



CSCI 631

Foundations of Computer Vision

Ifeoma Nwogu
ion@cs.rit.edu



Schedule

- Last class
 - Textures
- Today
 - Segmentation especially k-means
- Readings for today (Optional):
 - Forsyth and Ponce chapter 9



Digital image manipulations

- Image processing

image in → image out

- Image analysis

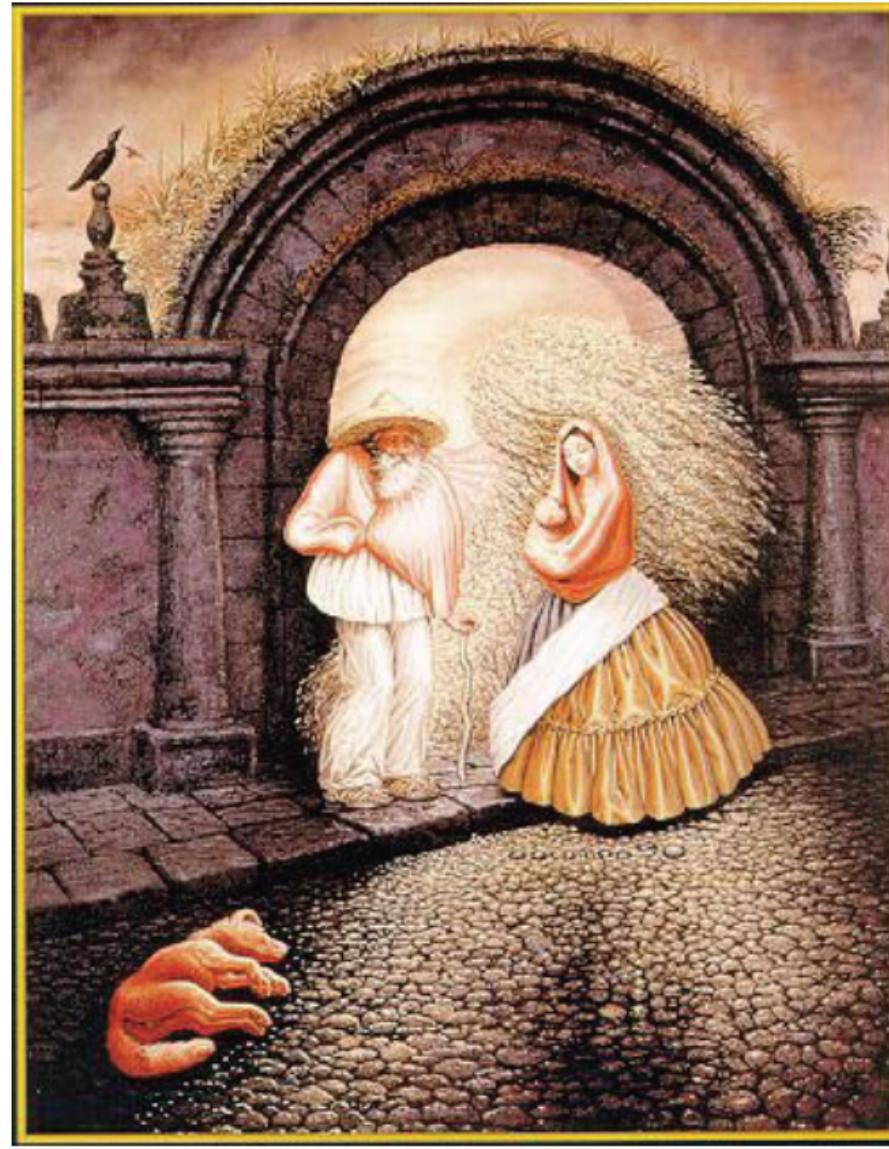
image in → measurements out

- Image understanding

image in → high-level description out



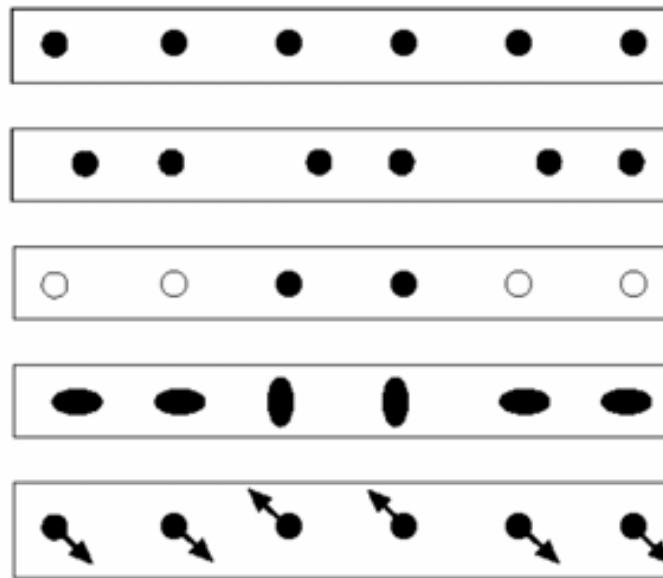
Perceptual grouping





Motion and perceptual organization

Humans interpret information “collectively” or in groups



1

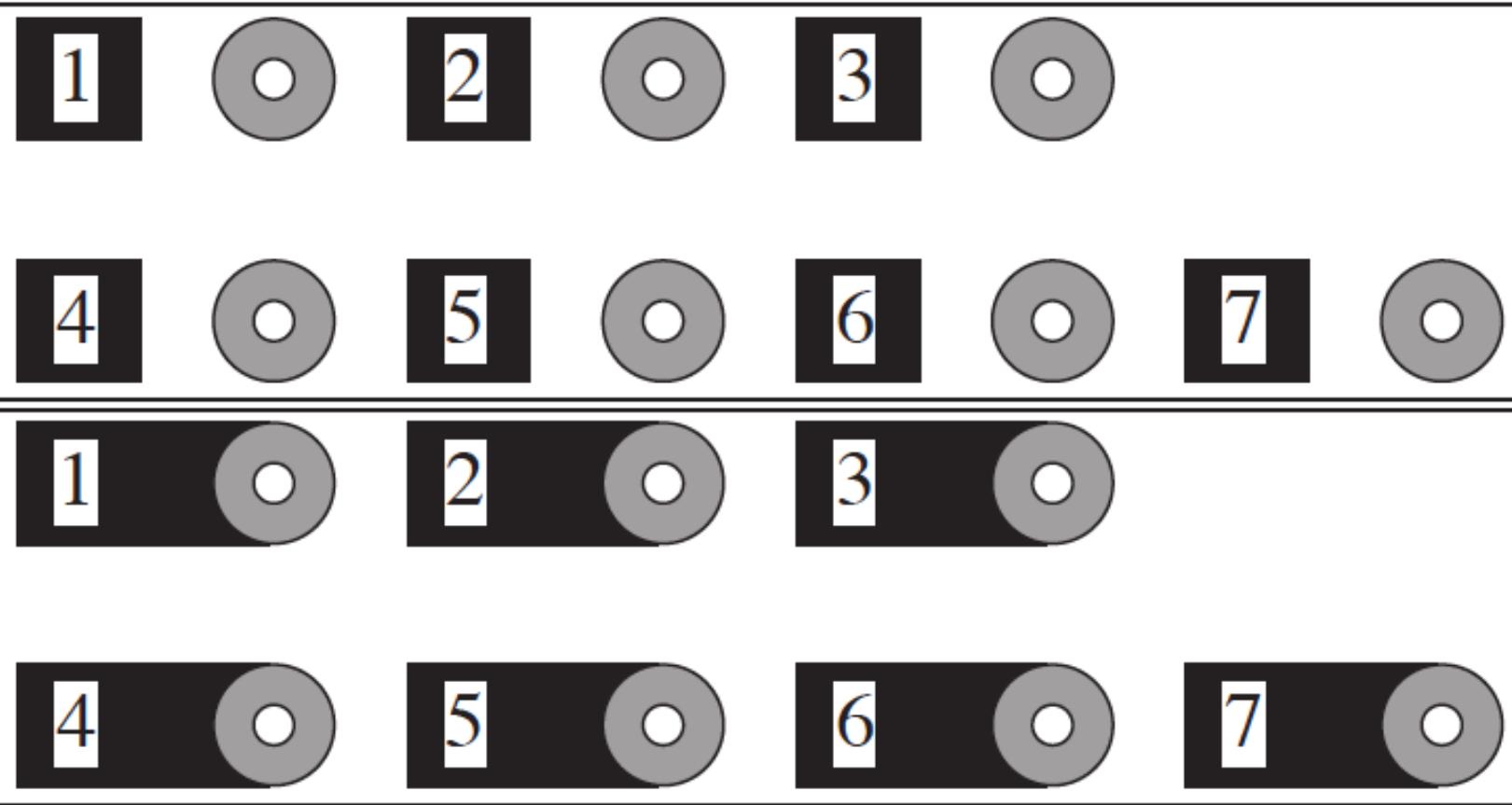


FIGURE 9.7: An example of grouping phenomena in real life. The buttons on an elevator in the computer science building at U.C. Berkeley used to be laid out as in the **top** figure. It was common to arrive at the wrong floor and discover that this was because you'd pressed the wrong button; the buttons are difficult to group unambiguously with the correct label, and it is easy to get the wrong grouping at a quick glance. A public-spirited individual filled in the gap between the numbers and the buttons, as in the **bottom** figure, and the confusion stopped because the proximity cue had been disambiguated.



Image segmentation

The main goal is to identify groups of pixels/regions that “go together perceptually”

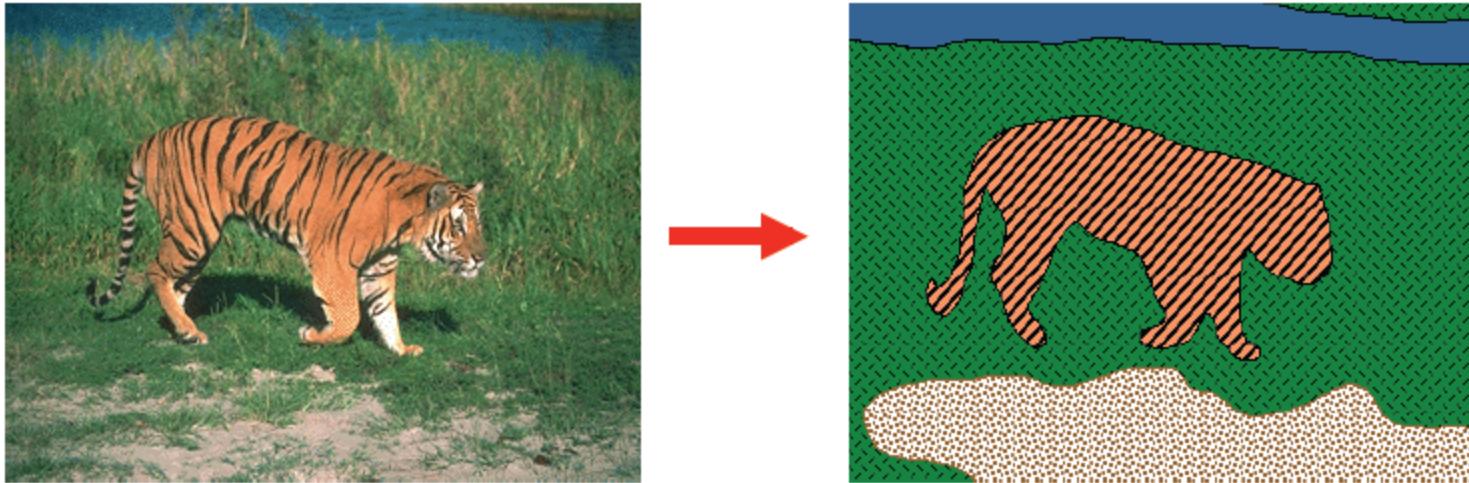




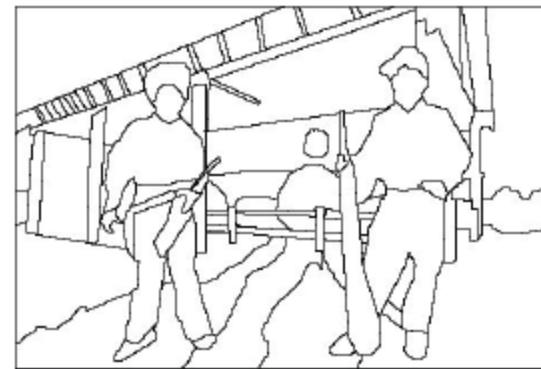
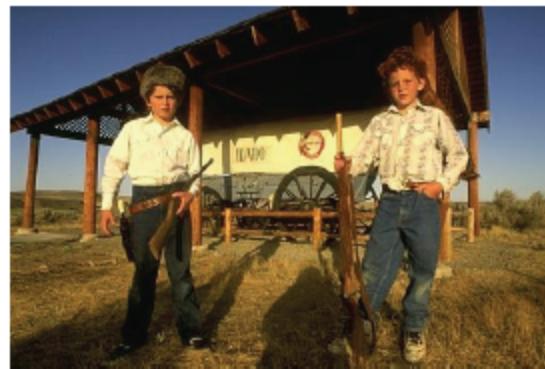
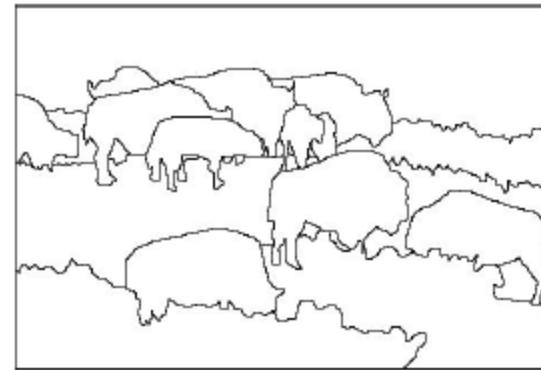
Image segmentation

Separate image into “coherent objects”

Image



Human segmentation





Examples of segmented images





Why do segmentation?

