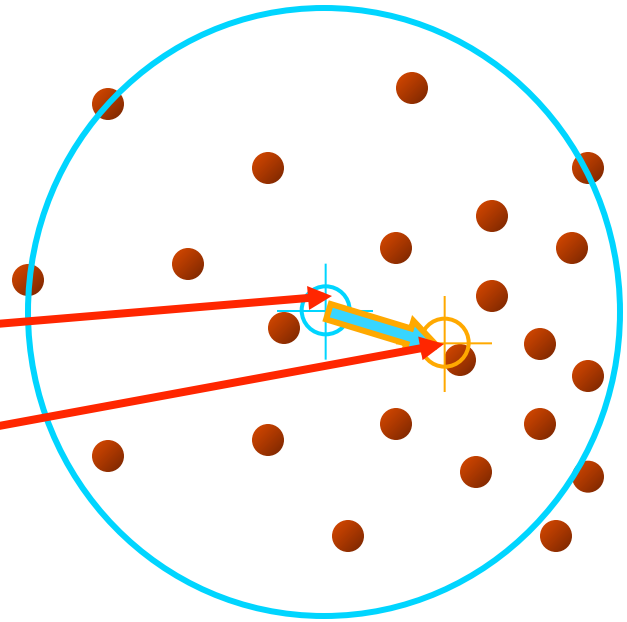
Computing the Mean Shift

Simple Mean Shift procedure:

- **Compute mean shift vector**

- **Translate the Kernel window by m(x)**

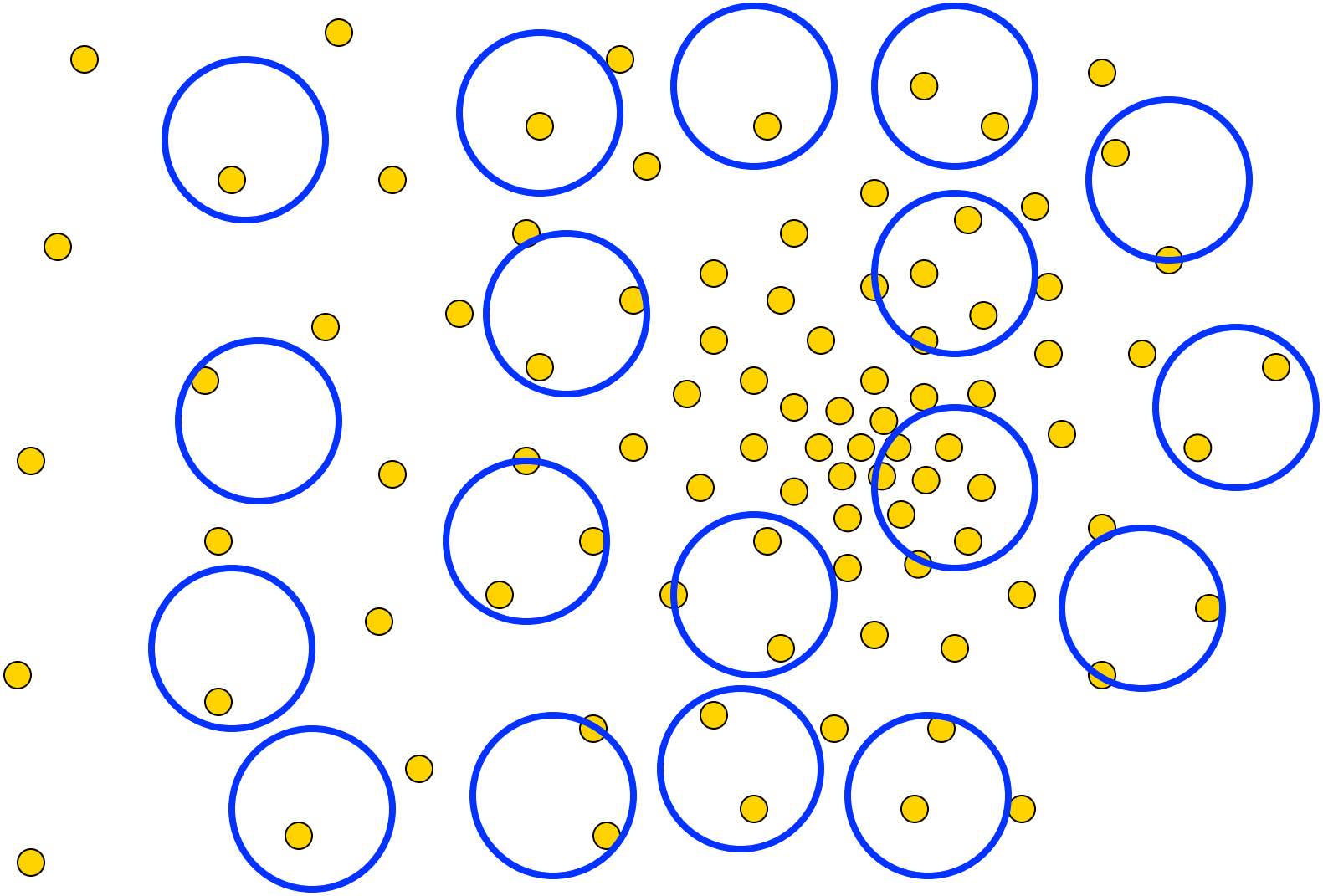
$$\mathbf{m}(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right)}{\sum_{i=1}^{n} g\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right)} \right] - \mathbf{x}$$



