



# *Lecture 7: Segmentation and Clustering, Part I*

**Pattern Recognition and Computer Vision**

Guanbin Li,

School of Computer Science and Engineering, Sun Yat-Sen University

# 扫码签到

---



# CS 131 Roadmap

Pixels	<b>Segments</b>	Images	Videos	Web
Convolutions Edges Descriptors	<b>Segmentation</b> <b>Clustering</b> Resizing	Recognition Detection Machine learning	Motion Tracking	Neural networks Convolutional neural networks

# What we will learn today

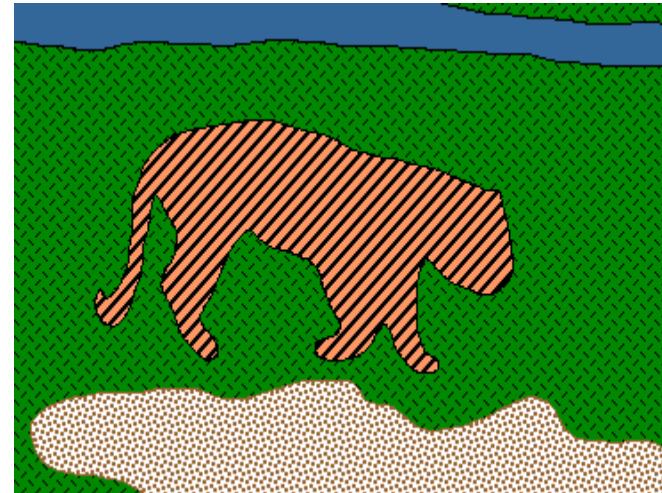
---

- Introduction to segmentation and clustering
- Gestalt theory for perceptual grouping
- Agglomerative clustering
- Graph-based segmentation

# Image Segmentation

---

- Goal: identify groups of pixels that go together



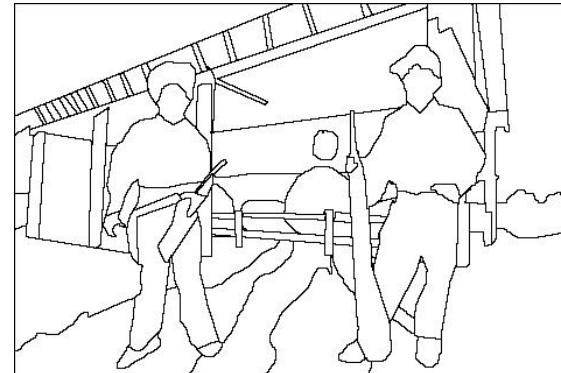
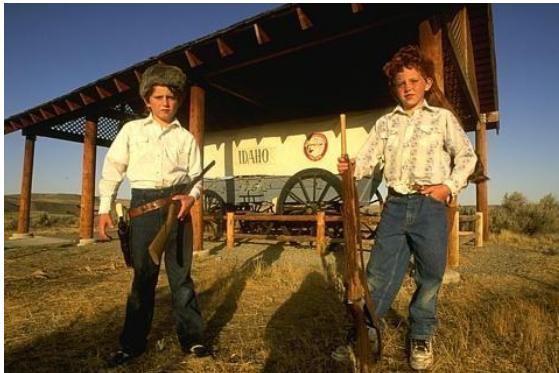
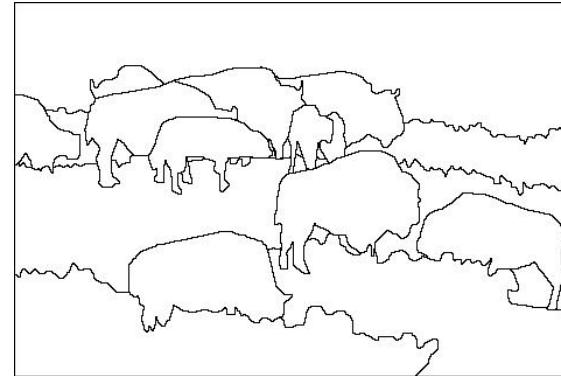
# The Goals of Segmentation

- Separate image into coherent “objects”

Image



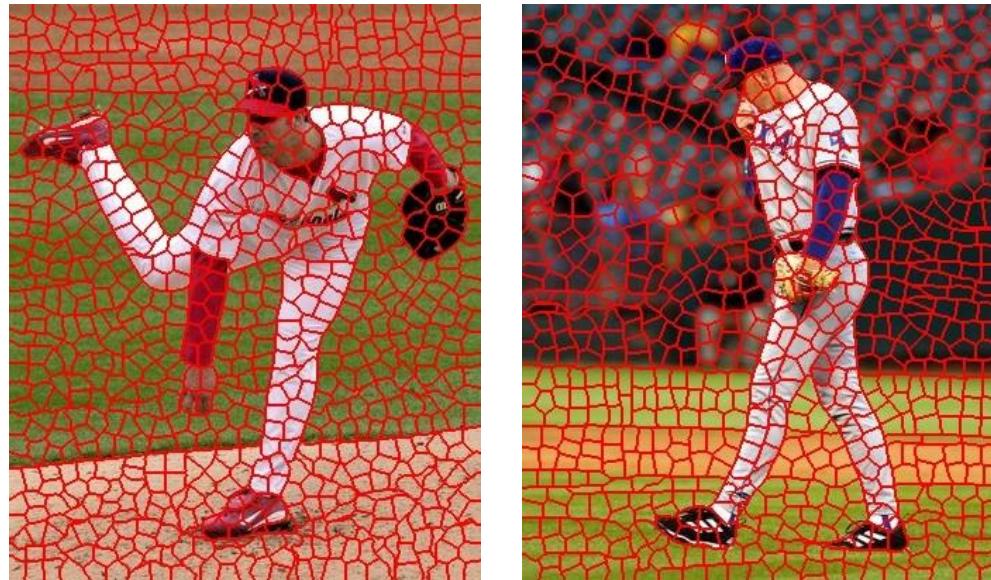
Human segmentation



# The Goals of Segmentation

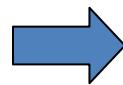
- Separate image into coherent “objects”
- Group together similar-looking pixels for efficiency of further processing

“superpixels”

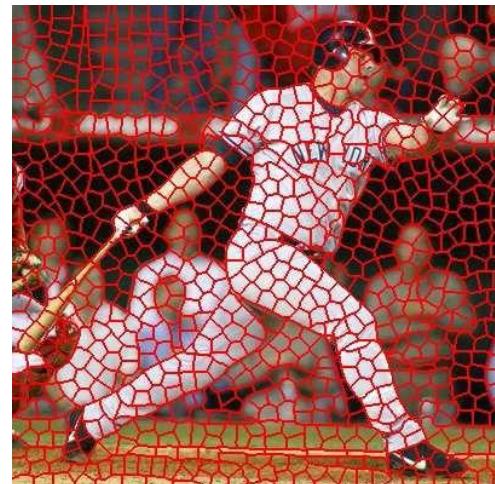
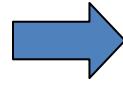


X. Ren and J. Malik. [Learning a classification model for segmentation](#). ICCV 2003.

# Segmentation for efficiency



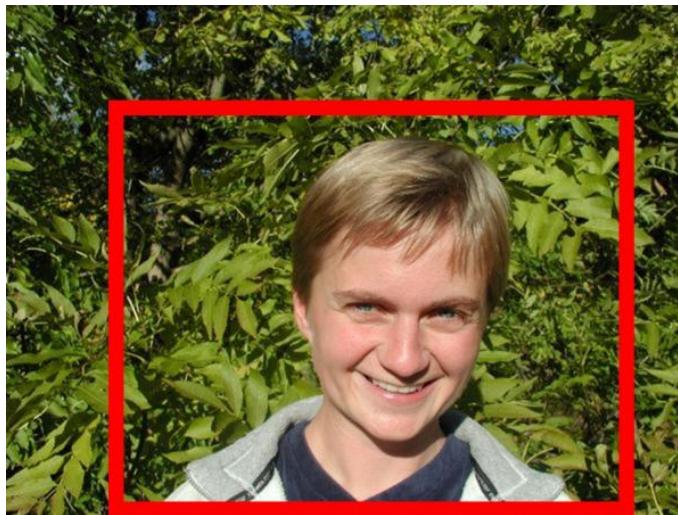
[Felzenszwalb and Huttenlocher 2004]



[Shi and Malik 2001]

[Hoiem et al. 2005, Mori 2005]