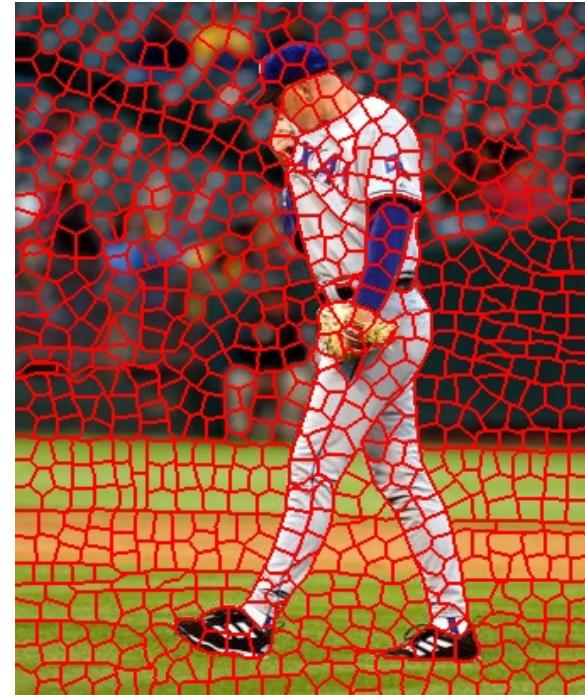
- To obtain primitives for other tasks
- For perceptual organization, recognition
- For graphics, image manipulation



Task 1: Primitives for other tasks

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

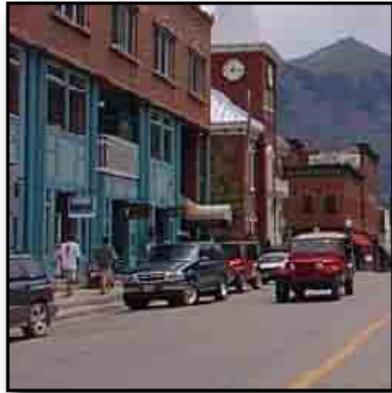
“superpixels”





Example of segments as primitives for recognition

-



(a) Query Image



(b) Retrieval Set



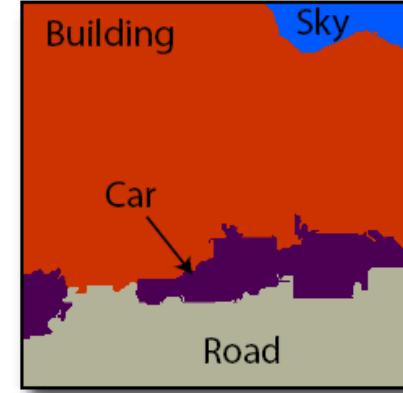
(c) Superpixels

Building

Road



(d) Per-class Likelihoods
Building
Road
Car
Sky



(e) Final Labeling

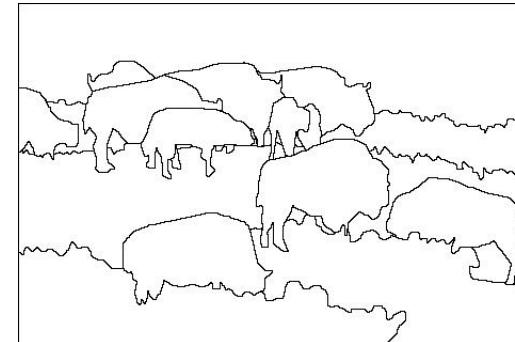


Task 2: Recognition

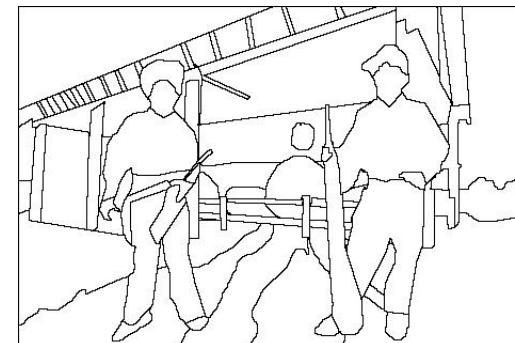
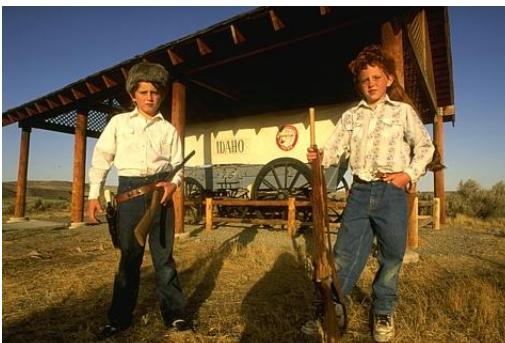
- Separate image into coherent “objects”
 - “Bottom-up” or “top-down” process?
 - Supervised or unsupervised?



image



human segmentation



Berkeley segmentation database:

09/16/19 <http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/>

CSCI 631 – Lecture 7

Page 13 of 78



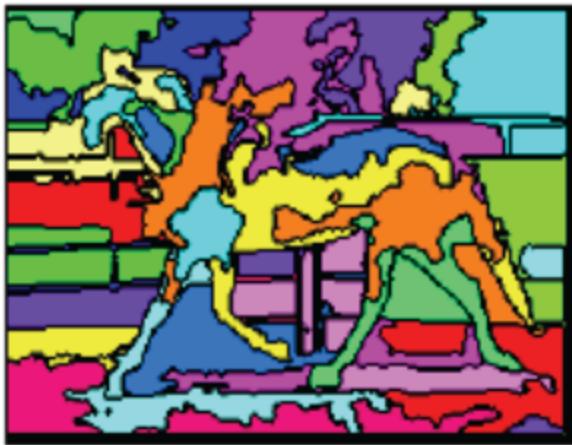
Task 3: Image manipulation

- Interactive segmentation for graphics

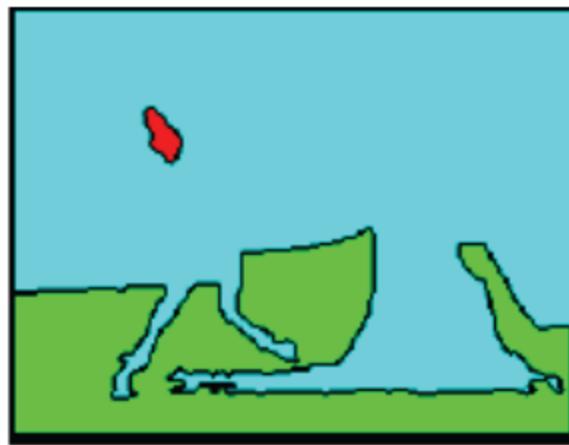




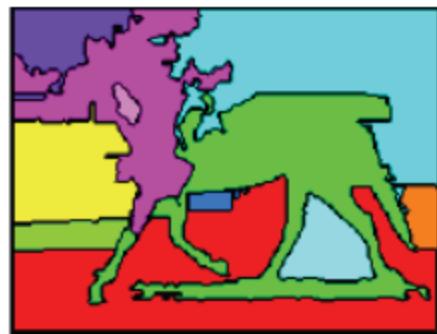
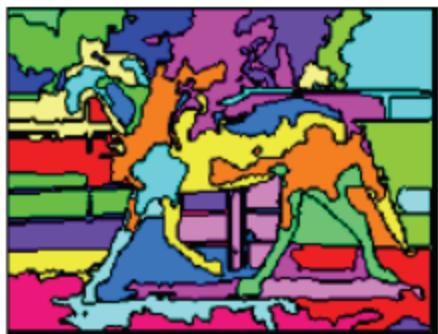
Challenges with segmentation



Oversegmentation



Undersegmentation

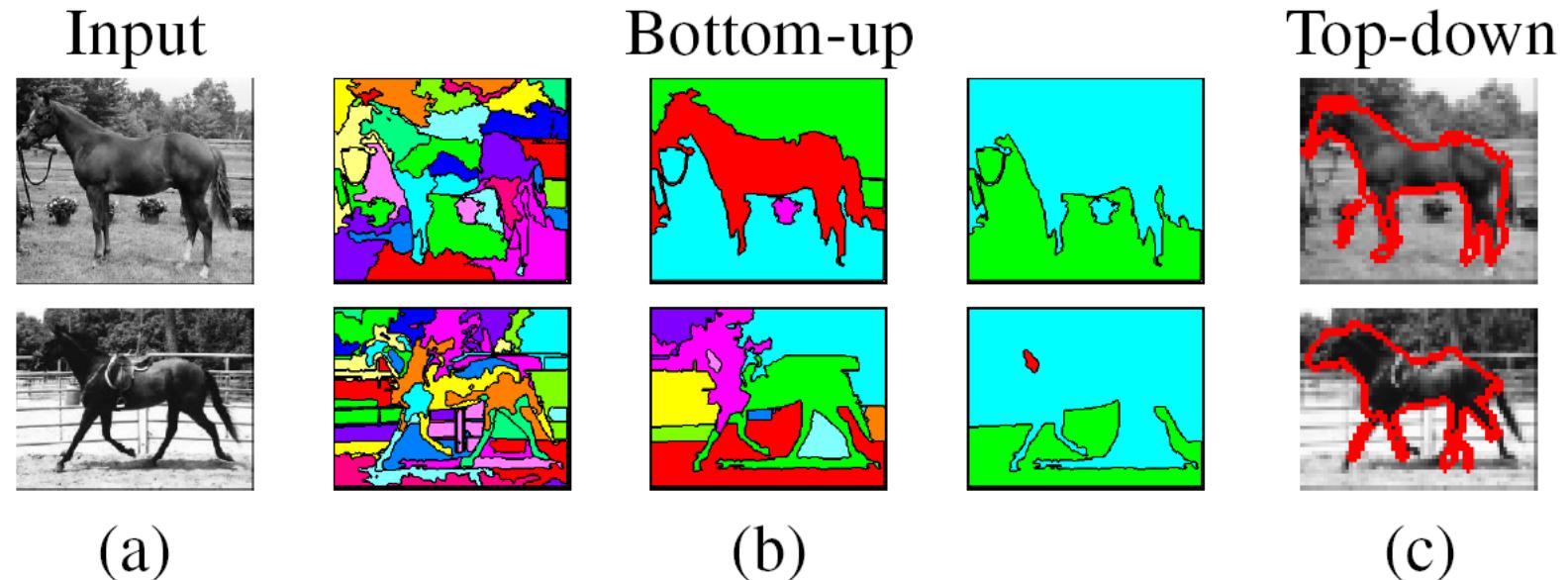


Multiple Segmentations



High-level approaches to segmentation

- Bottom-up: group tokens with similar features
- Top-down: group tokens that likely belong to the same object





Approaches to segmentation

- Clustering for segmentation
- Graph partitioning for segmentation
- Labeling as segmentation



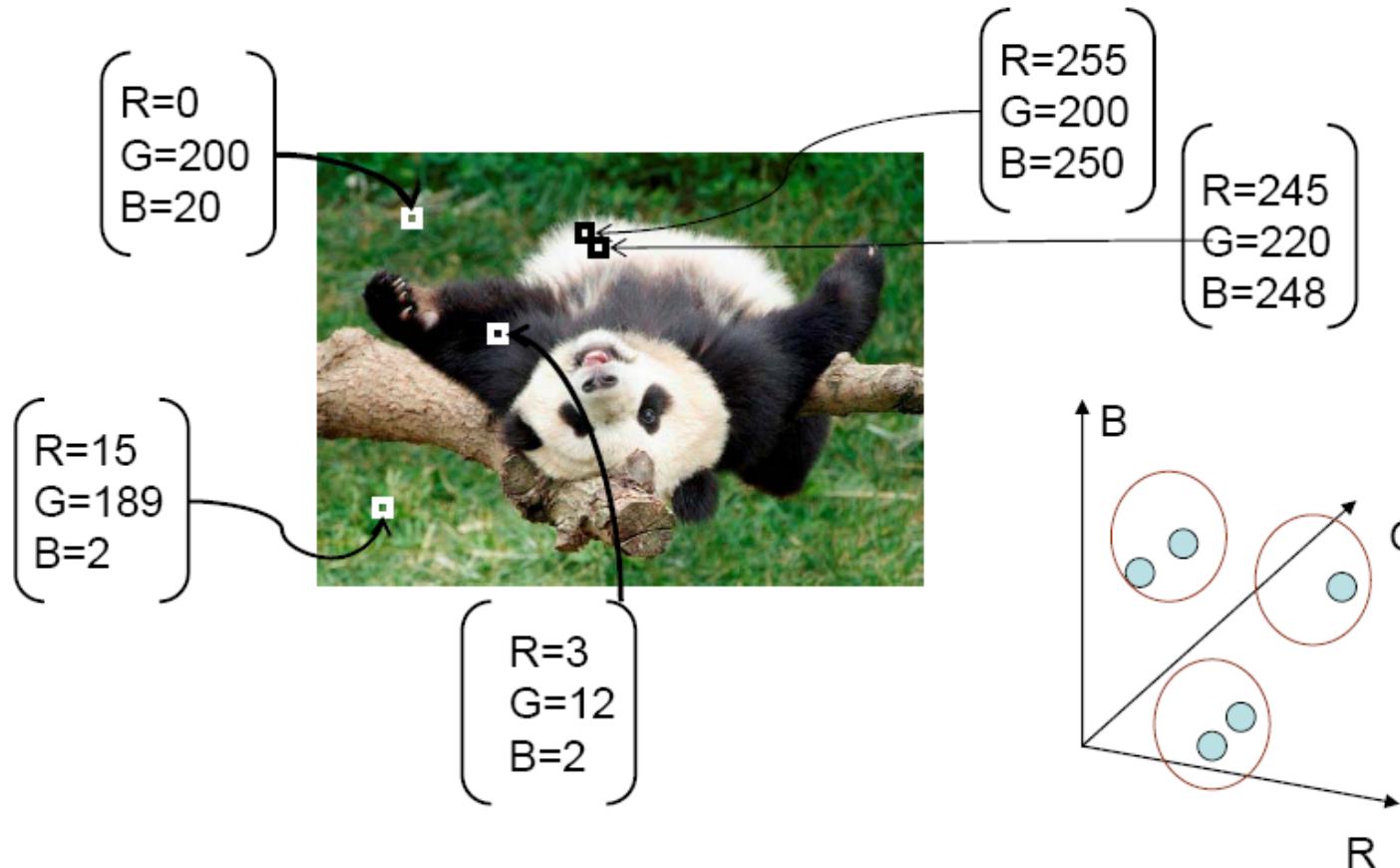
Segmentation as clustering

- **Clustering:** grouping 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?