$$g(\mathbf{x}) = -k'(\mathbf{x})$$

Slide by Y. Ukrainitz & B. Sarel

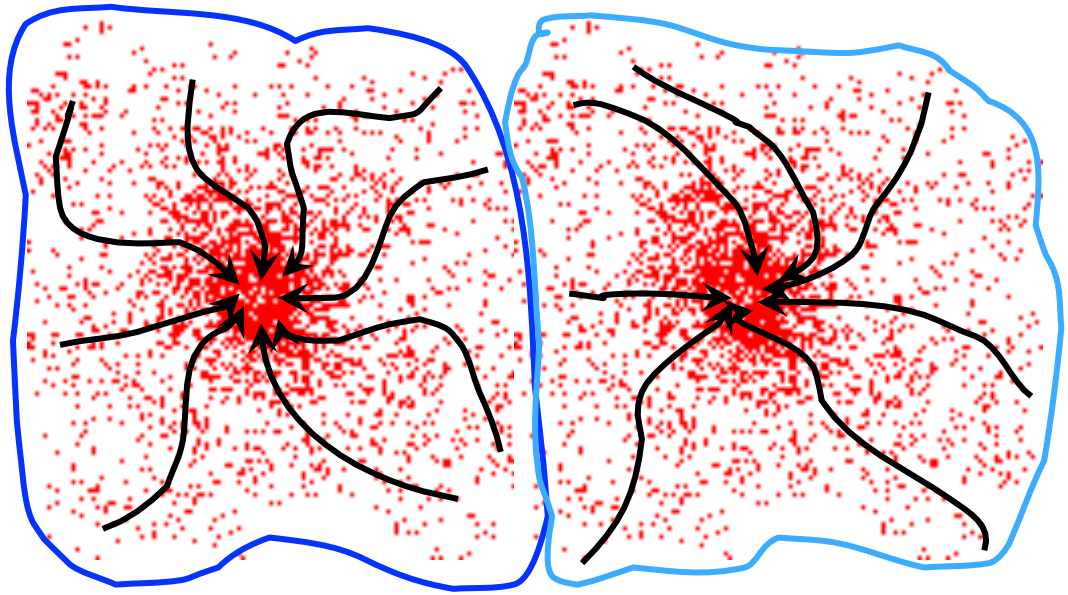
Real Modality Analysis



Attraction basin

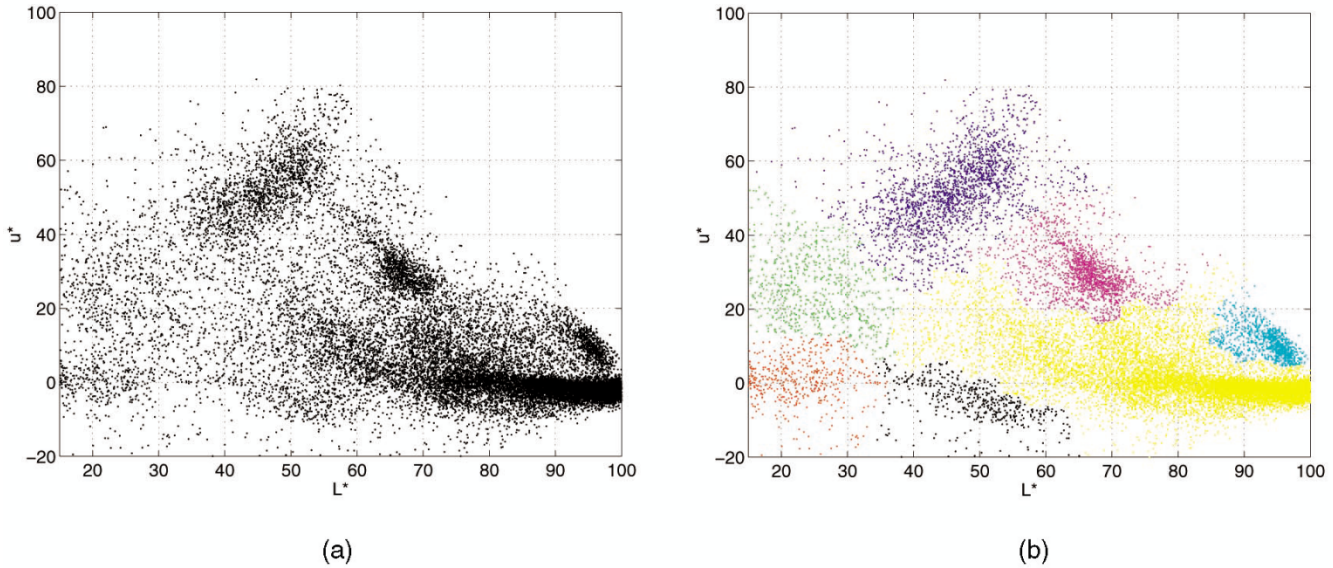
Attraction basin: the region for which all trajectories lead to the same mode

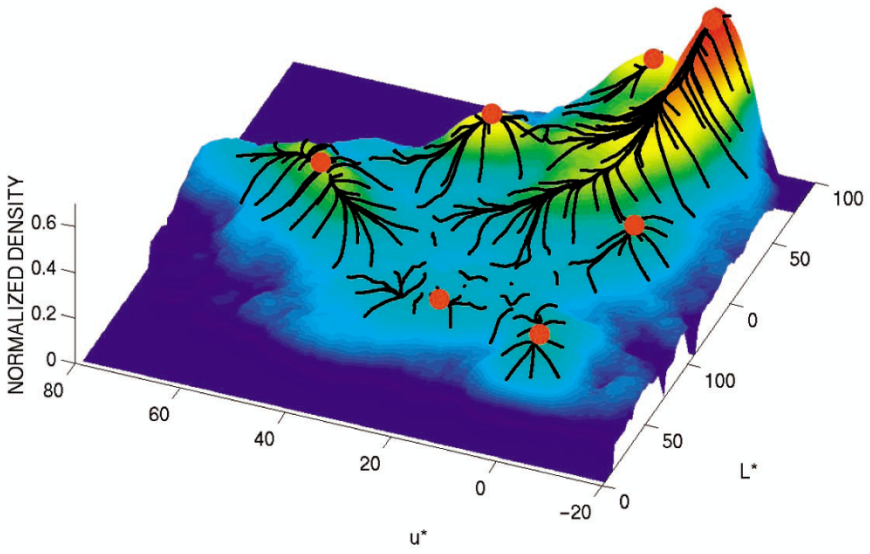
Cluster: all data points in the attraction basin of a mode