# Segmentation as a result



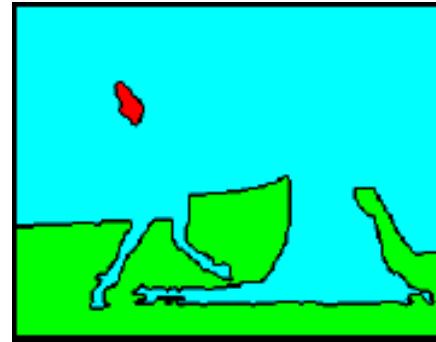
Rother et al. 2004

# Types of segmentations

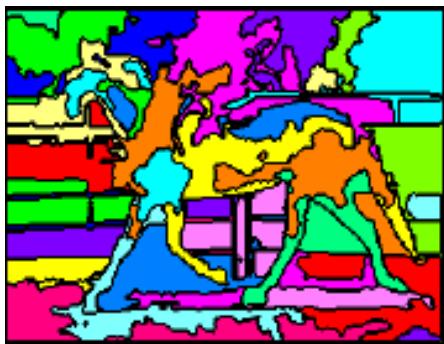
---



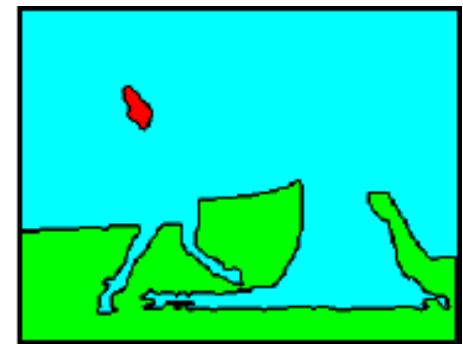
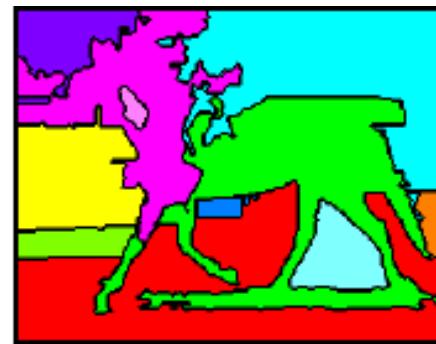
Oversegmentation



Undersegmentation



Multiple Segmentations



# One way to think about “segmentation” is Clustering

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

Key Challenges:

- 1) What makes two points/images/patches similar?
- 2) How do we compute an overall grouping from pairwise similarities?

# Why do we cluster?

---

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

# How do we cluster?

---

- **Agglomerative clustering**
  - Start with each point as its own cluster and iteratively merge the closest clusters
- **K-means (next lecture)**
  - Iteratively re-assign points to the nearest cluster center
- **Mean-shift clustering (next lecture)**
  - Estimate modes of pdf

# General ideas

---

- **Tokens**
    - whatever we need to group (pixels, points, surface elements, etc.)
  - **Bottom up clustering**
    - tokens belong together because they are locally coherent
  - **Top down clustering**
    - tokens belong together because they lie on the same visual entity (object, scene...)
- > These two are not mutually exclusive

# Examples of Grouping in Vision



Determining image regions



Grouping video frames into shots

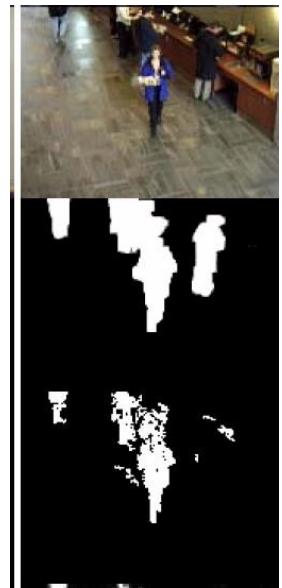
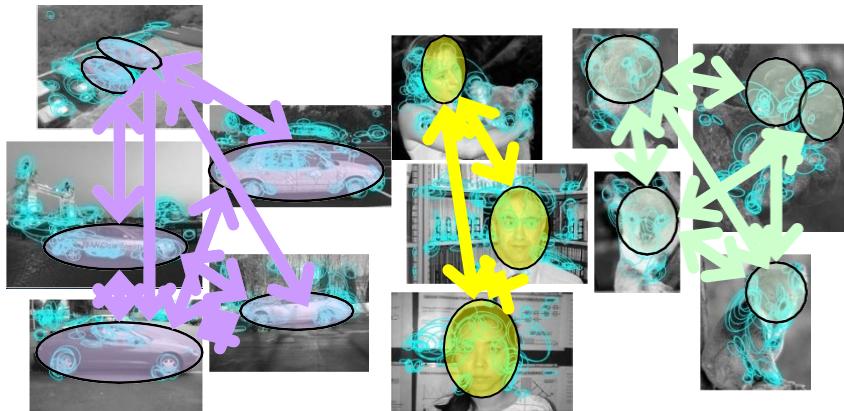


Figure-ground

*What things should  
be grouped?*

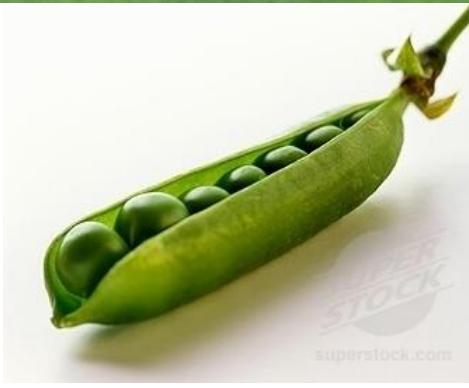
*What cues  
indicate groups?*



Object-level grouping

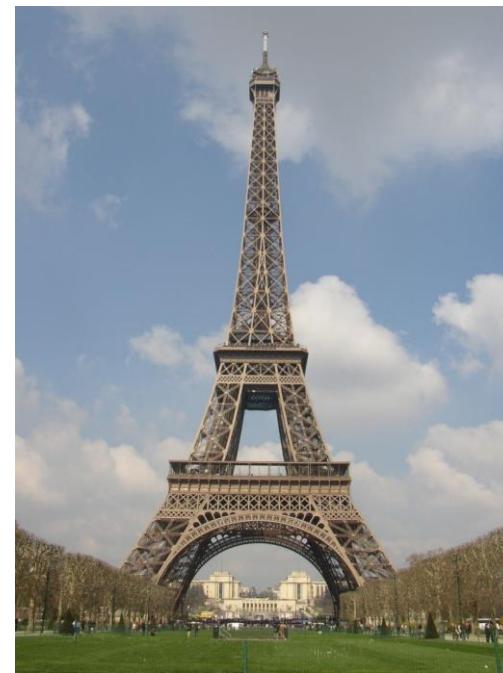
# Similarity

---



# Symmetry

---



# Common Fate

---



Image credit: Arthus-Bertrand (via F. Durand)



© 2005 Heiko Burkhardt, illano.com

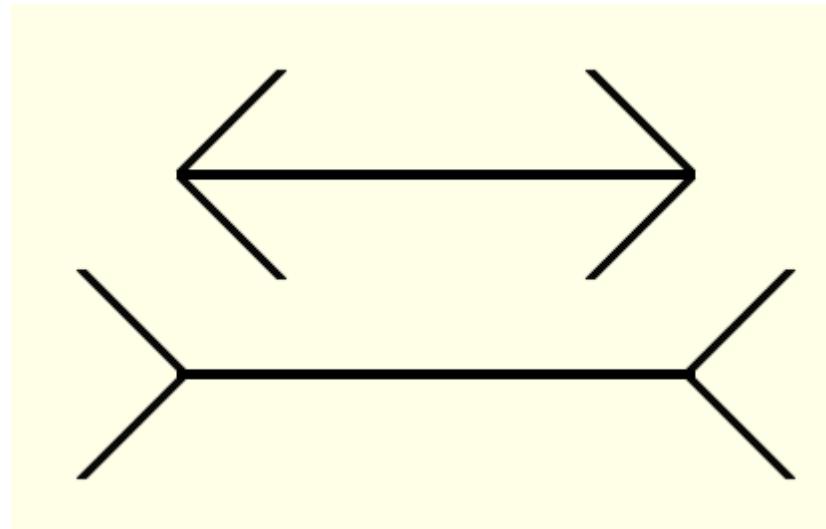
# Proximity

---



# Muller-Lyer Illusion

---



- What makes the bottom line look longer than the top line?