Segmentation as clustering

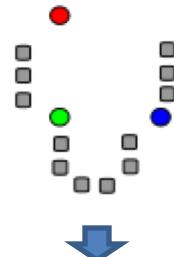
- Cluster similar pixels (features) together



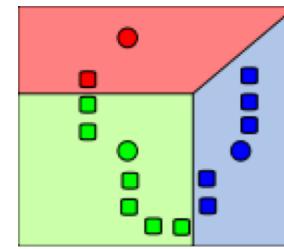


K-means algorithm

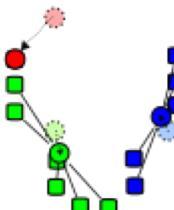
1. Randomly select K centers



2. Assign each point to nearest center



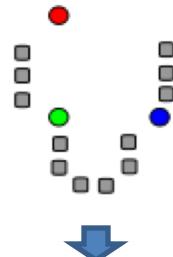
3. Compute new center (mean) for each cluster



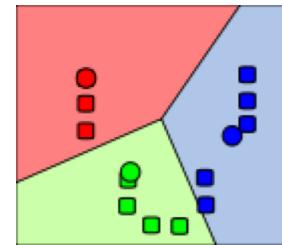


K-means algorithm

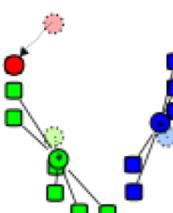
1. Randomly select K centers



2. Assign each point to nearest center



3. Compute new center (mean) for each cluster



Back to 2



K-means

1. Initialize cluster centers: \mathbf{c}^0 ; t=0
2. Assign each point to the closest center

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

3. Update cluster centers as the mean of the points

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

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



How to choose the number of clusters?

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



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



Common similarity/distance measures

- P-norms

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

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

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

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

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

}

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

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



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



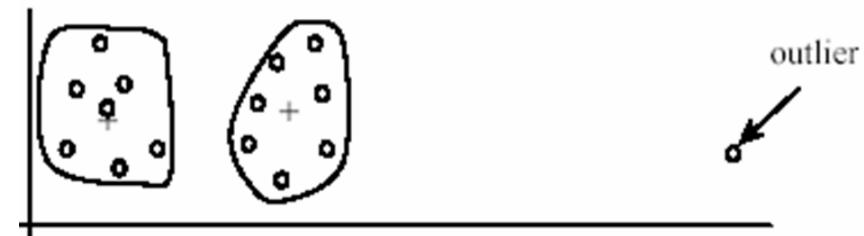
K-medoids

- 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

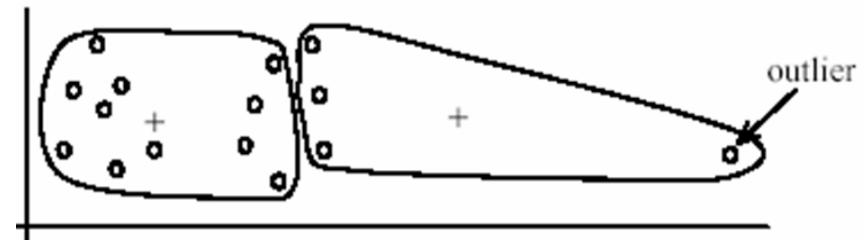


K-Means pros and cons

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



(B): Ideal clusters

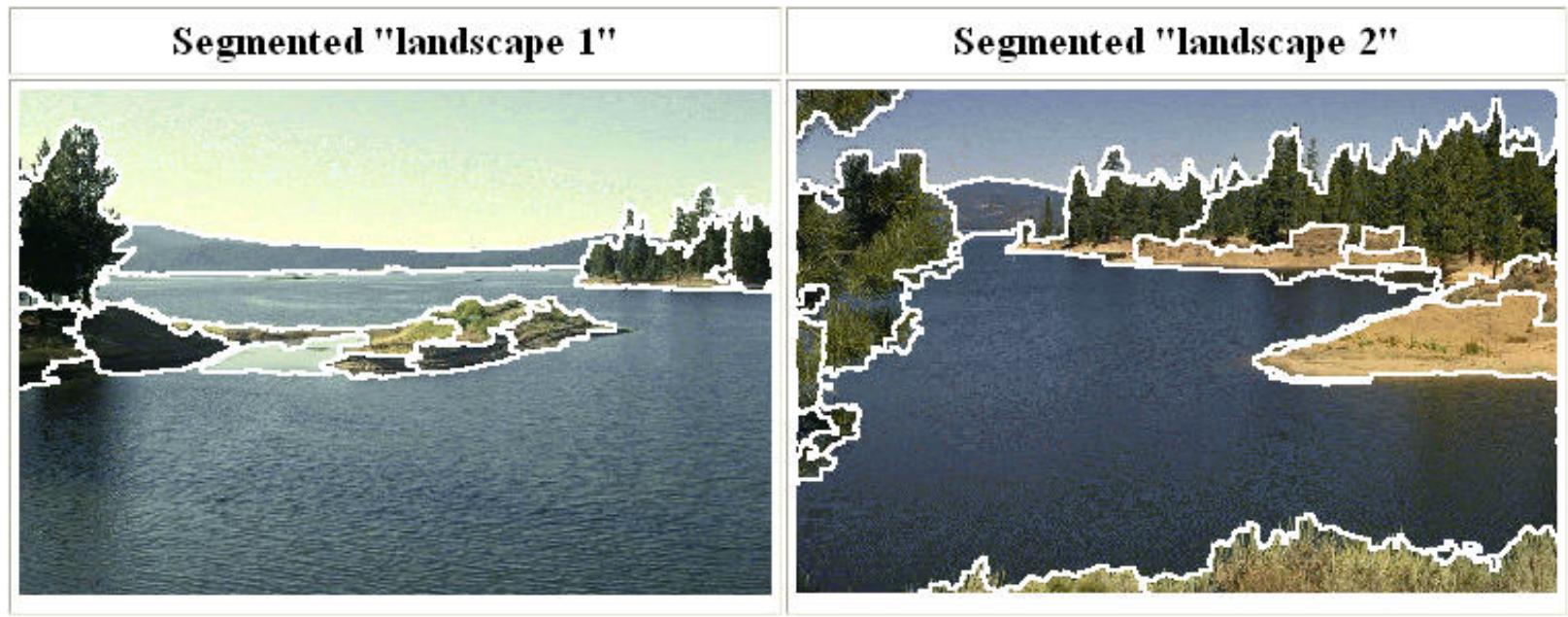




Mean shift segmentation

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

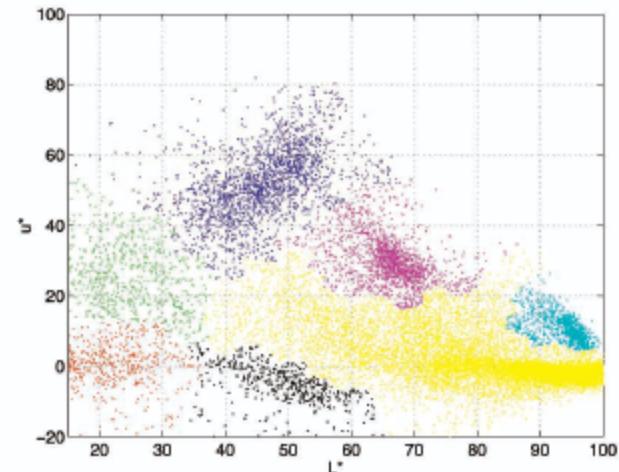
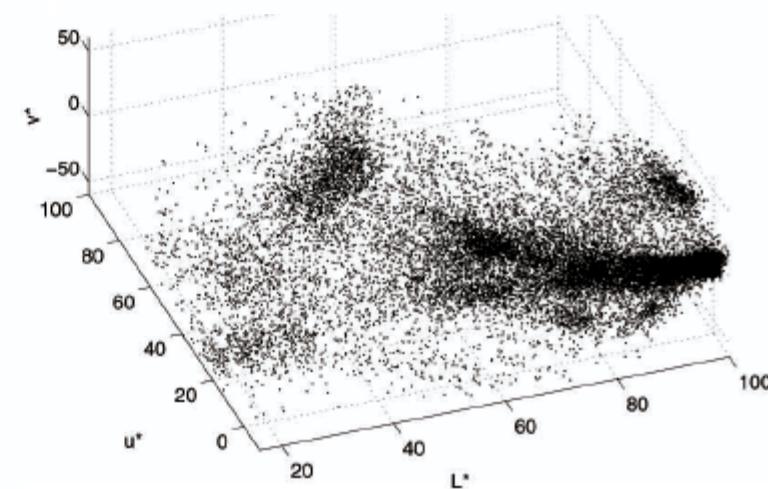
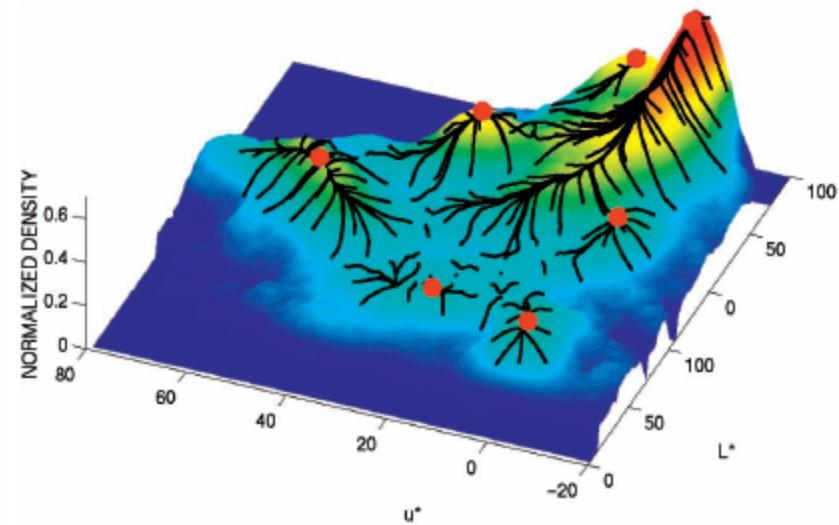
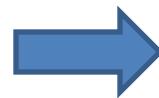
- Versatile technique for clustering-based segmentation





Mean shift algorithm

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





Kernel density estimation (KDE)

- A non-parametric way to estimate the probability density function of a random variable.
- Inferences about a population are made based only on a finite data sample.
- Also termed the Parzen–Rosenblatt window method,
 - Named after Emanuel Parzen and Murray Rosenblatt, who are usually credited with independently creating it in its current form



Kernel density estimation

Kernel density estimation function

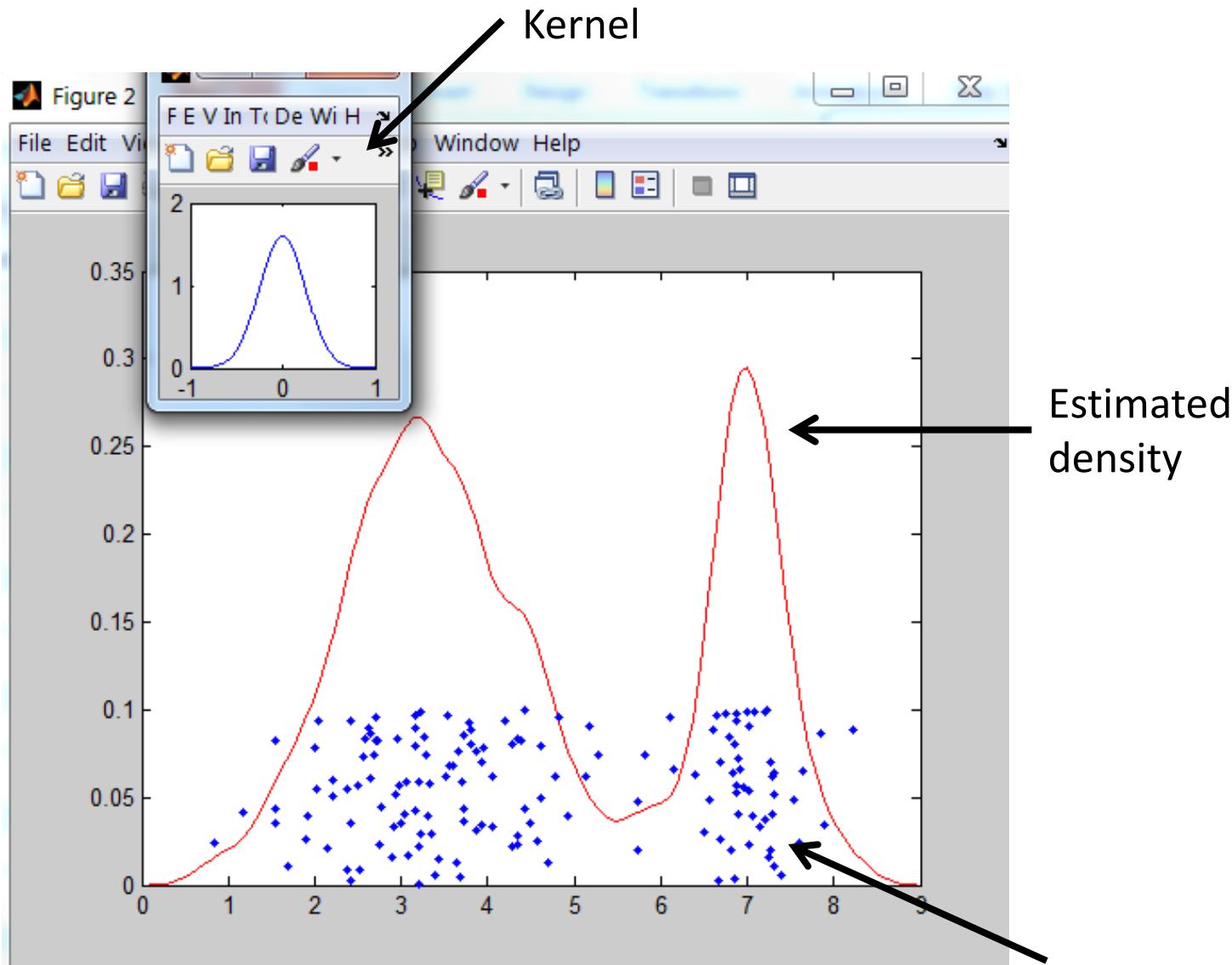
$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)$$