Slide by Y. Ukrainitz & B. Sarel

Attraction basin





Mean shift clustering

The mean shift algorithm seeks *modes* of the given set of points

1. Choose kernel and bandwidth
2. For each point:
   a) Center a window on that point
   b) Compute the mean of the data in the search window
   c) Center the search window at the new mean location
   d) Repeat (b,c) until convergence
3. Assign points that lead to nearby modes to the same cluster

Segmentation by Mean Shift

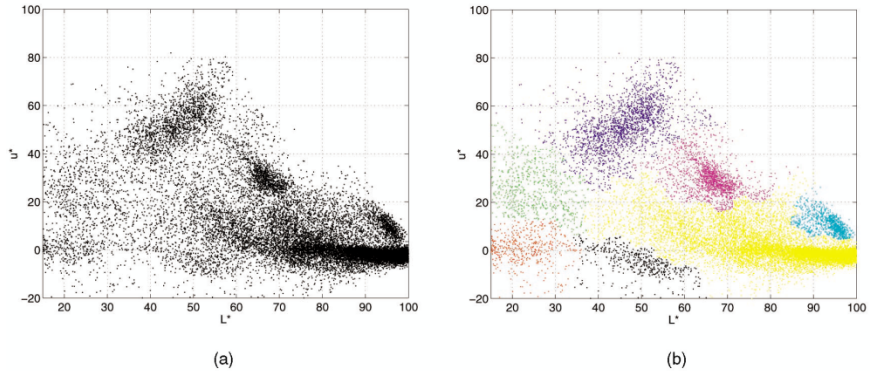
Compute features for each pixel (color, gradients, texture, etc); also store each pixel's position

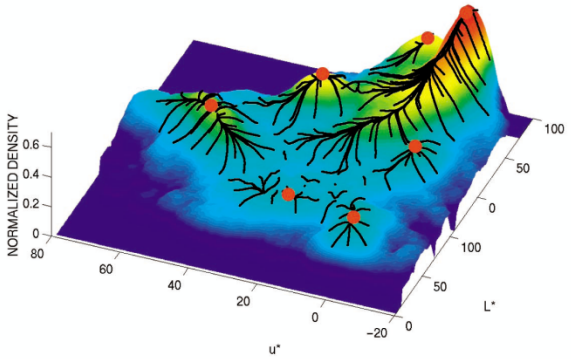
Set kernel size for features K_f and position K_s

Initialize windows at individual pixel locations

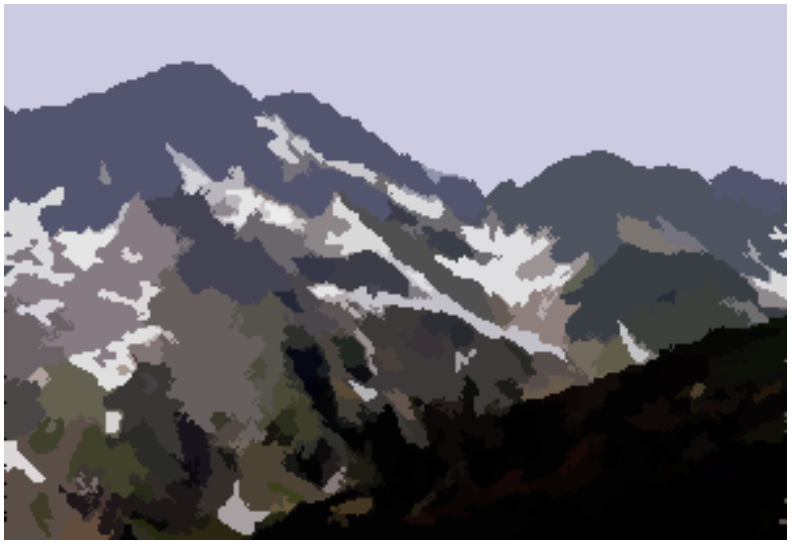
Perform mean shift for each window until convergence

Merge modes that are within width of K_f and K_s





Mean shift segmentation results



http://www.caip.rutgers.edu/~comanici/MSPAMI/msPami

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

Mean-shift: other issues

- Speedups
  - Binned estimation – replace points within some “bin” by point at center with mass
  - Fast search of neighbors – e.g., k-d tree or approximate NN
  - Update all windows in each iteration (faster convergence)

- Other tricks
  - Use kNN to determine window sizes adaptively

- Lots of theoretical support

  D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

Mean shift pros and cons

Pros

- Good general-purpose segmentation
- Flexible in number and shape of regions
- Robust to outliers

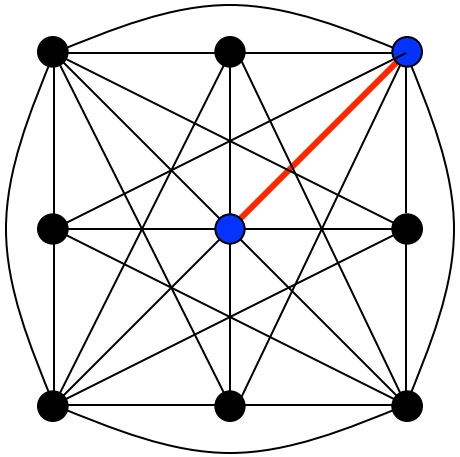
Cons

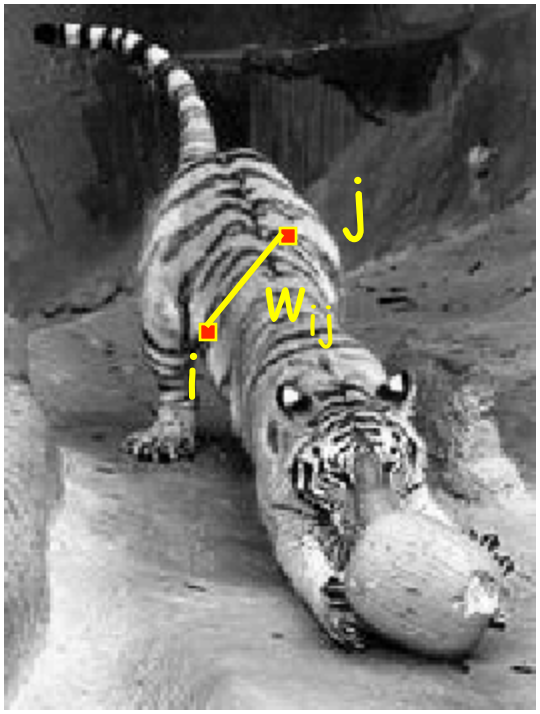
- Have to choose kernel size in advance
- Not suitable for high-dimensional features