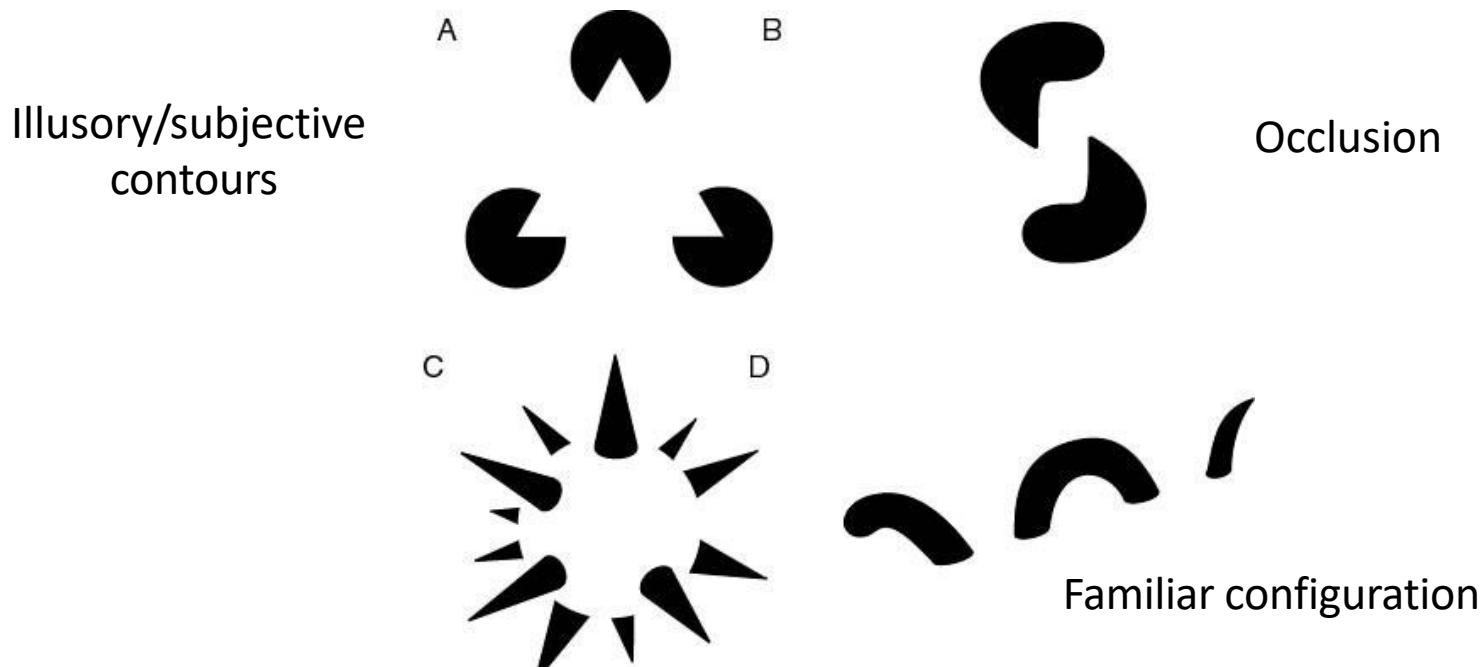
# What we will learn today

---

- Introduction to segmentation and clustering
- Gestalt theory for perceptual grouping
- Agglomerative clustering
- Graph-based segmentation

# The Gestalt School

- Grouping is key to visual perception
- Elements in a collection can have properties that result from **relationships**
  - “The whole is greater than the sum of its parts”



[http://en.wikipedia.org/wiki/Gestalt\\_psychology](http://en.wikipedia.org/wiki/Gestalt_psychology)

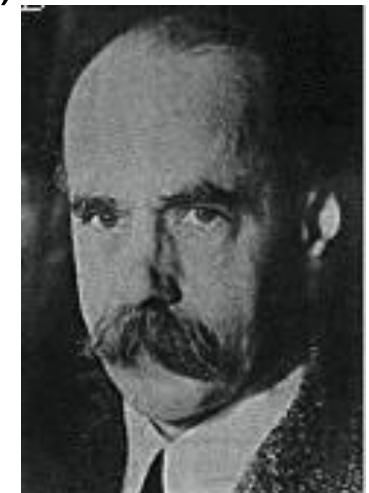
# Gestalt Theory

---

- Gestalt: whole or group
  - Whole is greater than sum of its parts
  - Relationships among parts can yield new properties/features
- Psychologists identified series of factors that predispose set of elements to be grouped (by human visual system)

*"I stand at the window and see a house, trees, sky.  
Theoretically I might say there were 327 brightnesses  
and nuances of colour. Do I have "327"? No. I have sky, house, and  
trees."*

Max Wertheimer  
(1880-1943)



Untersuchungen zur Lehre von der Gestalt,  
*Psychologische Forschung*, Vol. 4, pp. 301-350, 1923  
<http://psy.ed.asu.edu/~classics/Wertheimer/Forms/forms.htm>

# Gestalt Factors



Not grouped



Proximity



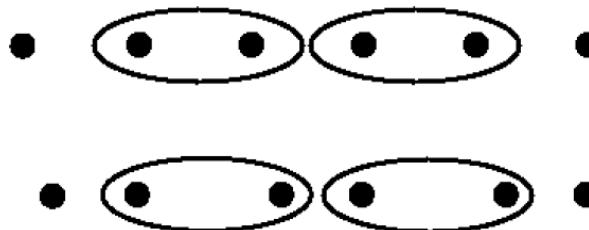
Similarity



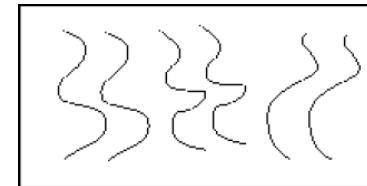
Similarity



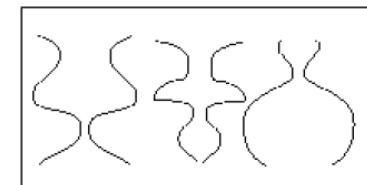
Common Fate



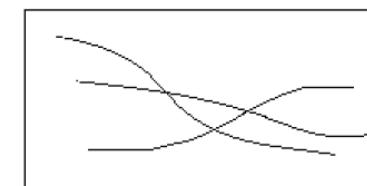
Common Region



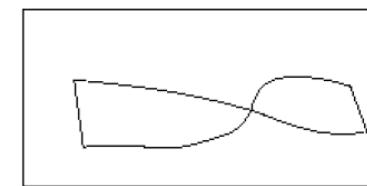
Parallelism



Symmetry



Continuity

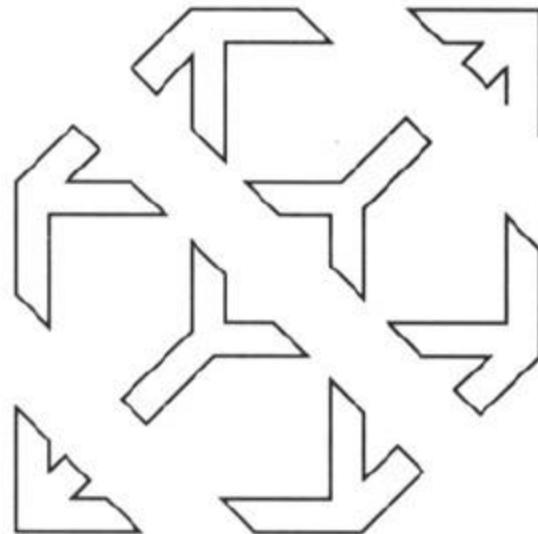


Closure

- These factors make intuitive sense, but are very difficult to translate into algorithms.