Gaussian kernel

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

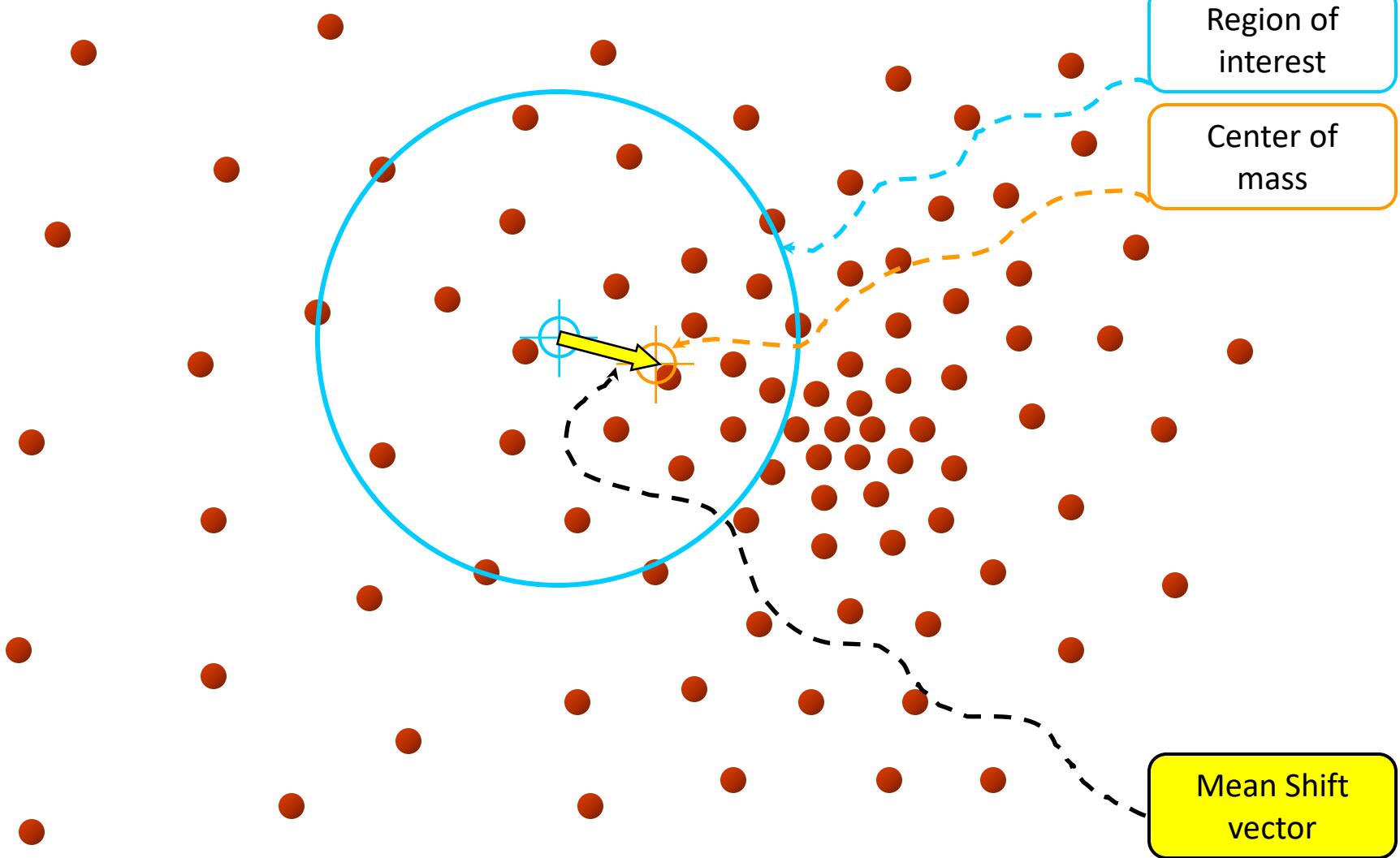


Kernel density estimation



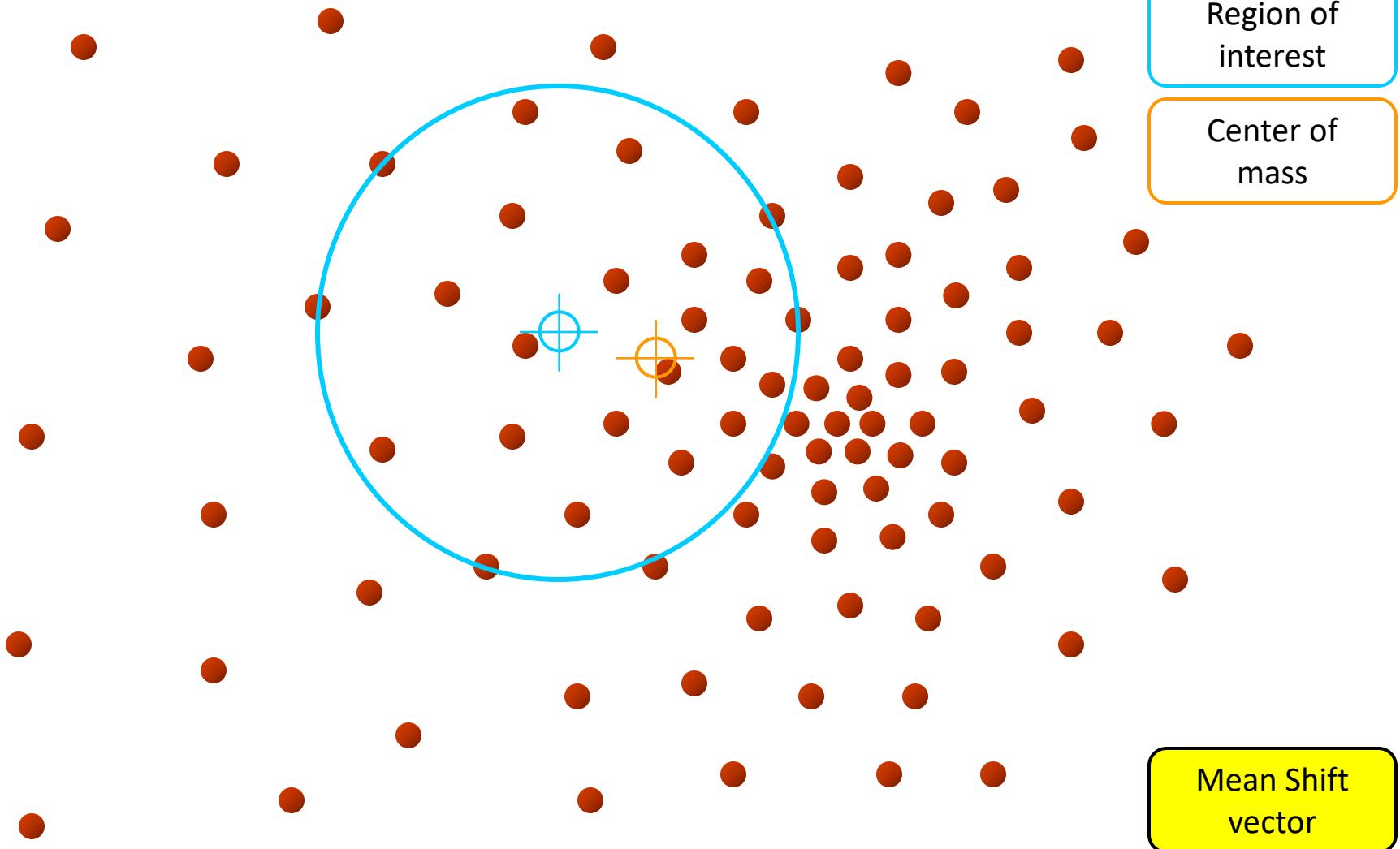


Mean shift



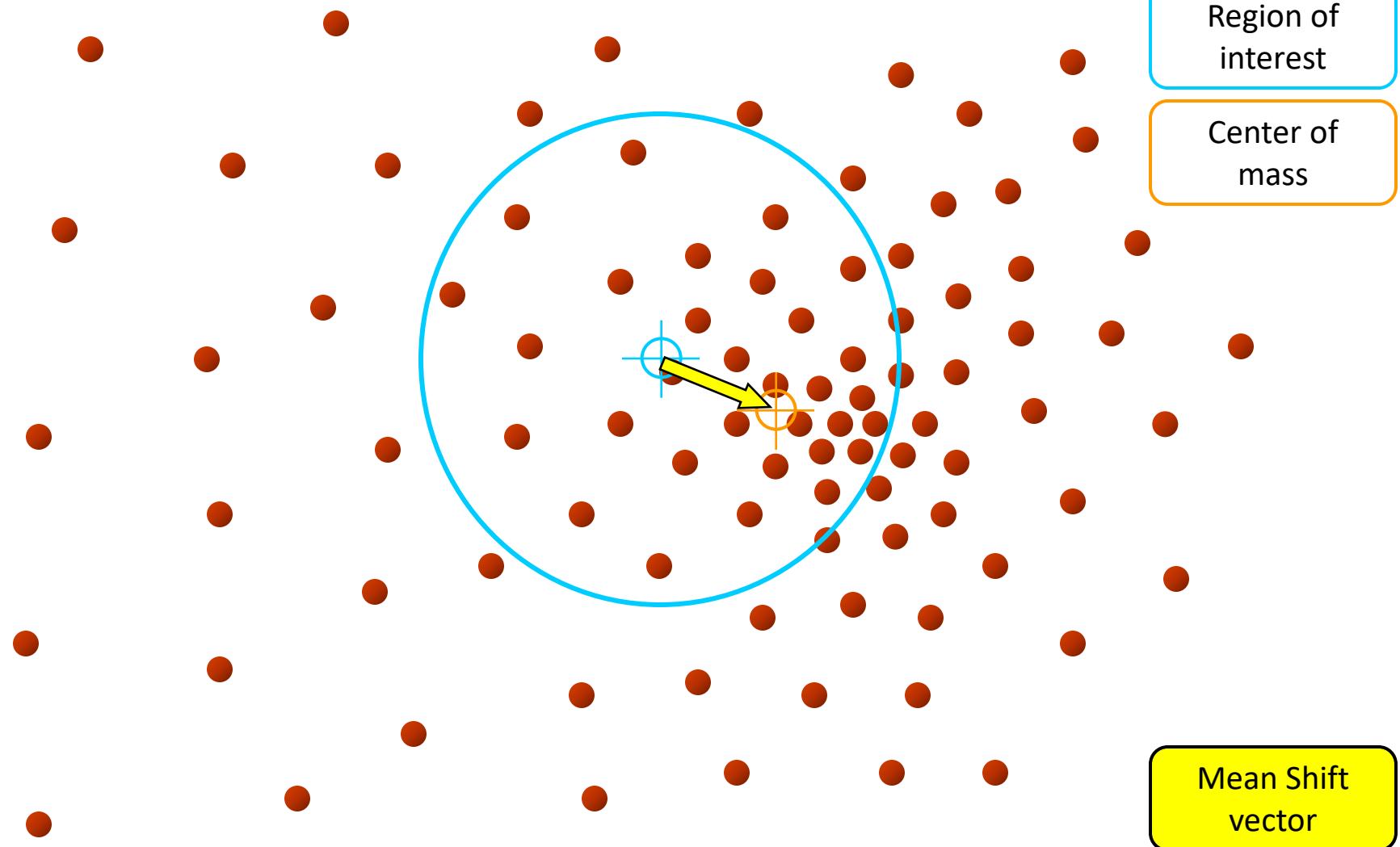


Mean shift



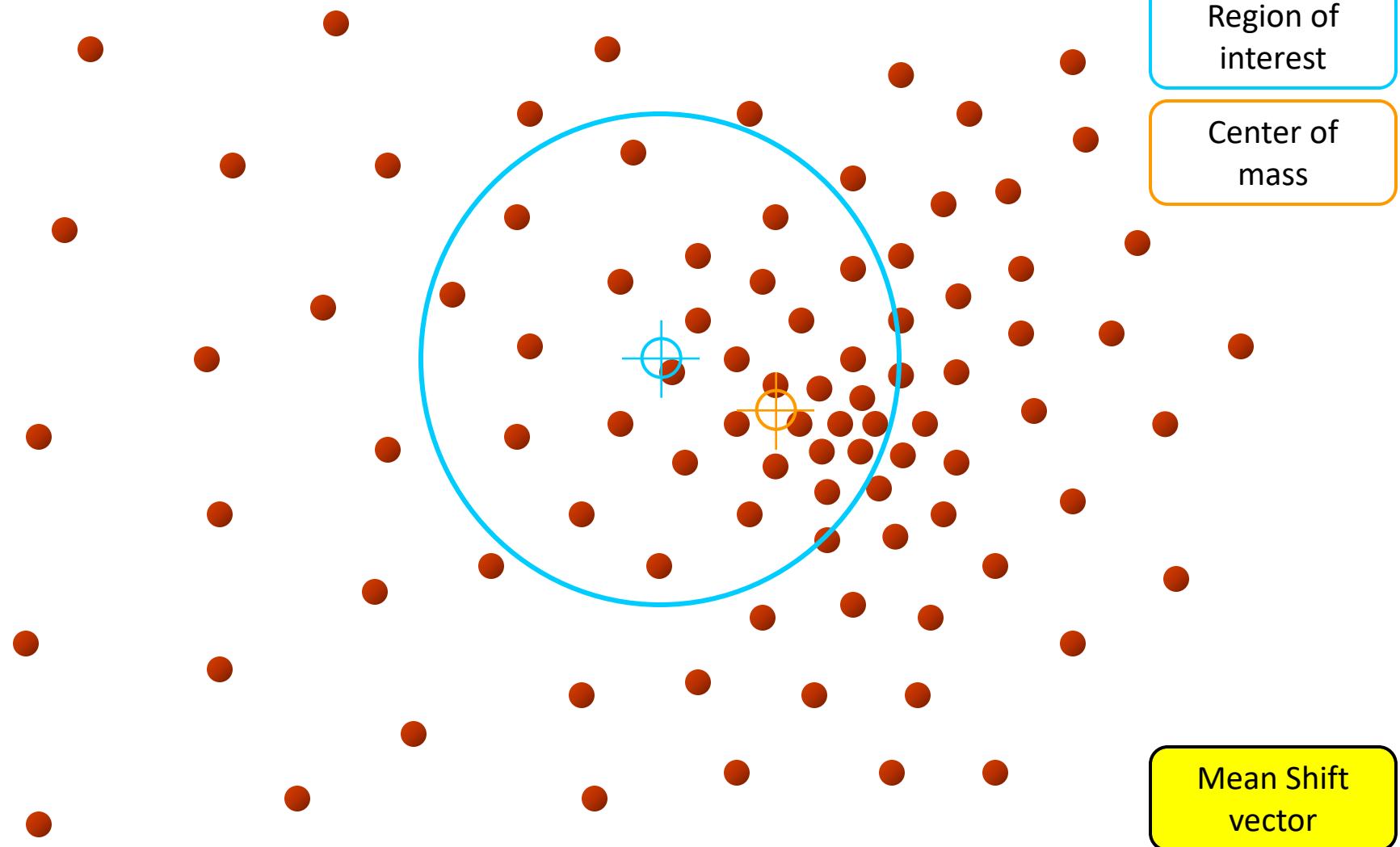


Mean shift



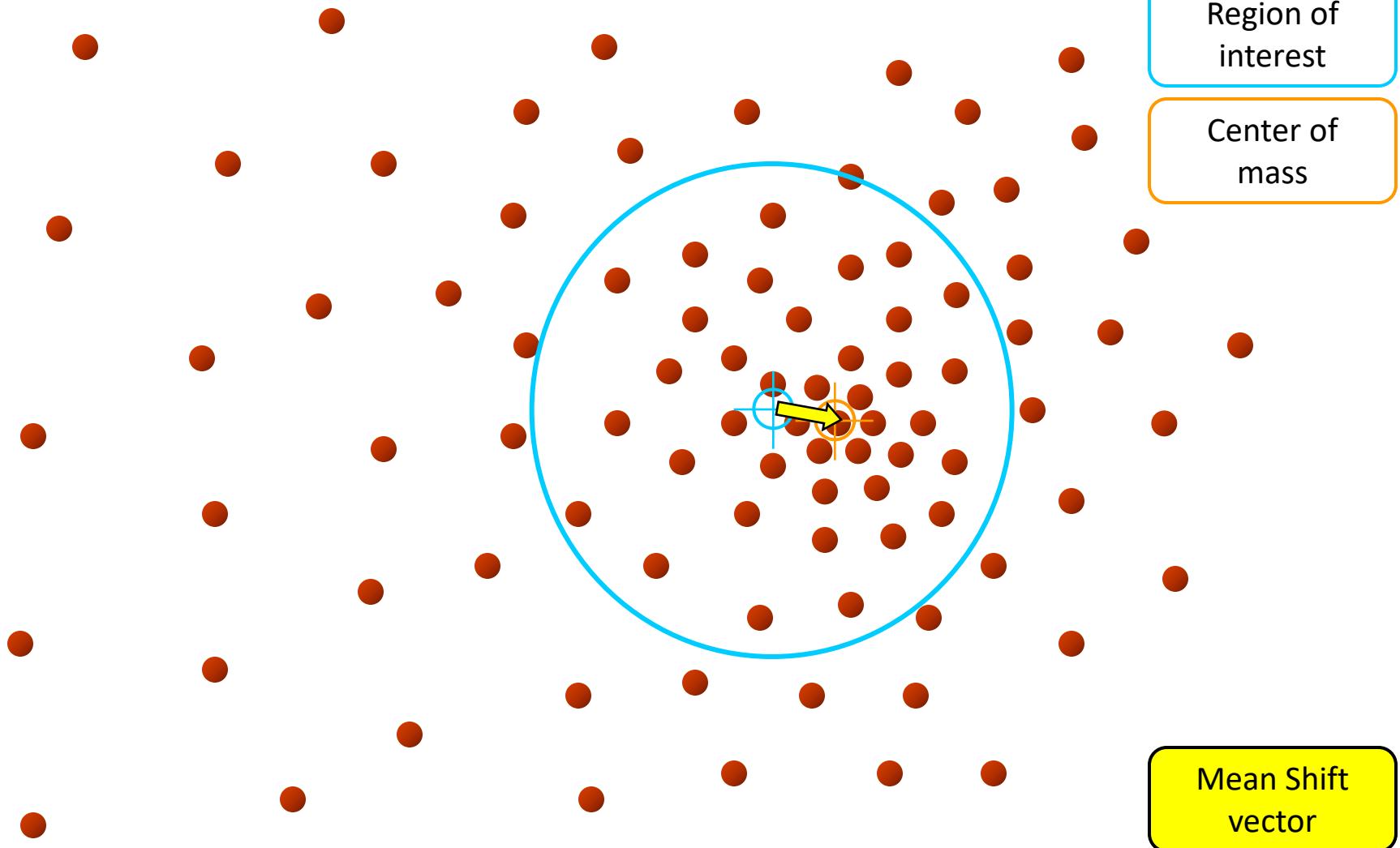


Mean shift



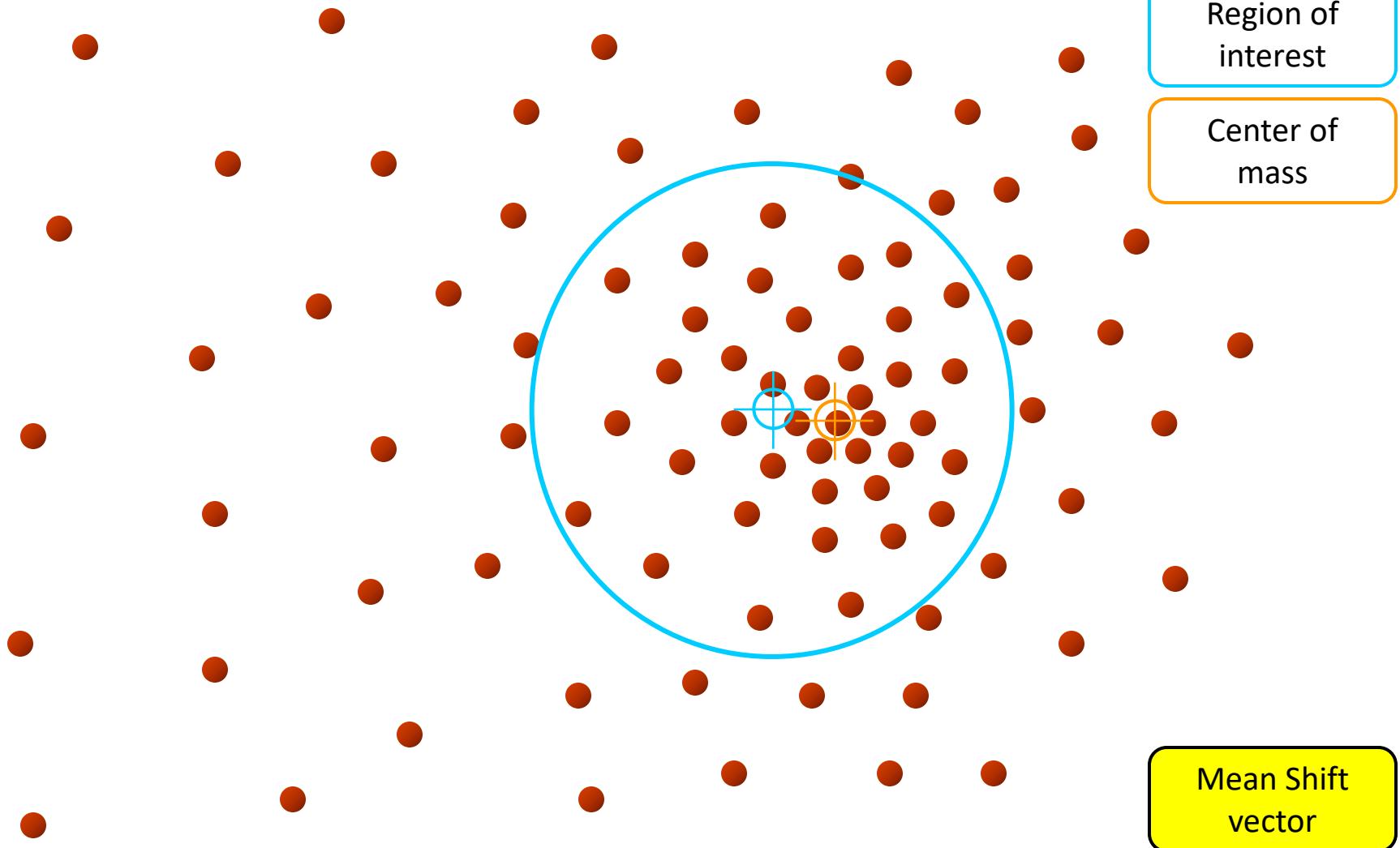


Mean shift