When to use it

- Oversegmentation
- Multiple segmentations
- Tracking, clustering, filtering applications
  - D. Comaniciu, V. Ramesh, P. Meer: *Real-Time Tracking of Non-Rigid Objects using Mean Shift*, *Best Paper Award*, IEEE Conf. Computer Vision and Pattern Recognition (CVPR'00), Hilton Head Island, South Carolina, Vol. 2, 142-149, 2000

Segmentation as graph partitioning





- Node for every pixel
- Edge between every pair of pixels (or every pair of “sufficiently close” pixels)
- Each edge is weighted by the *affinity* or similarity of the two nodes

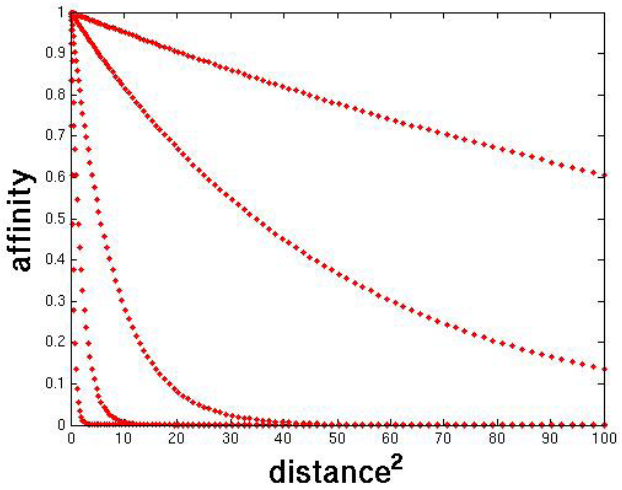
Source: S. Seitz

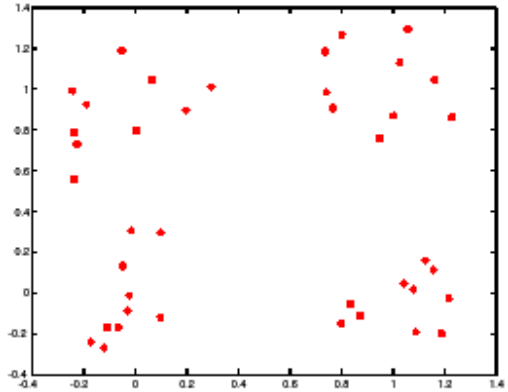
Measuring affinity

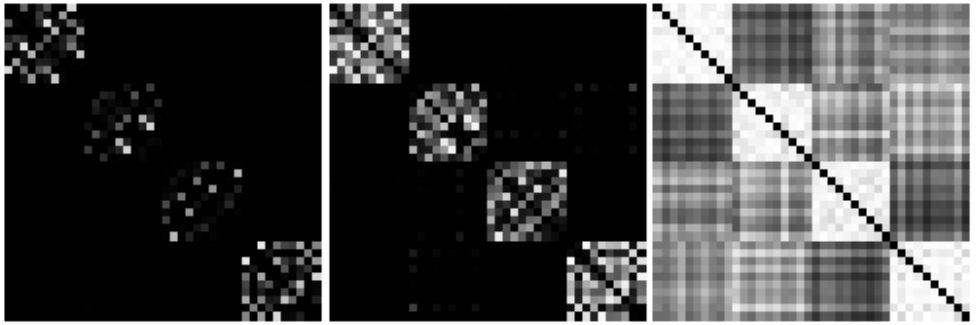
- Represent each pixel by a feature vector **x** and define an appropriate distance function

$$\text{affinity}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2\sigma^2}\text{dist}(\mathbf{x}_i, \mathbf{x}_j)^2\right)$$