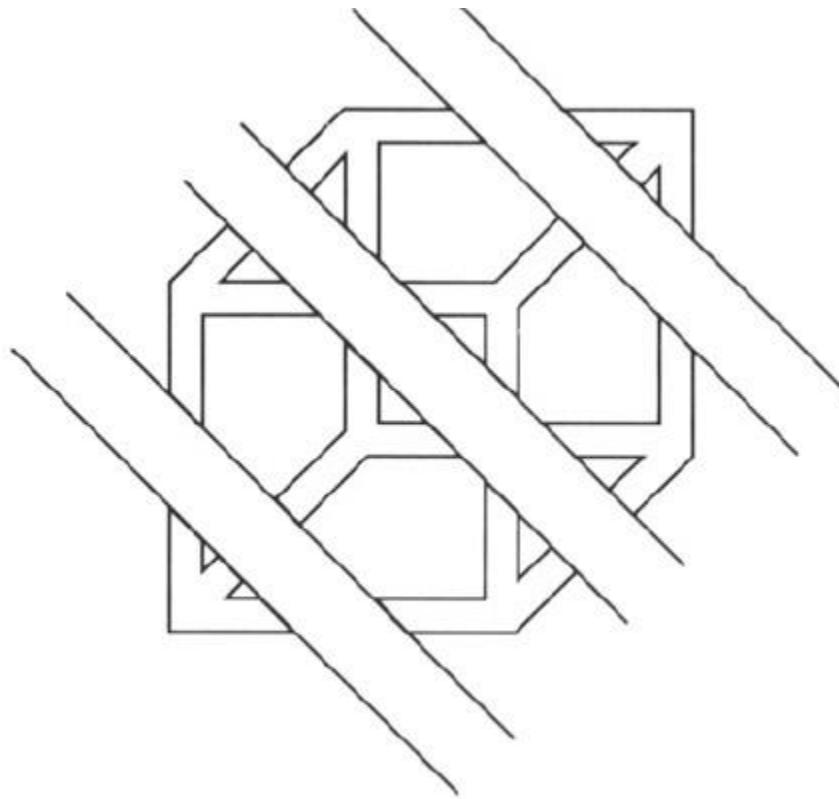
# Continuity through Occlusion Cues

---



# Continuity through Occlusion Cues

---



Continuity, explanation by occlusion

# Continuity through Occlusion Cues

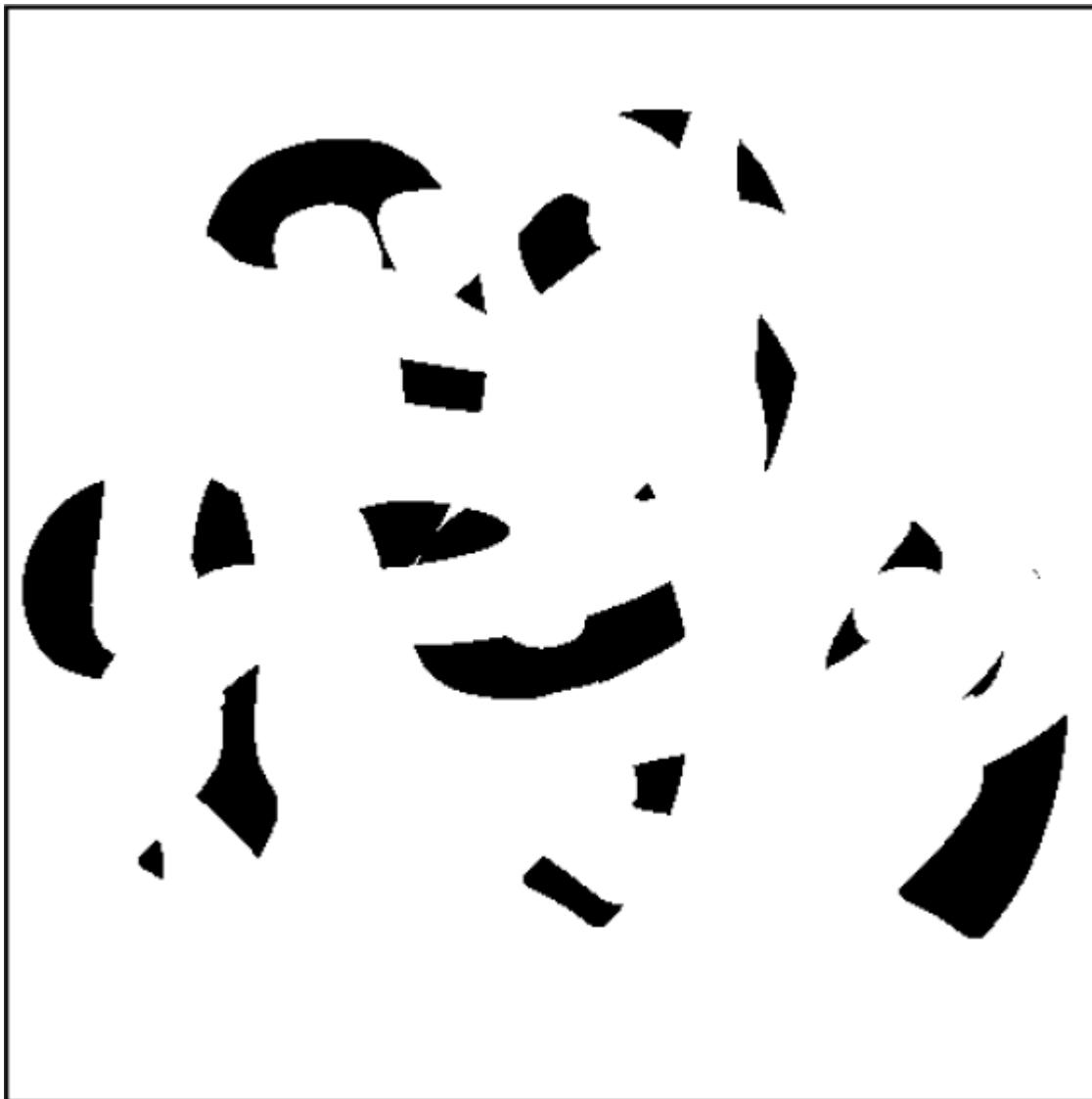


Image source: Forsyth & Ponce

# Continuity through Occlusion Cues

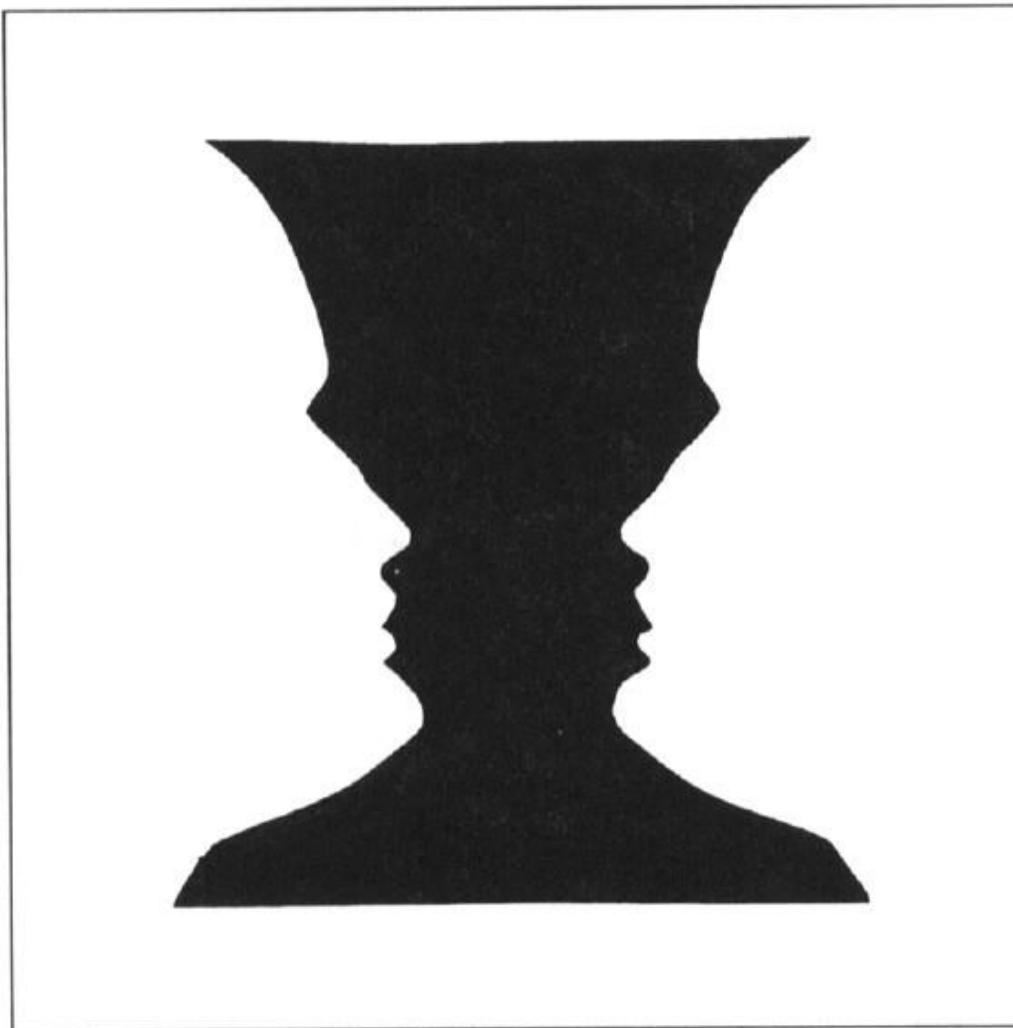
---



Image source: Forsyth & Ponce

# Figure-Ground Discrimination

---



# The Ultimate Gestalt?

---



# What we will learn today

---

- Introduction to segmentation and clustering
- Gestalt theory for perceptual grouping
- **Agglomerative clustering**
- Graph-based segmentation

# Clustering: distance measure

---

Clustering is an unsupervised learning method. Given items  $x_1, \dots, x_n \in \mathbb{R}^D$ , the goal is to group them into clusters. We need a pairwise distance/similarity function between items, and sometimes the desired number of clusters.