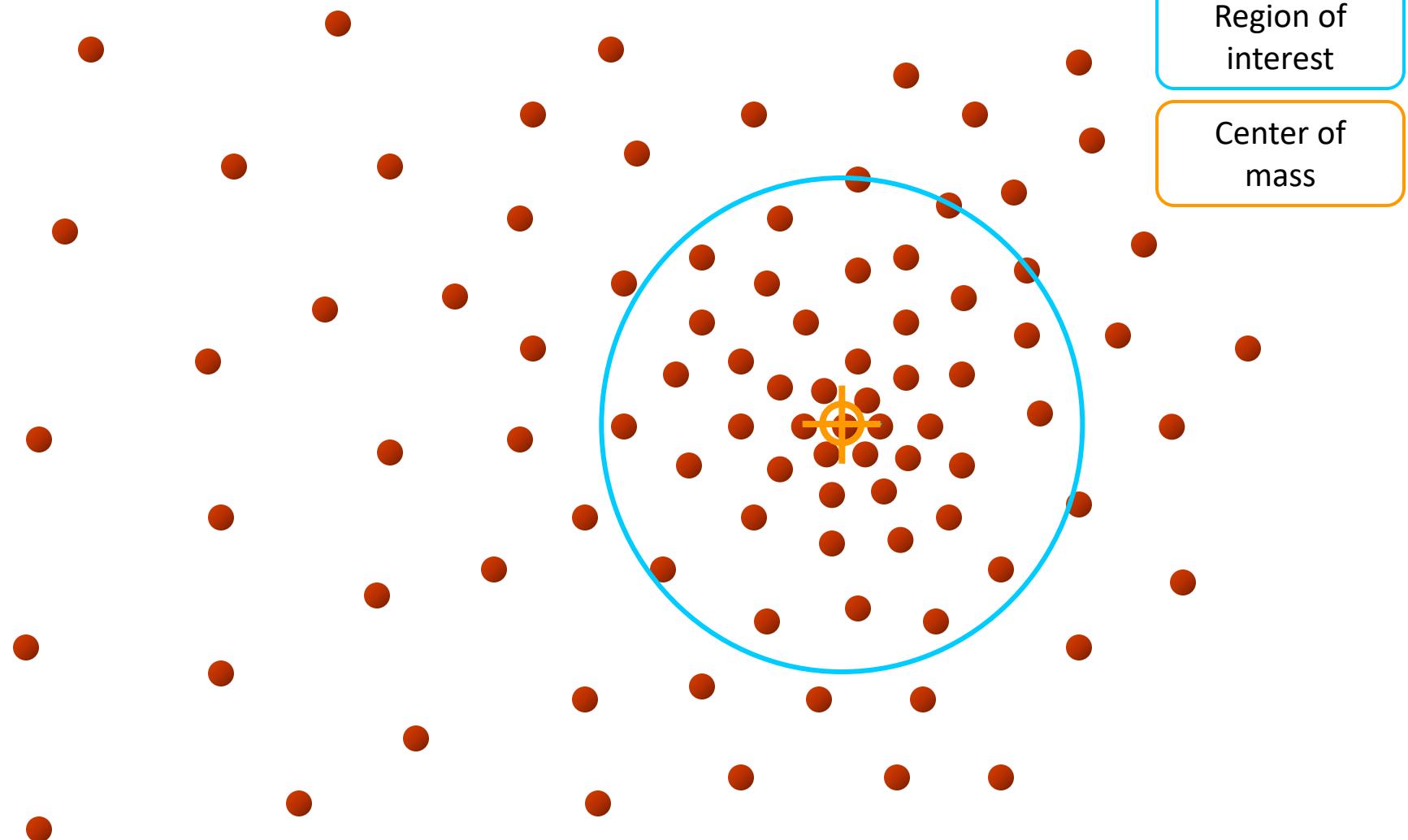


Mean shift





Mean shift



Region of
interest

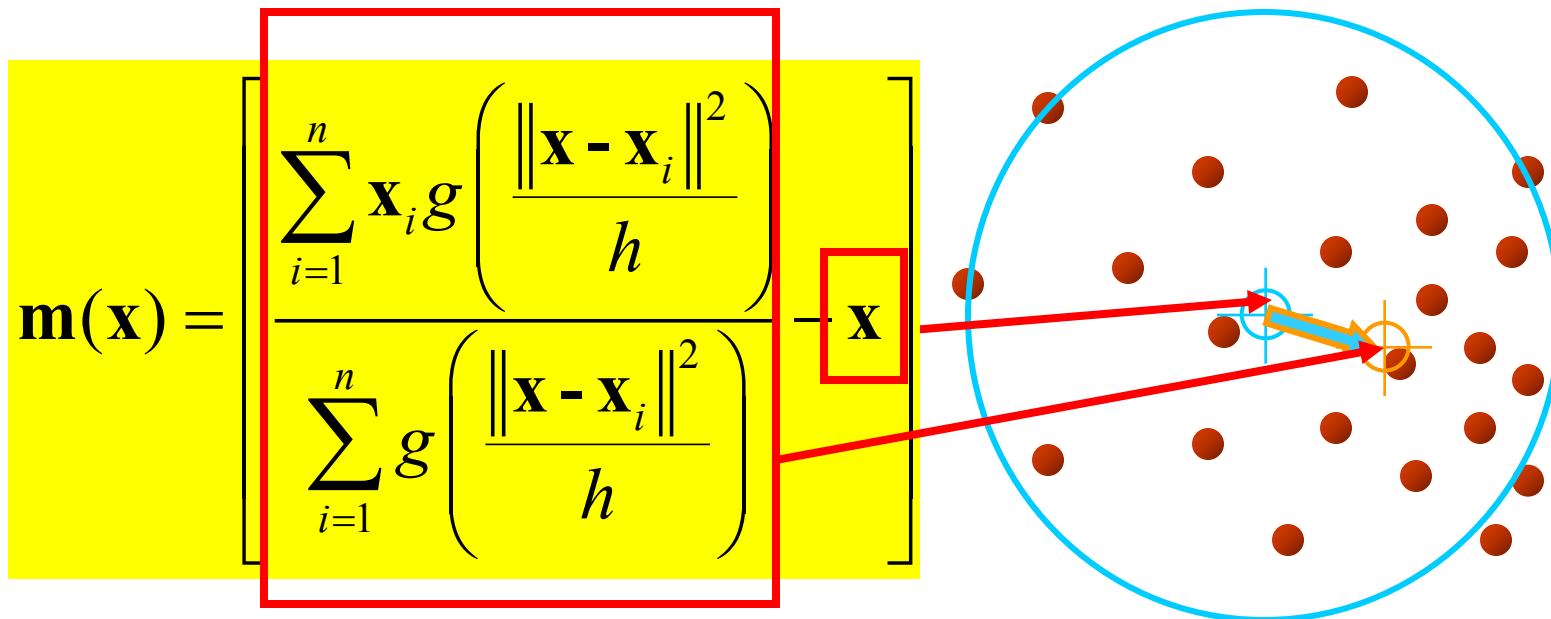
Center of
mass



Computing the Mean Shift

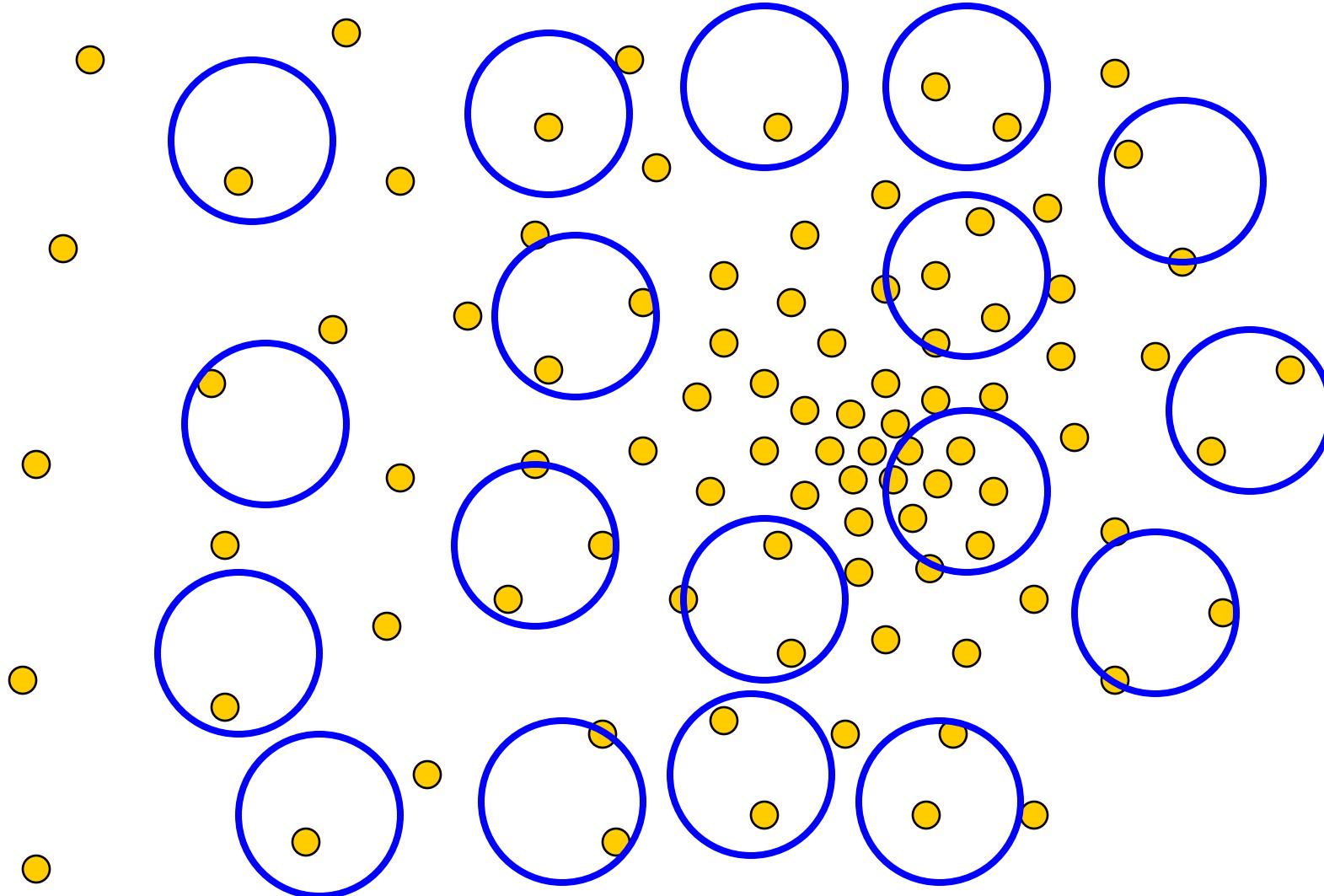
Simple Mean Shift procedure:

- Compute mean shift vector
- Translate the Kernel window by $\mathbf{m}(\mathbf{x})$

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




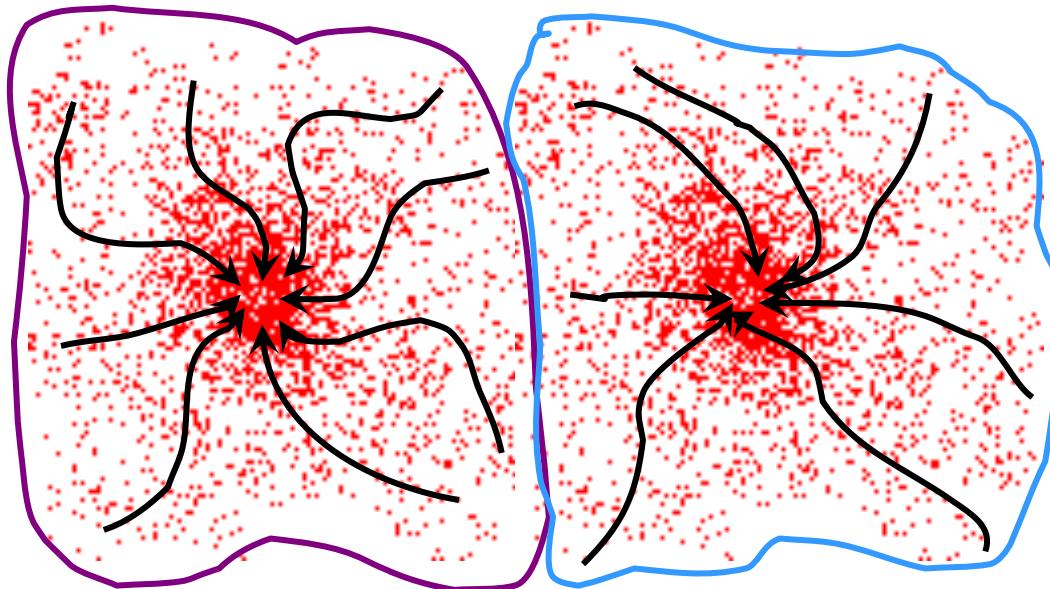