Role of σ



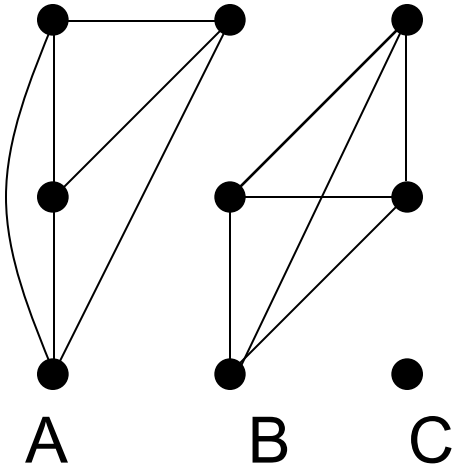




small σ

large σ

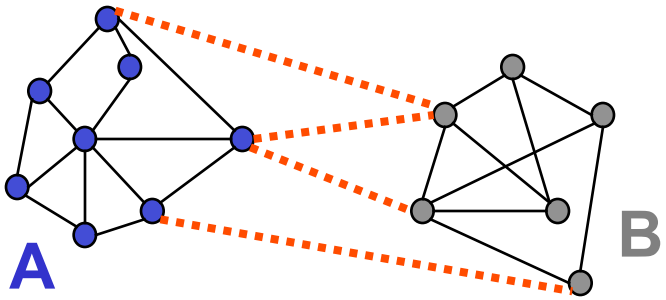
Segmentation as graph partitioning



- 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

Source: S. Seitz

Graph cut



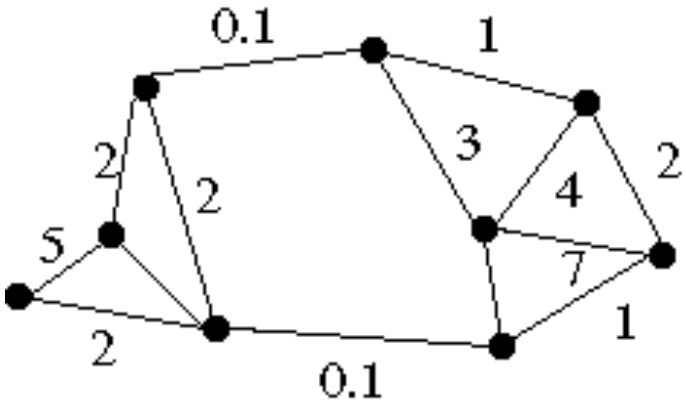
- Set of edges whose removal makes a graph disconnected
- Cost of a cut: sum of weights of cut edges
- A graph cut gives us a segmentation
  - What is a “good” graph cut and how do we find one?

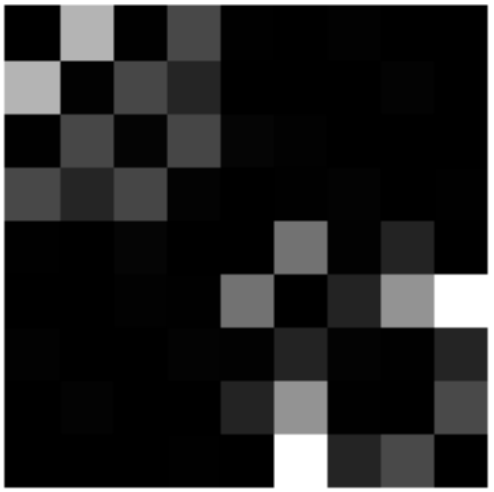
Source: S. Seitz

Minimum cut

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

Minimum cut example

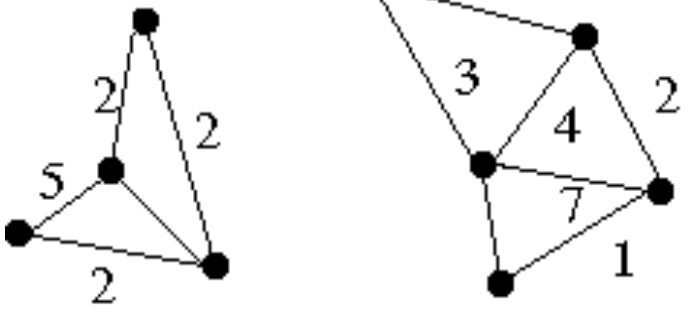


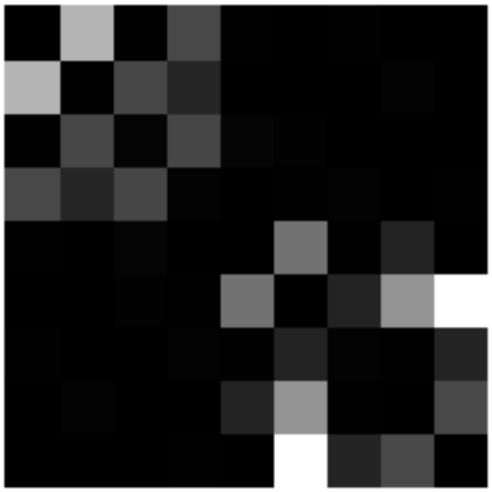


Minimum cut

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

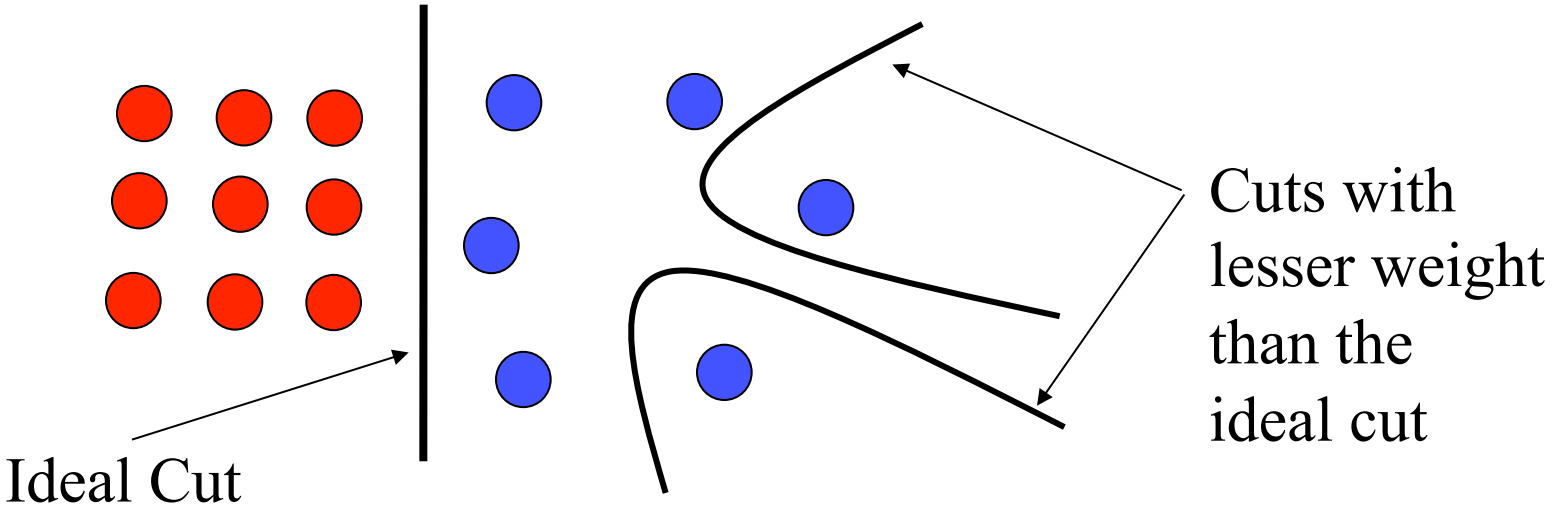
Minimum cut example





Normalized cut

- Drawback: minimum cut tends to cut off very small, isolated components



* Slide from Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Normalized cut

- Drawback: minimum cut tends to cut off very small, isolated components
- This can be fixed by normalizing the cut by the weight of all the edges incident to the segment
- The *normalized cut* cost is:

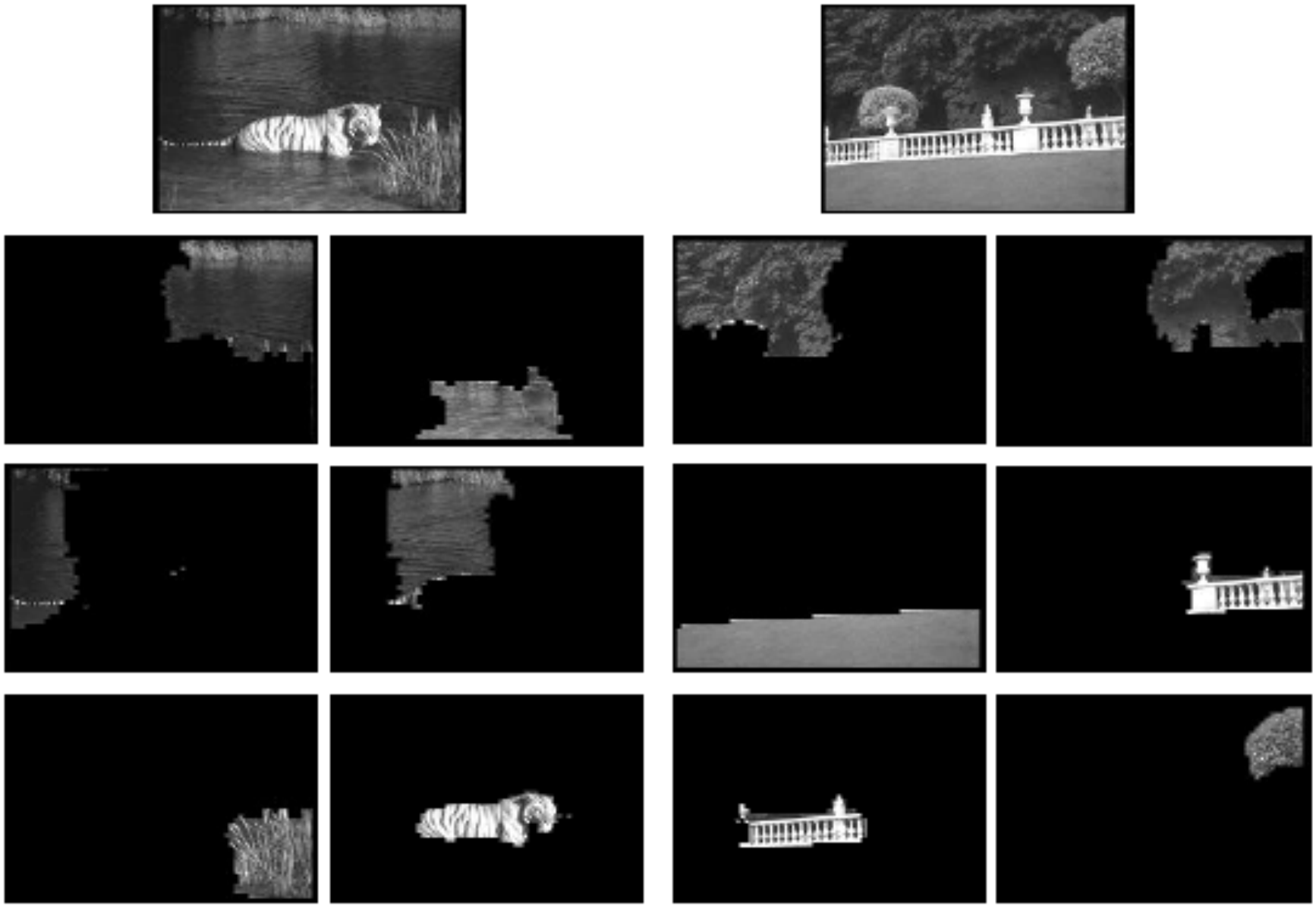
$$\frac{w(A,B)}{w(A,V)} + \frac{w(A,B)}{w(B,V)}$$

$w(A, B)$ = sum of weights of all edges between A and B

- Finding the globally optimal cut is NP-complete, but a relaxed version can be solved using a generalized eigenvalue problem

J. Shi and J. Malik. Normalized cuts and image segmentation. PAMI 2000

Example result



Results: Berkeley Segmentation Engine

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

Results: Berkeley Segmentation Engine

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

Normalized cuts: Pro and con

- Pro
  - Generic framework, can be used with many different features and affinity formulations
- Con
  - High storage requirement and time complexity: involves solving a generalized eigenvalue problem of size n x n, where n is the number of pixels

Efficient graph-based segmentation

- Runs in time nearly linear in the number of edges
- Easy to control coarseness of segmentations
- Results can be unstable

P. Felzenszwalb and D. Huttenlocher,
Efficient Graph-Based Image Segmentation, IJCV 2004

Segmentation as labeling

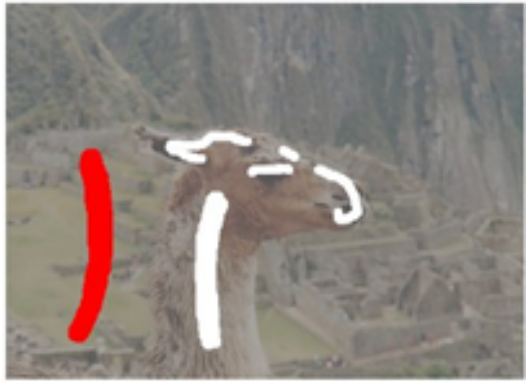
- Suppose we want to segment an image into foreground and background
  - Binary labeling problem



Credit: N. Snavely

Segmentation as labeling

- Suppose we want to segment an image into foreground and background
  - Binary labeling problem



User sketches out a few strokes on foreground and background…

How do we label the rest of the pixels?