When data (e.g. images, objects, documents) are represented by feature vectors, commonly used measures are:

- *Euclidean distance.*
- *cosine similarity.*

# Defining Distance Measures

---

Let  $x$  and  $x'$  be two objects from the universe of possible objects. The distance (similarity) between  $x$  and  $x'$  is a real number denoted by  $\text{sim}(x, x')$ .

- The Euclidean distance is defined as

$$\text{dist}(x, x') = \sqrt{\sum (x_i - x'_i)^2}$$

- In contrast, the cosine similarity measure would be

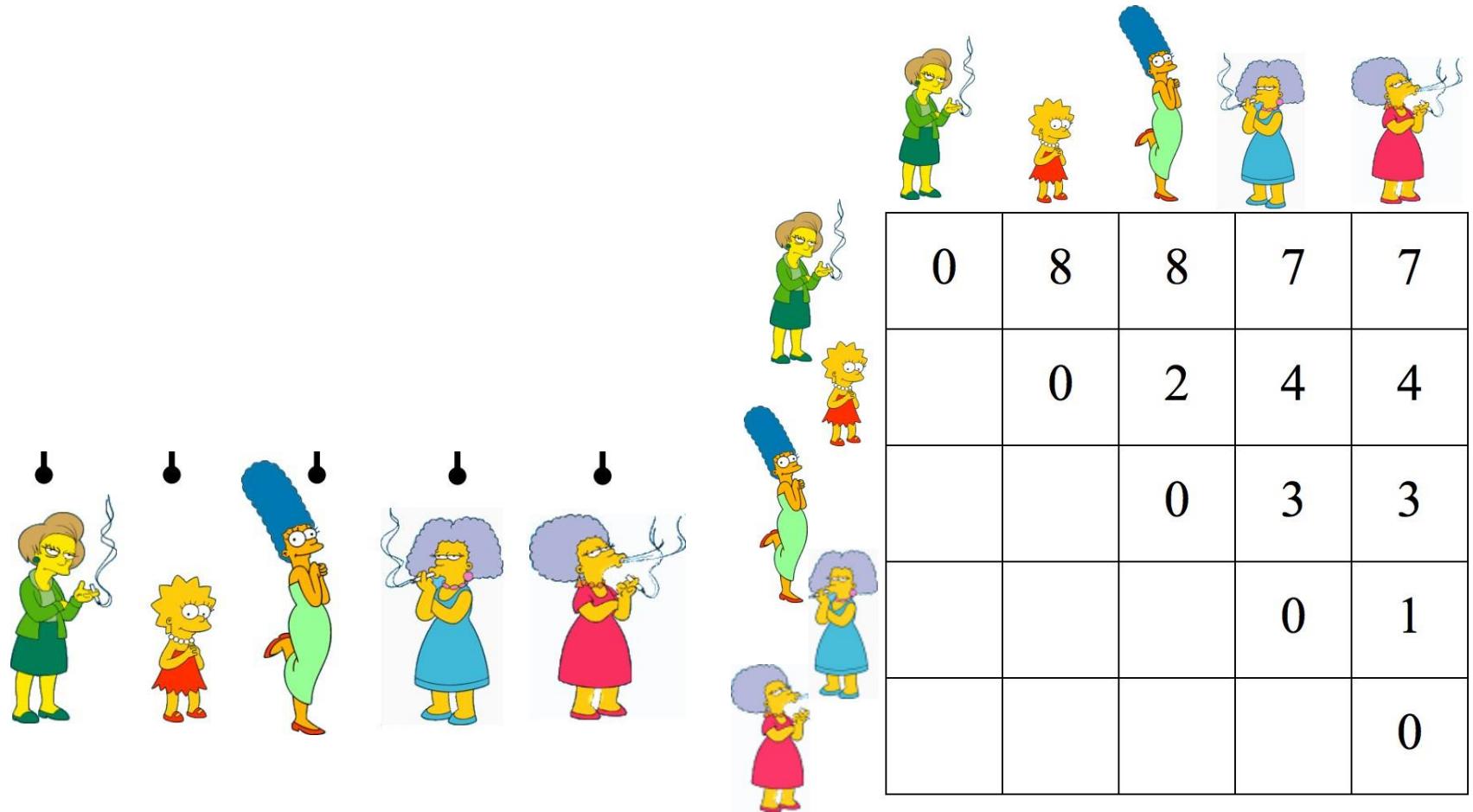
$$\begin{aligned}\text{sim}(x, x') &= \cos(\theta) \\ &= \frac{x^\top x'}{\|x\| \cdot \|x'\|} \\ &= \frac{x^\top x'}{\sqrt{x^\top x} \sqrt{x'^\top x'}}.\end{aligned}$$

# Desirable Properties of Clustering Algorithms

---

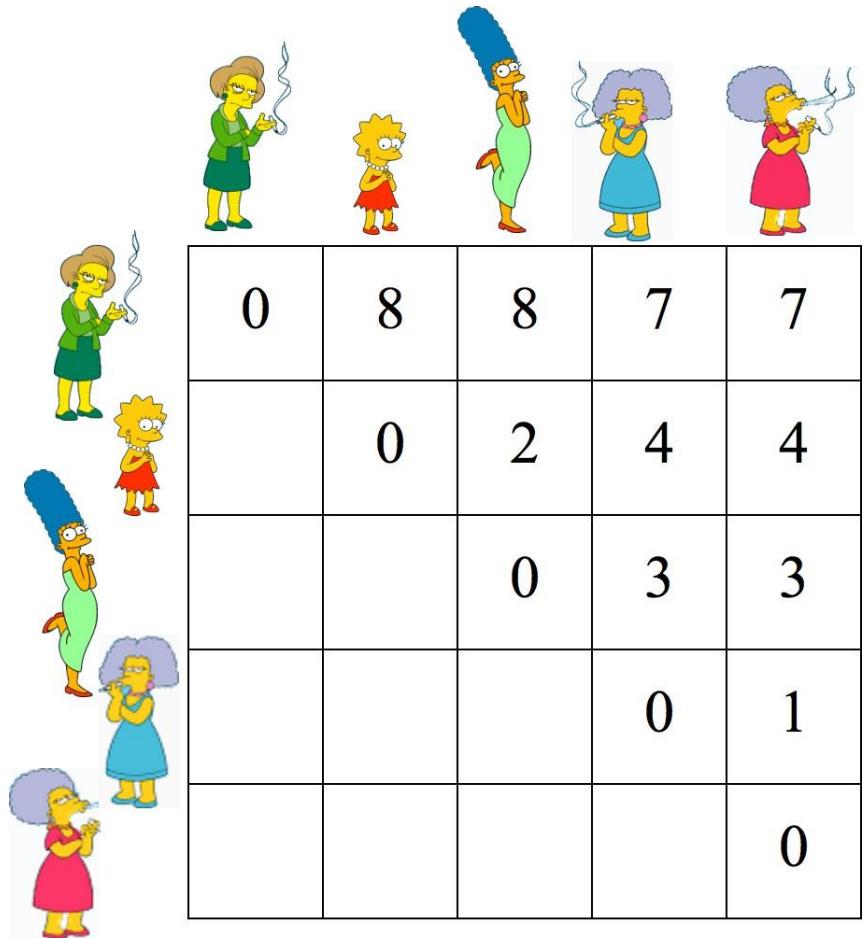
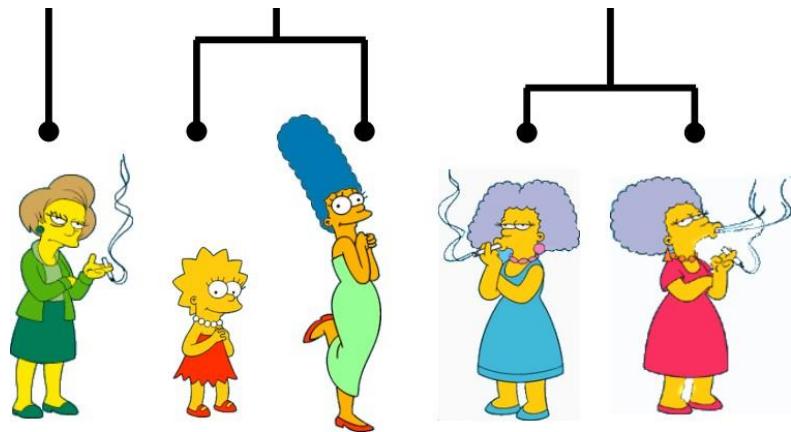
- Scalability (in terms of both time and space)
- Ability to deal with different data types
- Minimal requirements for domain knowledge to determine input parameters
- Interpretability and usability Optional
  - Incorporation of user-specified constraints