Real Modality Analysis





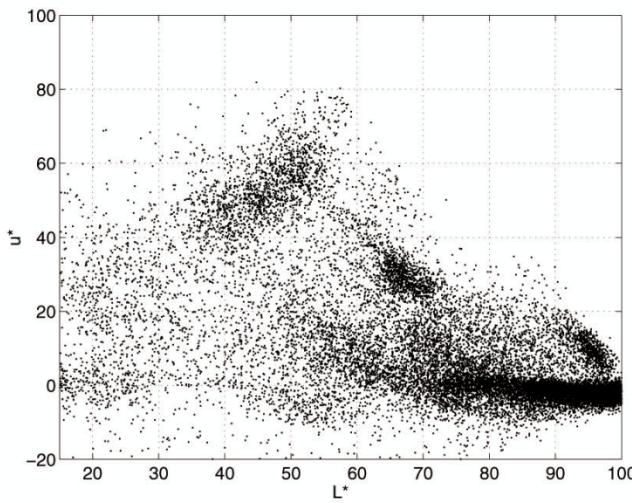
Attraction basin

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

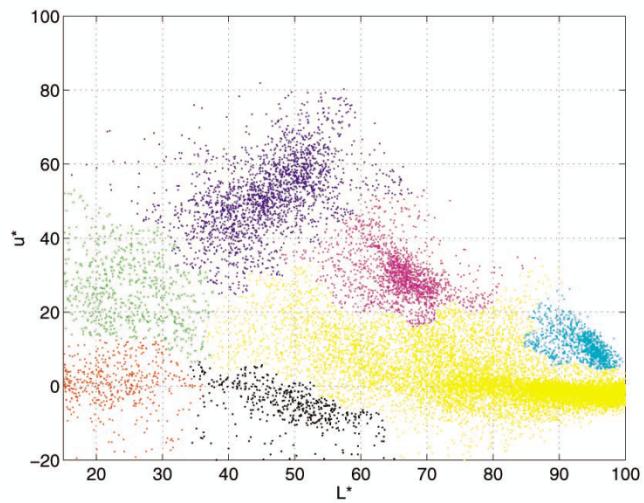




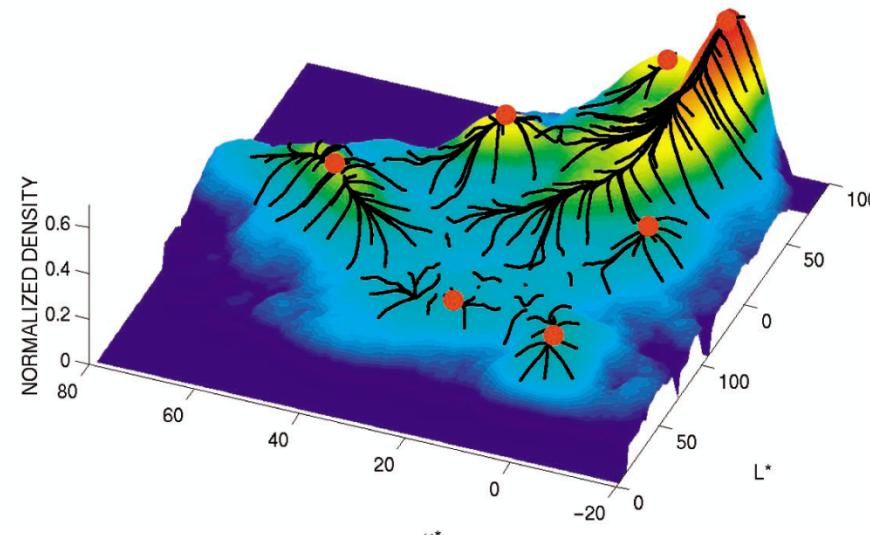
Attraction basin

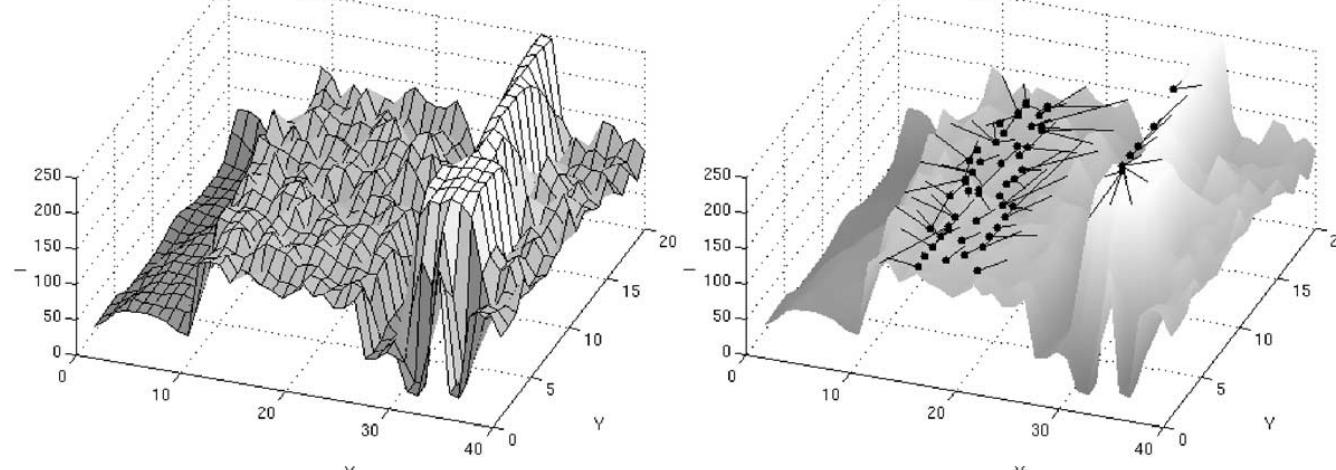


(a)

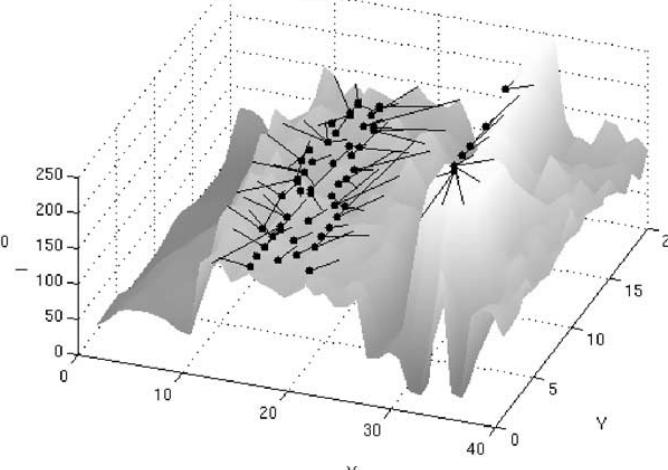


(b)

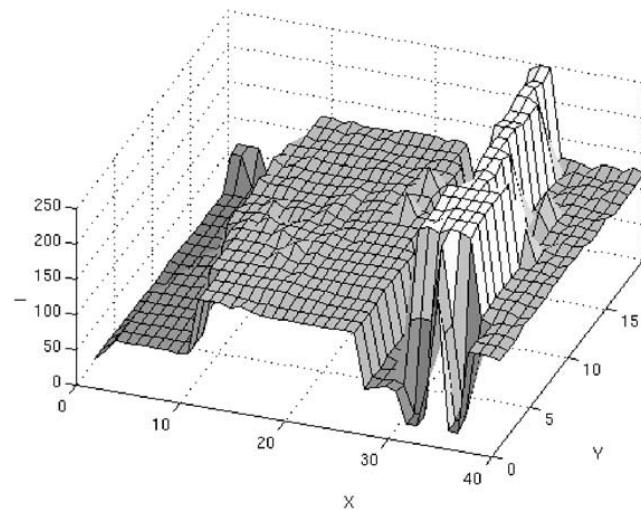




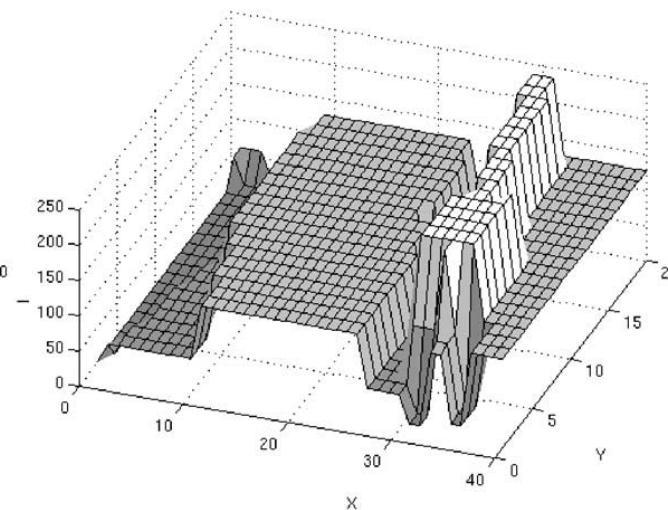
(a)