Source: N. Snavely

Binary segmentation as energy minimization

- Define a labeling *L* as an assignment of each pixel with a 0-1 label (background or foreground)
- Find the labeling *L* that minimizes

$$E(L) = E_d(L) + \lambda E_s(L)$$

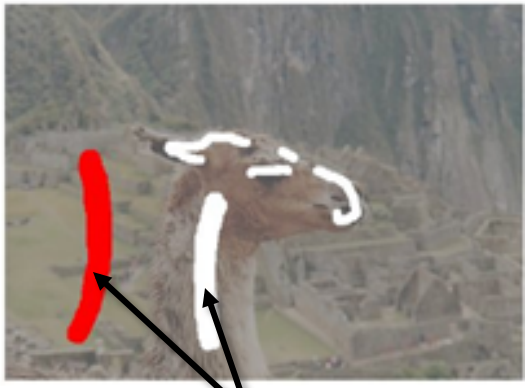
data term **smoothness term**

How similar is each labeled pixel to the foreground or background?

Encourage spatially coherent segments

Source: N. Snavely

$$E(L) = E_d(L) + \lambda E_s(L)$$



$\tilde{L}(x, y)$

$$E_d(L) = \sum_{(x,y)} C(x, y, L(x, y))$$

$$C(x, y, L(x, y)) = \begin{cases} \infty & \text{if } L(x, y) \neq \tilde{L}(x, y) \\ C'(x, y, L(x, y)) & \text{otherwise} \end{cases}$$

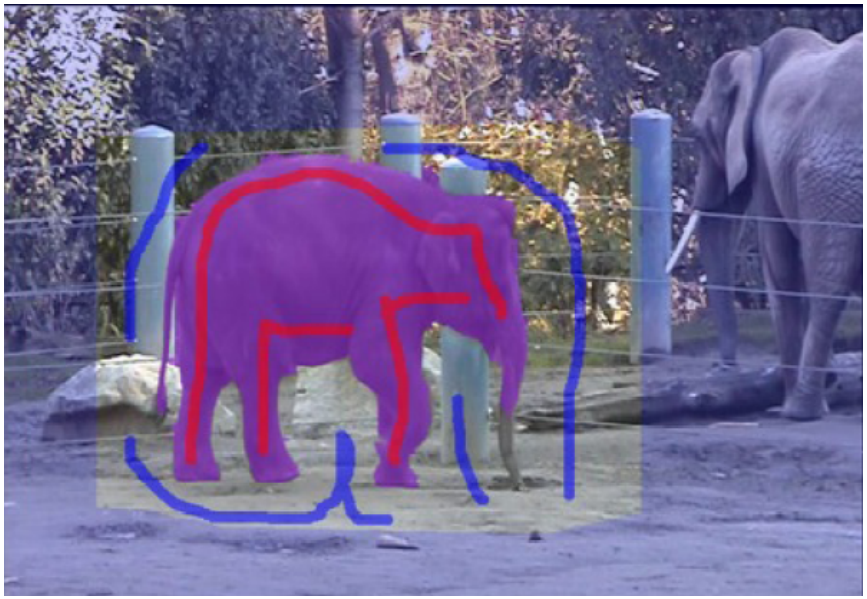
$C'(x, y, 0)$: "distance" from pixel to background

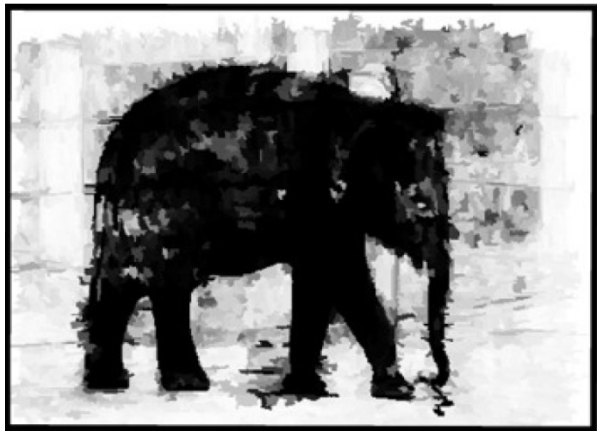
$C'(x, y, 1)$: "distance" from pixel to foreground

} computed by creating a color model from user-labeled pixels

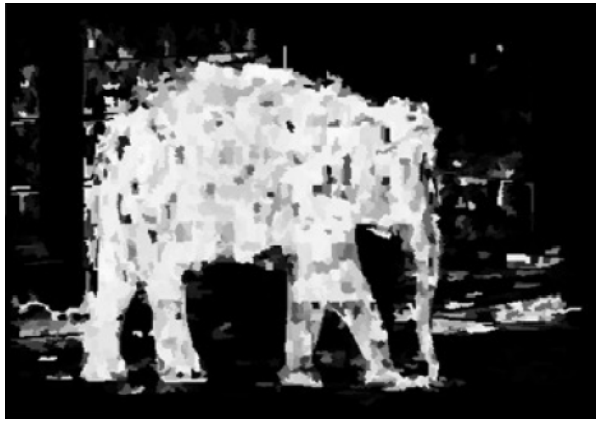
Source: N. Snavely

$$E(L) = E_d(L) + \lambda E_s(L)$$





$C'(x, y, 0)$

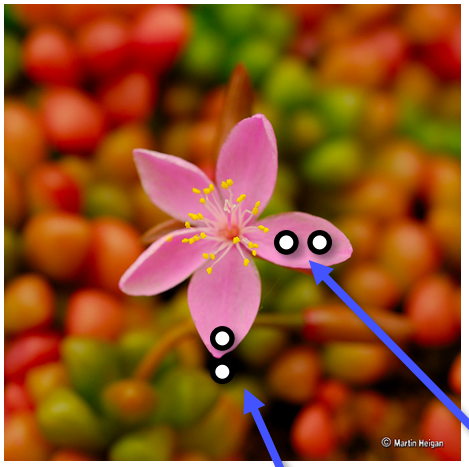


$C'(x, y, 1)$

Source: N. Snavely

$$E(L) = E_d(L) + \lambda E_s(L)$$

- Neighboring pixels should generally have the same labels
  - Unless the pixels have very different intensities