# Animated example



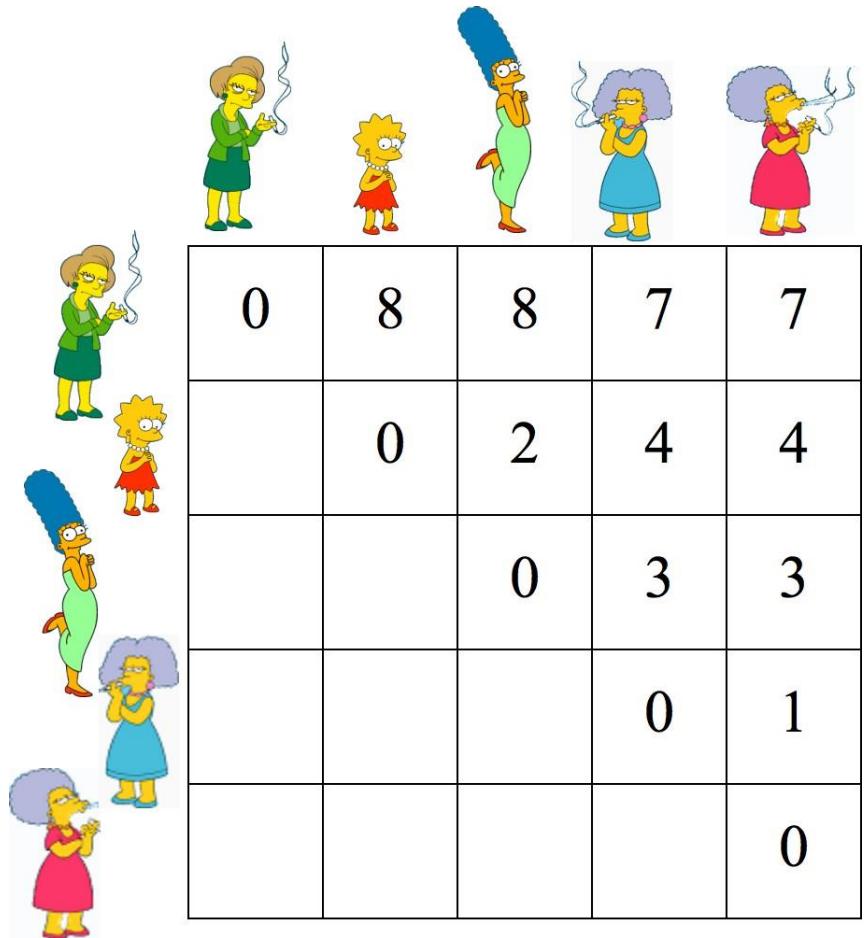
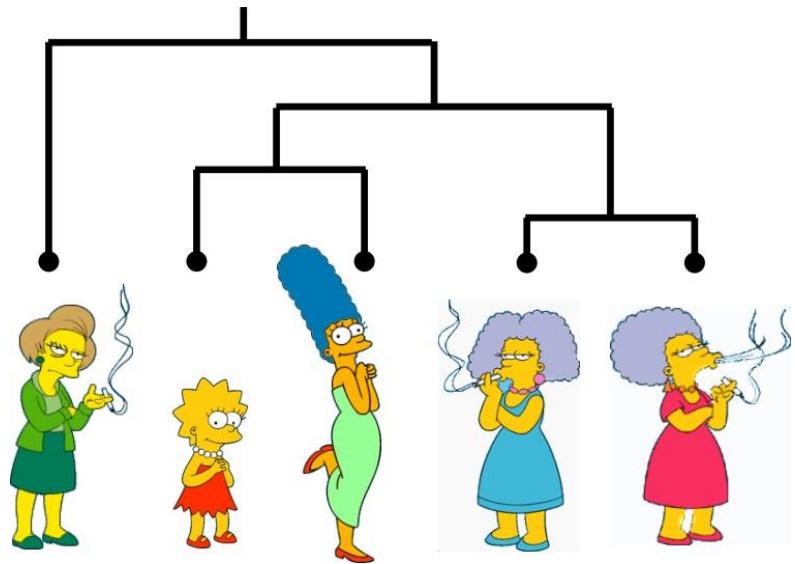
[source](#)

# Animated example



[source](#)

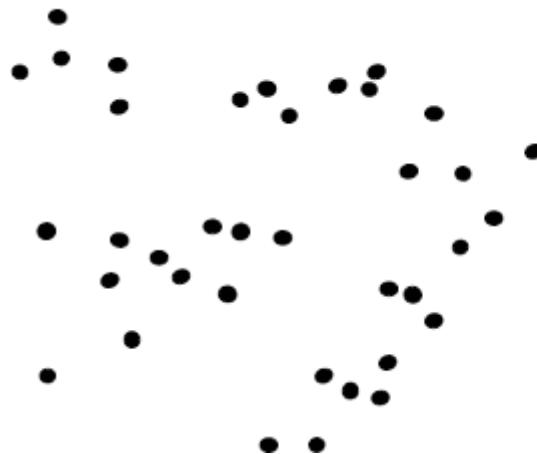
# Animated example



[source](#)

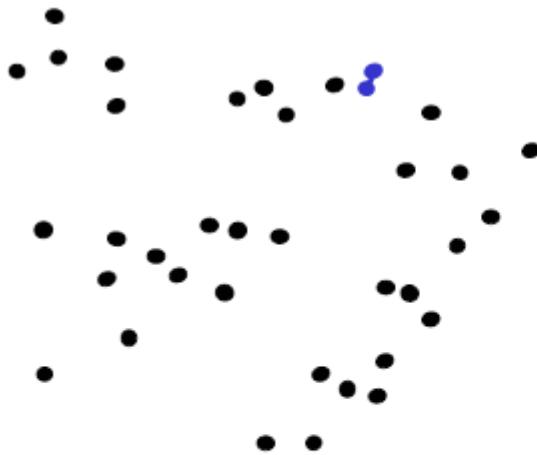
# Agglomerative clustering

---



1. Say "Every point is its own cluster"

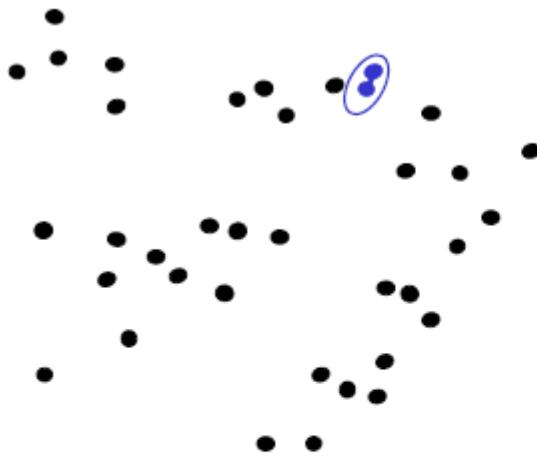
# Agglomerative clustering



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



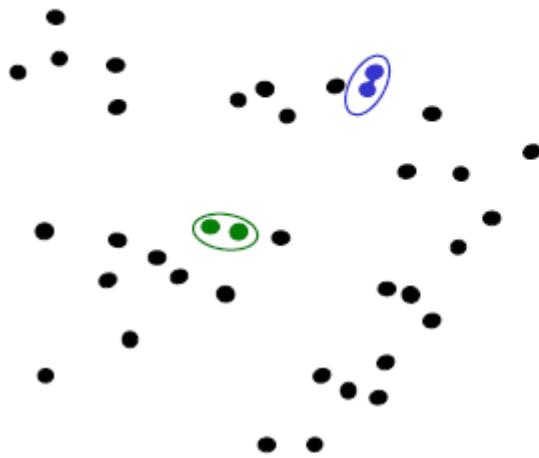
# Agglomerative clustering



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



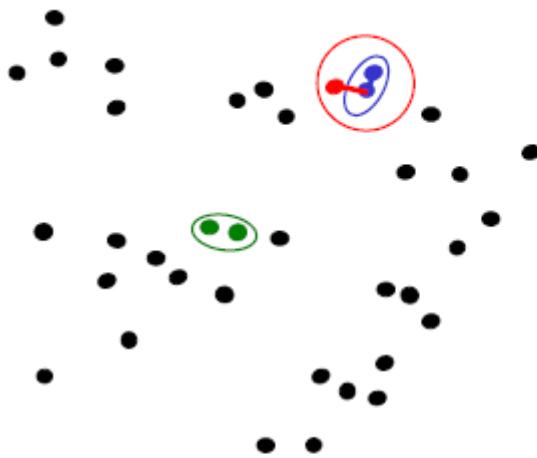
# Agglomerative clustering



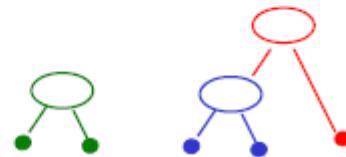
1. Say “Every point is its own cluster”
2. Find “most similar” pair of clusters
3. Merge it into a parent cluster
4. Repeat