(b)



(c)



(d)

Mean shift filtering and segmentation for grayscale data; (a) input data (b) mean shift paths for the pixels on the plateaus (c) filtering result (d) segmentation result



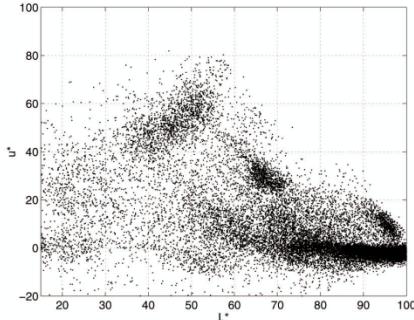
Mean shift clustering

- 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

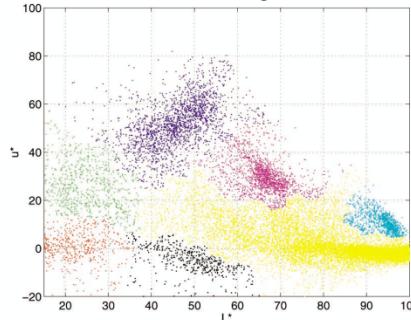


Segmentation by Mean Shift

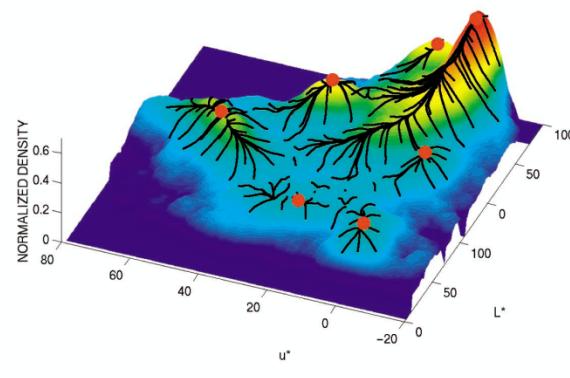
- Compute features for each pixel (color, gradients, texture, etc); also store each pixel's position
- Set kernel size for features K_f and position K_s
- Initialize windows at individual pixel locations
- Perform mean shift for each window until convergence
- Merge modes that are within width of K_f and K_s



(a)



(b)

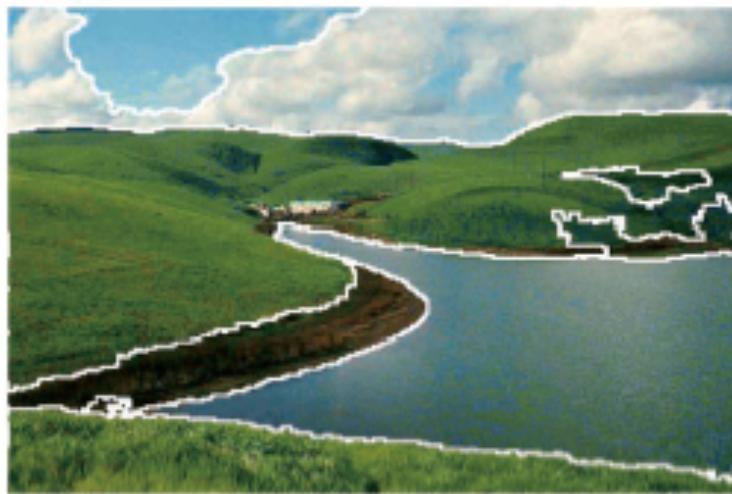


(c)



Mean shift segmentation results







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

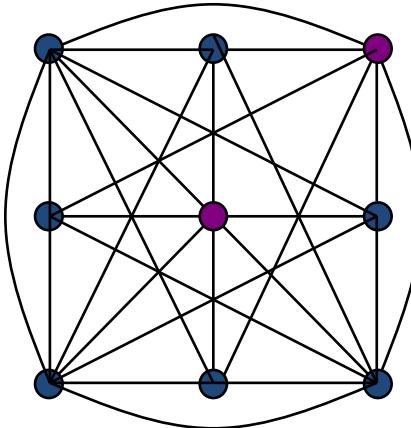


Further reading on mean shift

- 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 --- feel free to look at it but your code for segmentation will be different
- 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>



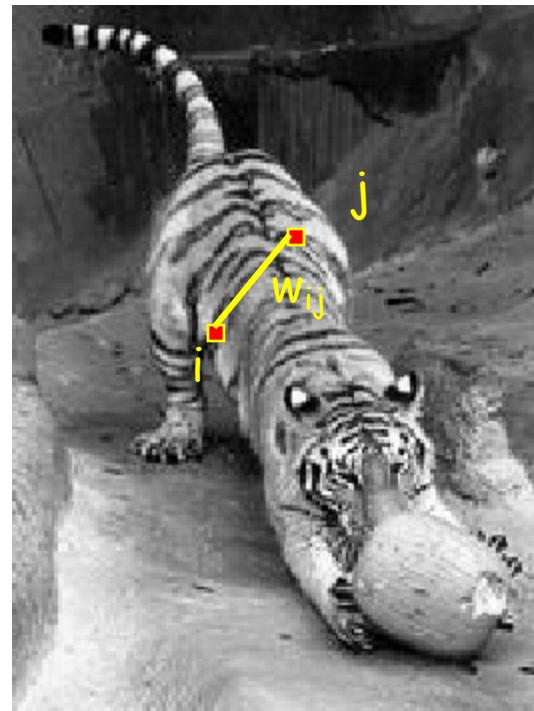
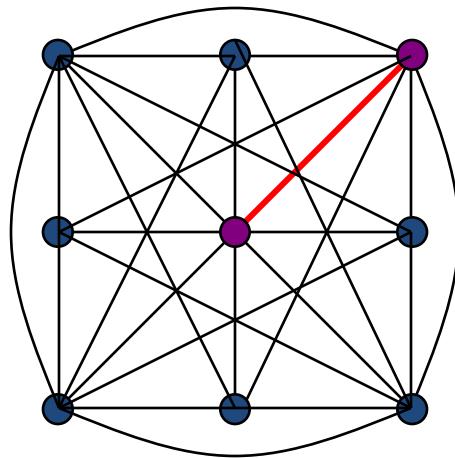
Segmentation as graph partitioning



- The set of points in an arbitrary feature space can be represented as a weighted undirected complete graph
 - $G = (V, E)$, where the nodes of the graph are the points in the feature space.
 - The weight w_{ij} of an edge $(i, j) \in E$ is a function of the similarity between the nodes and i and j .
- We can formulate image segmentation problem as a graph partitioning problem that asks for a partition of V_1, \dots, V_k of the vertex set V ; such that the vertices in V_i have high similarity and those in V_i, V_j have low similarity.



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



Measuring affinity

Property	Affinity function	Notes
Distance	$\exp \left\{ - \left((x - y)^t (x - y) / 2\sigma_d^2 \right) \right\}$	
Intensity	$\exp \left\{ - \left((I(x) - I(y))^t (I(x) - I(y)) / 2\sigma_I^2 \right) \right\}$	$I(x)$ is the intensity of the pixel at x .
Color	$\exp \left\{ - \left(\text{dist}(c(x), c(y))^2 / 2\sigma_c^2 \right) \right\}$	$c(x)$ is the color of the pixel at x .
Texture	$\exp \left\{ - \left((f(x) - f(y))^t (f(x) - f(y)) / 2\sigma_f^2 \right) \right\}$	$f(x)$ is a vector of filter outputs describing the pixel at x computed as in Section 6.1.

TABLE 9.1: Different affinity functions comparing pixels for a graph based segmenter. Notice that affinities can be combined. One attractive feature of the exponential form is that, say, location, intensity and texture affinities could be combined by multiplying them.

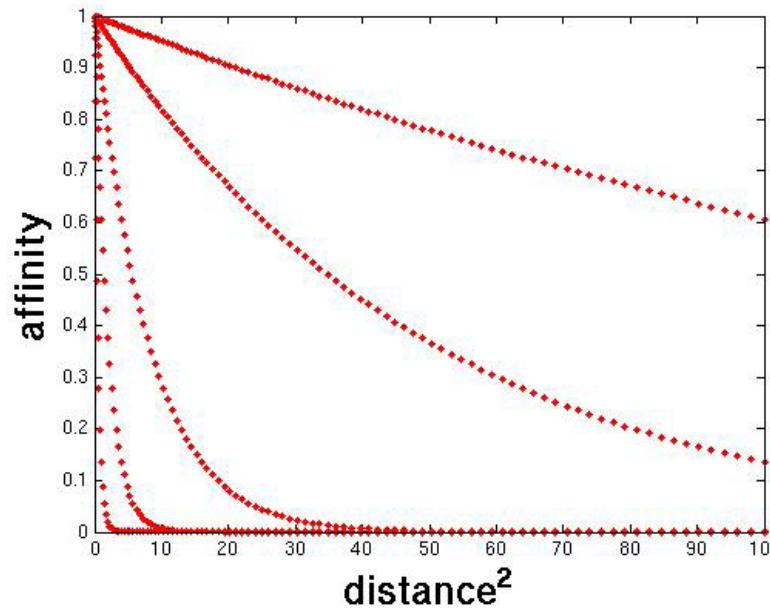


Measuring affinity

- Represent each pixel by a feature vector \mathbf{x} and define an appropriate distance function

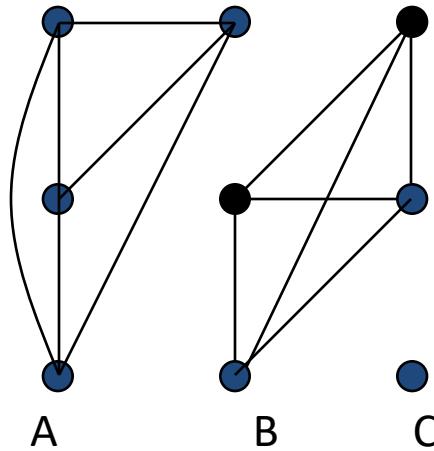
$$\text{affinity}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2\sigma^2} \text{dist}(\mathbf{x}_i, \mathbf{x}_j)^2\right)$$

Role of σ





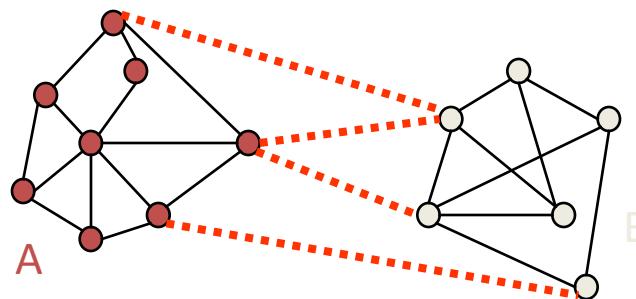
Segmentation as graph partitioning



- 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



General: Graph cut

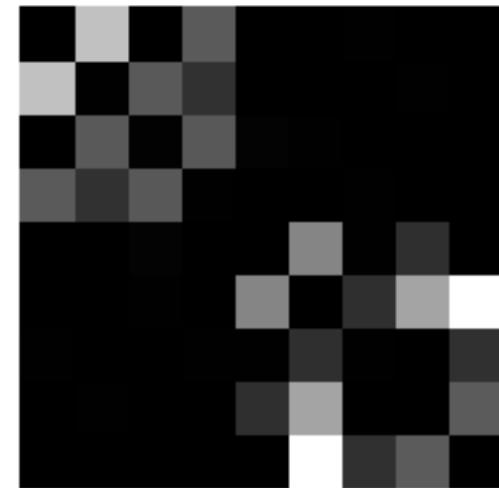
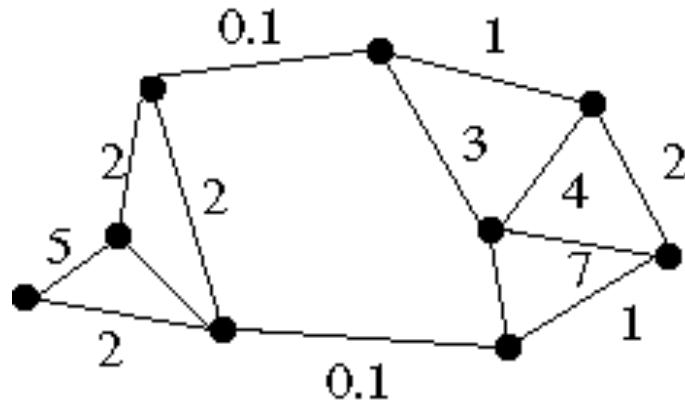


- Set of edges whose removal makes a graph disconnected
- Cost of a cut: sum of weights of cut edges
- A graph cut gives us a segmentation
 - What is a “good” graph cut and how do we find one?