w_{pq} = 10.0

w_{pq} = 0.1

$$E_s(L) = \sum_{\text{neighbors } (p,q)} w_{pq}|L(p) - L(q)|$$

w_{pq} : similarity in intensity of *p* and *q*

Source: N. Snavely

Binary segmentation as energy minimization

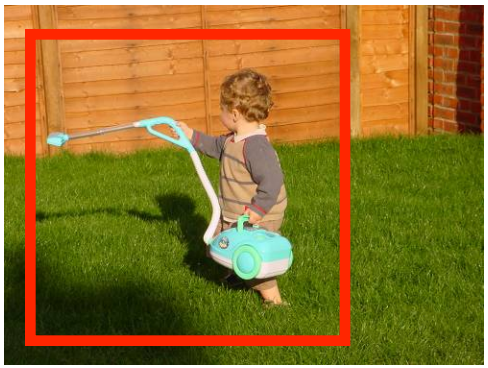
$$E(L) = E_d(L) + \lambda E_s(L)$$

- For this problem, we can efficiently find the global minimum using the max flow / min cut algorithm

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

Source: N. Snavely

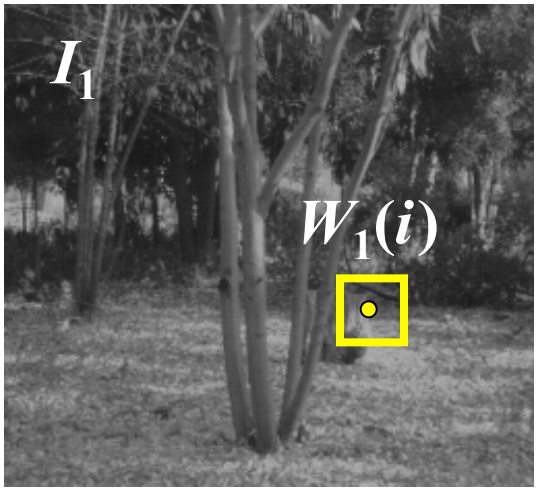
GrabCut

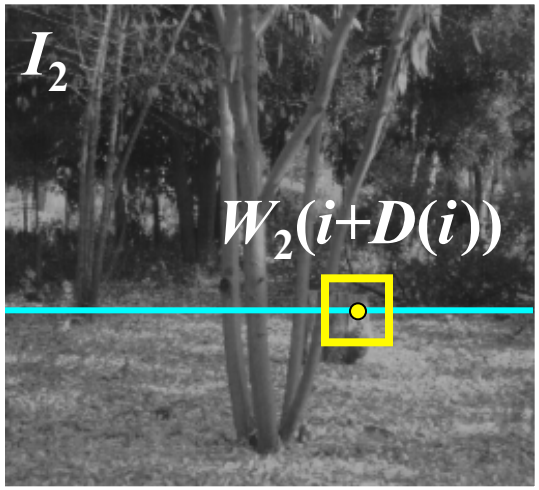


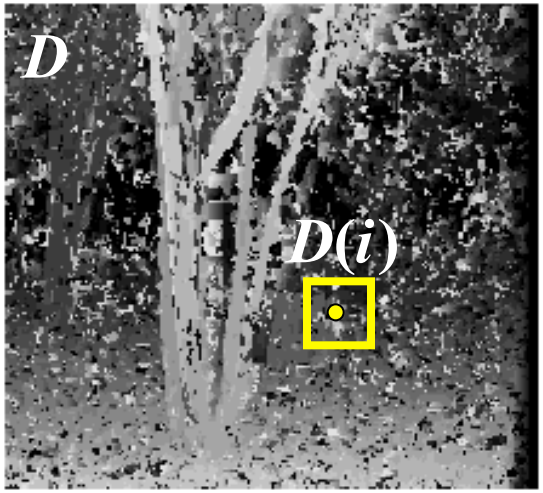



C. Rother, V. Kolmogorov, and A. Blake,
"GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts,
SIGGRAPH 2004

Recall: Stereo as energy minimization







$$E(D) = \underbrace{\sum_i \left(W_1(i) - W_2(i + D(i))\right)^2}_{\text{data term}} + \lambda \underbrace{\sum_{\text{neighbors } i,j} \rho\left(D(i) - D(j)\right)}_{\text{smoothness term}}$$

- Energy functions of this form can be minimized using *graph cuts*

Y. Boykov, O. Veksler, and R. Zabih,
Fast Approximate Energy Minimization via Graph Cuts, PAMI 2001

Further reading

- Nicely written mean-shift explanation (with math)

http://saravananthirumuruganathan.wordpress.com/2010/04/01/introduction-to-mean-shift-algorithm/

  - Includes .m code for mean-shift clustering

- Mean-shift paper by Comaniciu and Meer

http://www.caip.rutgers.edu/~comanici/Papers/MsRobustApproach.pdf

- Adaptive mean shift in higher dimensions

http://mis.hevra.haifa.ac.il/~ishimshoni/papers/chap9.pdf

- Contours to regions (watershed): Arbelaez et al. 2009

http://www.eecs.berkeley.edu/~arbelaez/publications/Arbelaez_Maire_Fowlkes_Malik_CVPR2009.pdf