# Agglomerative clustering



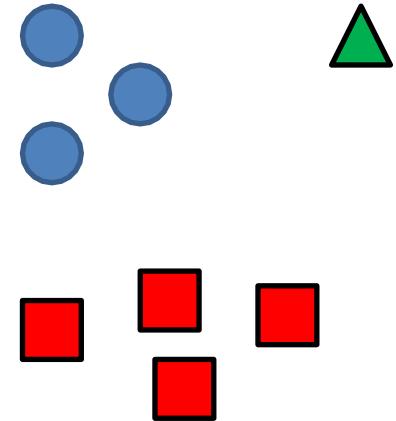
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat



# Agglomerative clustering

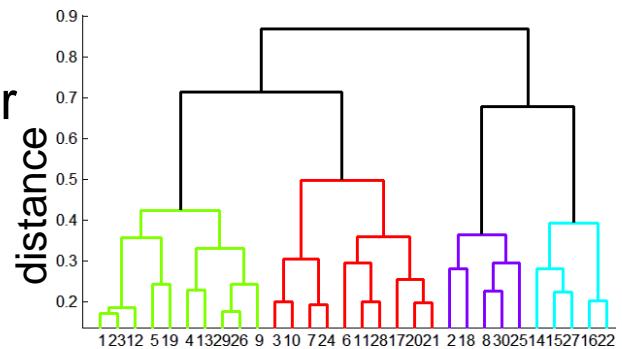
How to define cluster similarity?

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



How many clusters?

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



# Agglomerative Hierarchical Clustering - Algorithm

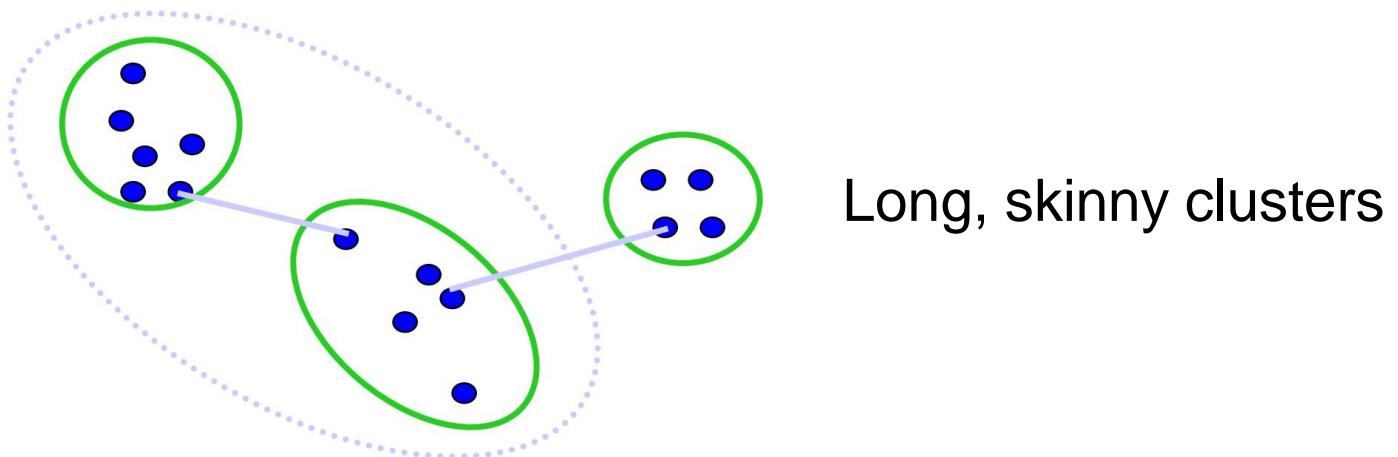
---

1. Initially each item  $x_1, \dots, x_n$  is in its own cluster  $C_1, \dots, C_n$ ;
2. Merge the nearest clusters, say  $C_i$  and  $C_j$ ;
3. Repeat until there is only one cluster left

# Different measures of nearest clusters

## Single Link

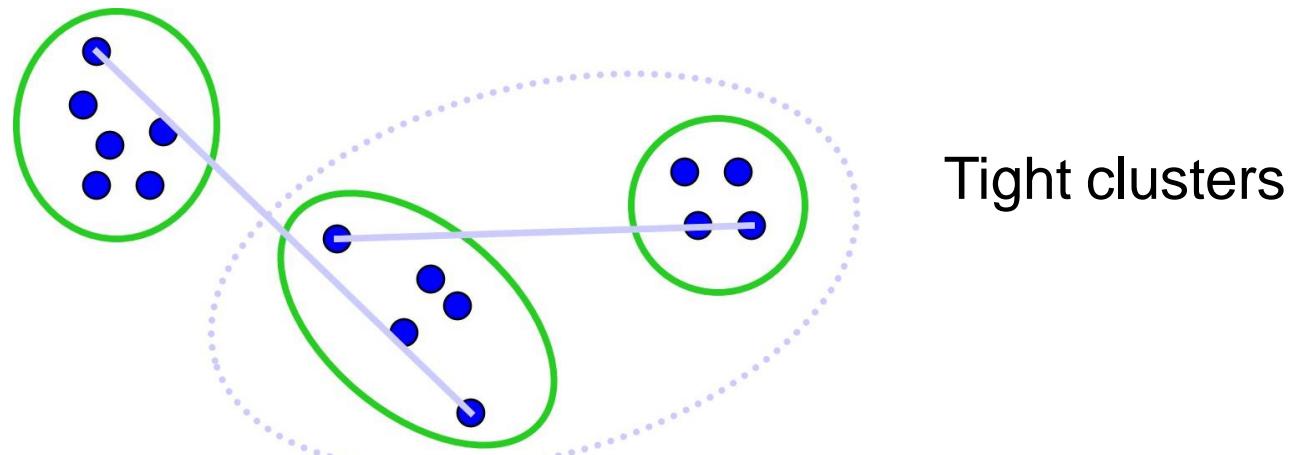
- $d(C_i, C_j) = \min_{x \in C_i, x' \in C_j} d(x, x')$ . This is known as *single-linkage*. It is equivalent to the minimum spanning tree algorithm. One can set a threshold and stop clustering once the distance between clusters is above the threshold. Single-linkage tends to produce long and skinny clusters.



# Different measures of nearest clusters

## Complete Link

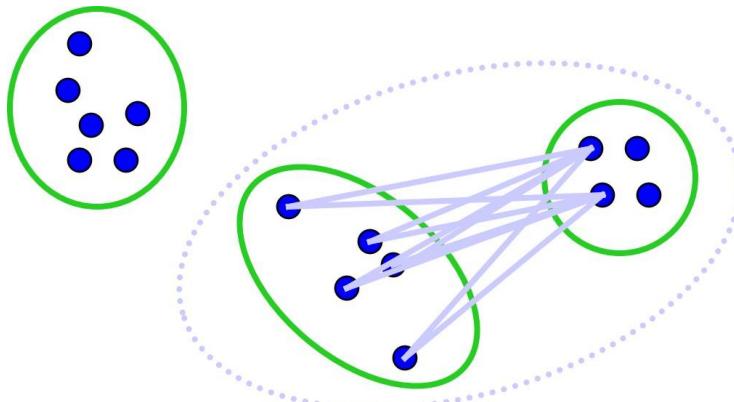
- $d(C_i, C_j) = \max_{x \in C_i, x' \in C_j} d(x, x')$ . This is known as *complete-linkage*. Clusters tend to be compact and roughly equal in diameter.



# Different measures of nearest clusters

## Average Link

- $d(C_i, C_j) = \frac{\sum_{x \in C_i, x' \in C_j} d(x, x')}{|C_i| \cdot |C_j|}$ . This is the average distance between items. Somewhere between single-linkage and complete-linkage.



Robust against noise.

# Conclusions: Agglomerative Clustering

---

## Good

- Simple to implement, widespread application.
- Clusters have adaptive shapes.
- Provides a hierarchy of clusters.
- No need to specify number of clusters in advance.

## Bad

- May have imbalanced clusters.
- Still have to choose number of clusters or threshold.
- Does not scale well. Runtime of  $O(n^3)$ .
- Can get stuck at a local optima.