Minimum cut

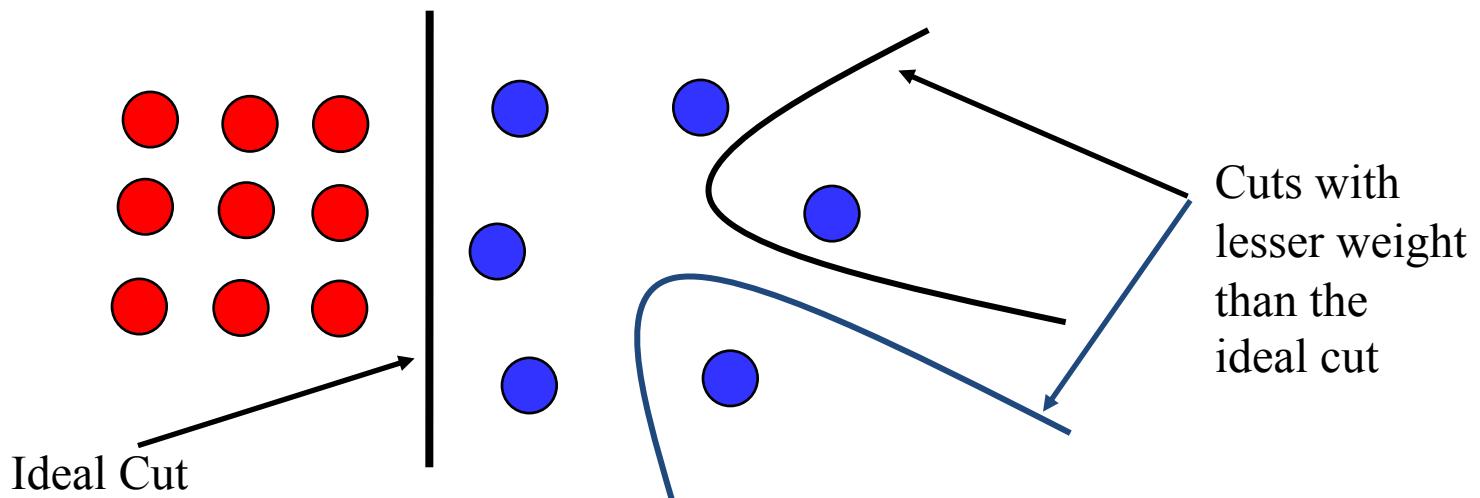
- We can do segmentation by finding the *minimum cut* in a graph
 - Efficient algorithms exist for doing this





Normalized cut

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





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:

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

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

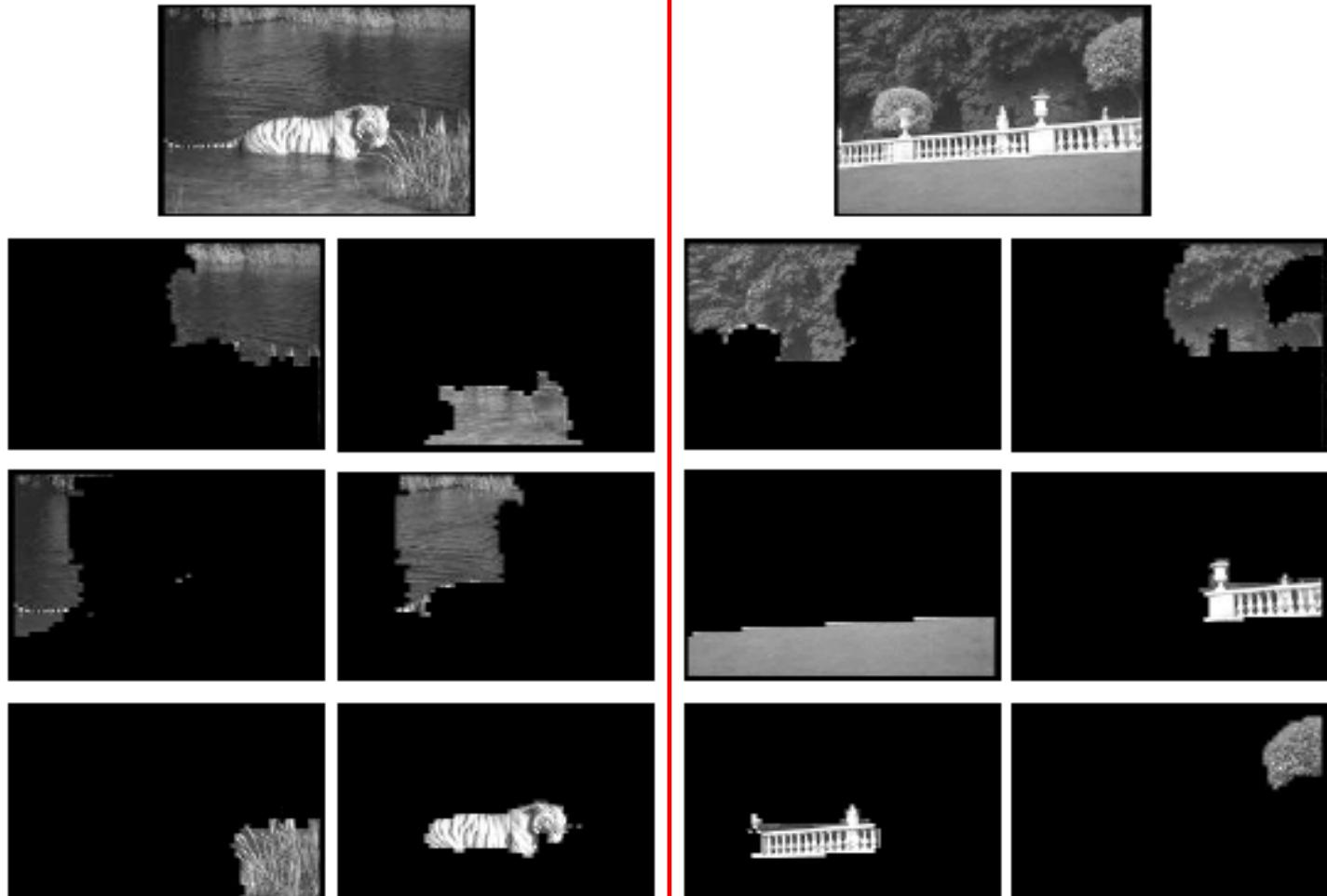


FIGURE 9.24: The images on top are segmented using the normalized cuts framework, described in the text, into the components shown. The affinity measures used involved intensity and texture, as in Table 9.1. The image of the swimming tiger yields one segment that is essentially tiger, one that is grass, and four components corresponding to the lake. Similarly, the railing shows as three reasonably coherent segments. Note the improvement over k-means segmentation obtained by having a texture measure. *This figure was originally published as Figure 2 of “Image and video segmentation: the normalized cut framework,” by J. Shi, S. Belongie, T. Leung, and J. Malik, Proc. IEEE Int. Conf. Image Processing, 1998 © IEEE, 1998.*



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 $n \times n$, where n is the number of pixels



Segmentation as labeling

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





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?



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





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



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

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

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

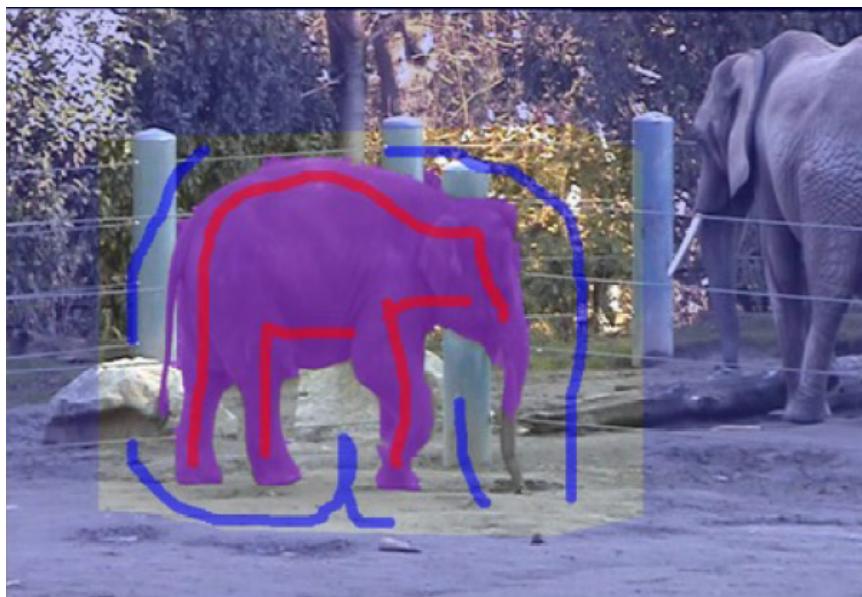
$C'(x, y, 0)$: “distance” from pixel to background

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

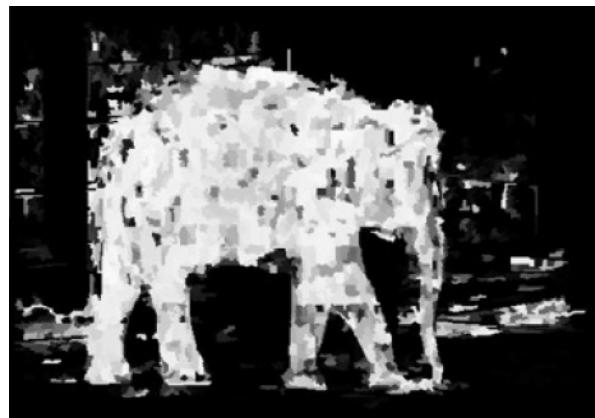
} computed by
creating a *color
model* from user-
labeled pixels



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



$$C'(x, y, 0)$$



$$C'(x, y, 1)$$



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

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



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

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

$$w_{pq} = 0.1$$
$$w_{pq} = 10.0$$



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



Efficient graph-based segmentation

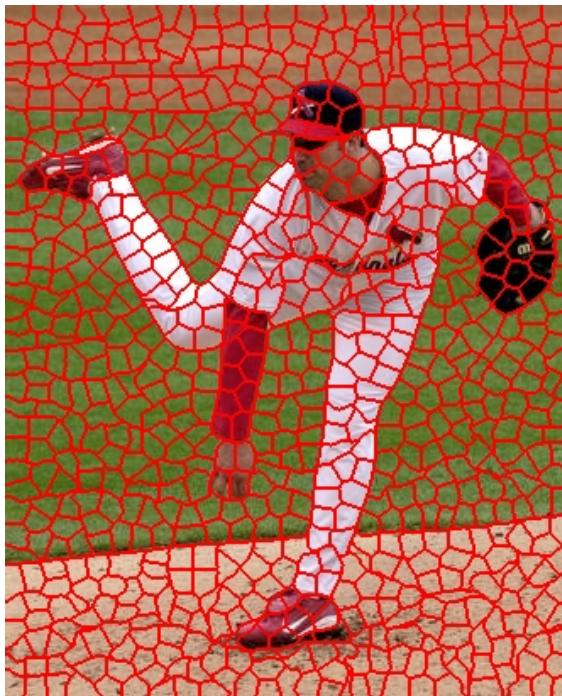




Superpixels

- 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.
CSCI 631 – Lecture 7



Felzenszwab and Huttenlocher: Graph-Based Segmentation

<http://www.cs.brown.edu/~pff/segment/>



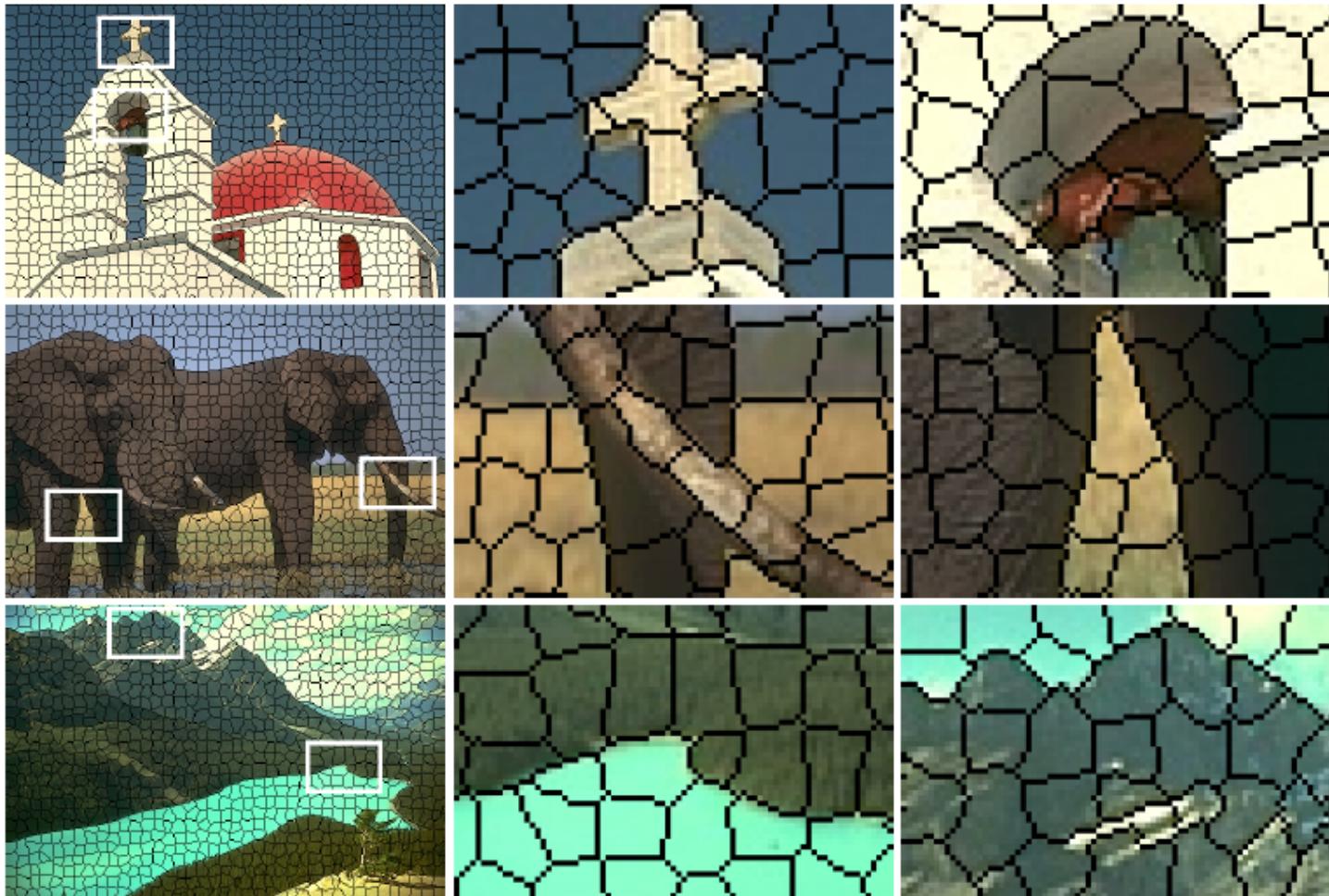
- + Good for thin regions
- + Fast, runs in time nearly linear in the number of edges
- + Easy to control coarseness of segmentations
- + Can include both large and small regions
- Often creates regions with strange shapes
- Sometimes makes very large errors



Turbo Pixels: Levinstein et al. 2009

<http://www.cs.toronto.edu/~kyros/pubs/09.pami.turbopixels.pdf>

Tries to preserve boundaries and produces more regular regions





Applications of segmentation

- Shot boundary detection
 - summarize videos by
 - finding shot boundaries
 - obtaining “most representative” frame
- Background subtraction
 - find “interesting bits” of image by subtracting known background
 - e.g. sports videos
 - e.g. find person in an office
 - e.g. find cars on a road
- Interactive segmentation
 - user marks some foreground/background pixels
 - system cuts object out of image
 - useful for image editing, etc.



Final thoughts

- Segmentation is a difficult topic with a huge variety of implementations.
 - It is typically hard to assess the performance of a segmenter at a level more useful than that of showing some examples
- There really is not much theory available to predict what should be clustered and how.
- Everyone should know about some clustering techniques like k-means, mean shift, and at least one graph-based clustering algorithm
 - these ideas are just so useful for so many applications
 - segmentation is just one application of clustering.



Slide Credits

- Svetlana Lazebnik – UIUC
- Derek Hoiem – UIUC
- David Forsyth - UIUC



Questions