# What we will learn today?

---

- Introduction to segmentation and clustering
- Gestalt theory for perceptual grouping
- Agglomerative clustering
- Graph-based segmentation

# Example

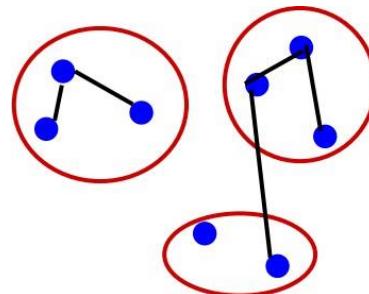


- Graph-based clustering for Image Segmentation
  - Introduced by *Felzenszwalb and Huttenlocher* in the paper titled *Efficient Graph-Based Image Segmentation*.

# Problem Formulation

---

- Graph  $G = (V, E)$
- $V$  is set of nodes (i.e. pixels)
- $E$  is a set of undirected edges between pairs of pixels
- $w(v_i, v_j)$  is the weight of the edge between nodes  $v_i$  and  $v_j$ .
- $S$  is a segmentation of a graph  $G$  such that  $G' = (V, E')$  where  $E' \subset E$ .
- $S$  divides  $G$  into  $G'$  such that it contains distinct clusters  $C$ .



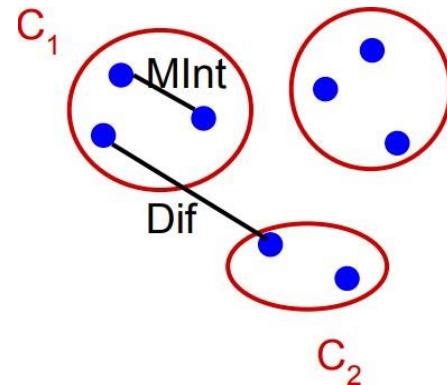
# Predicate for Segmentation

- Predicate D determines whether there is a boundary for segmentation.

$$\text{Merge}(C_1, C_2) = \begin{cases} \text{True} & \text{if } \text{dif}(C_1, C_2) < \text{in}(C_1, C_2) \\ \text{False} & \text{otherwise} \end{cases}$$

Where

- $\text{dif}(C_1, C_2)$  is the difference between two clusters.
- $\text{in}(C_1, C_2)$  is the internal difference in the clusters  $C_1$  and  $C_2$



# Predicate for Segmentation

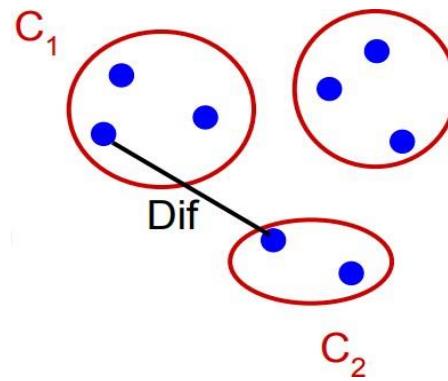
---

- Predicate D determines whether there is a boundary for segmentation.

$$Merge(C_1, C_2) = \begin{cases} True & \text{if } dif(C_1, C_2) < in(C_1, C_2) \\ False & \text{otherwise} \end{cases}$$

$$dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (C_1, C_2) \in E} w(v_i, v_j)$$

The difference between two components is the minimum weight edge that connects a node  $v_i$  in clusters  $C_1$  to node  $v_j$  in  $C_2$



# Predicate for Segmentation

---

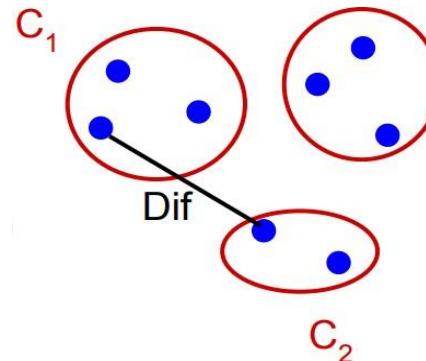
- Predicate D determines whether there is a boundary for segmentation.

$$\text{Merge}(C_1, C_2) = \begin{cases} \text{True} & \text{if } \text{dif}(C_1, C_2) < \text{in}(C_1, C_2) \\ \text{False} & \text{otherwise} \end{cases}$$

$$\text{dif}(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (C_1, C_2) \in E} w(v_i, v_j)$$

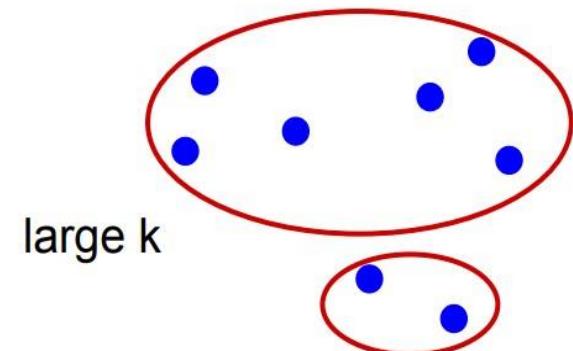
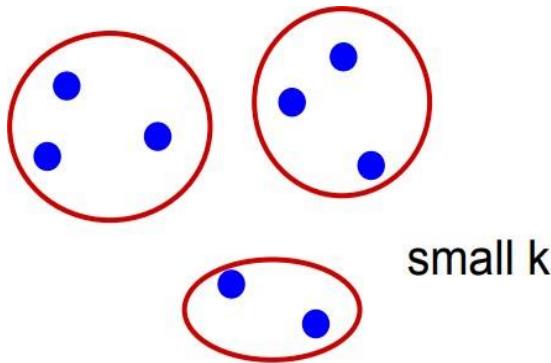
$$\text{in}(C_1, C_2) = \min_{C \in \{C_1, C_2\}} \left[ \max_{v_i, v_j \in C} \left[ w(v_i, v_j) + \frac{k}{|C|} \right] \right]$$

$\text{In}(C_1, C_2)$  is to the maximum weight edge that connects two nodes in the same component.



# Predicate for Segmentation

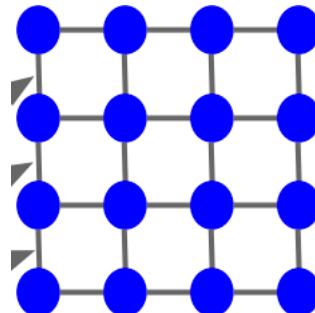
- $k/|C|$  sets the threshold by which the components need to be different from the internal nodes in a component.
- Properties of constant  $k$ :
  - If  $k$  is large, it causes a preference of larger objects.
  - $k$  does not set a minimum size for components.



# Features and weights

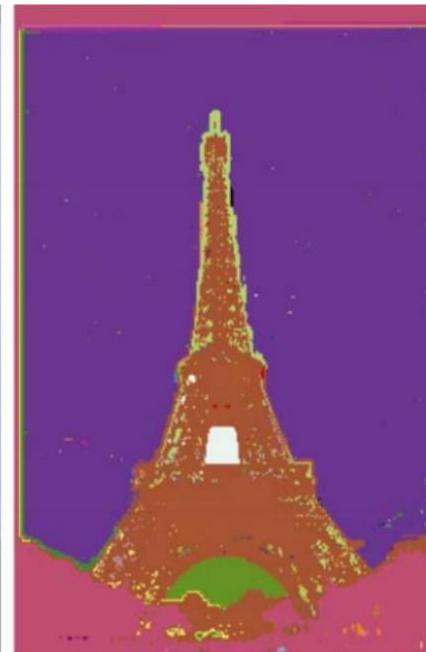
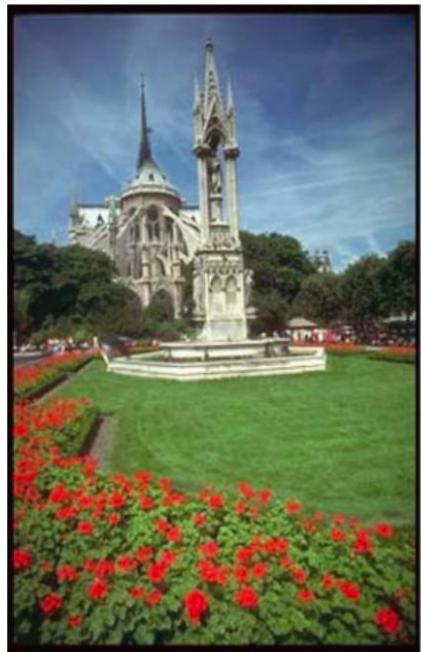
---

- Project every pixel into feature space defined by (x, y, r, g, b).
- Every pixel is connected to its neighboring pixels and the weights are determined by the difference in intensities.
- Weights between pixels are determined using L2 (Euclidian) distance in feature space.
- Edges are chosen for only top ten nearest neighbors in feature space to ensure run time of  $O(n \log n)$  where n is number of pixels.



# Results

---





# *Lecture 9: Segmentation and Clustering, Part II*

**Pattern Recognition and Computer Vision**

Guanbin Li,

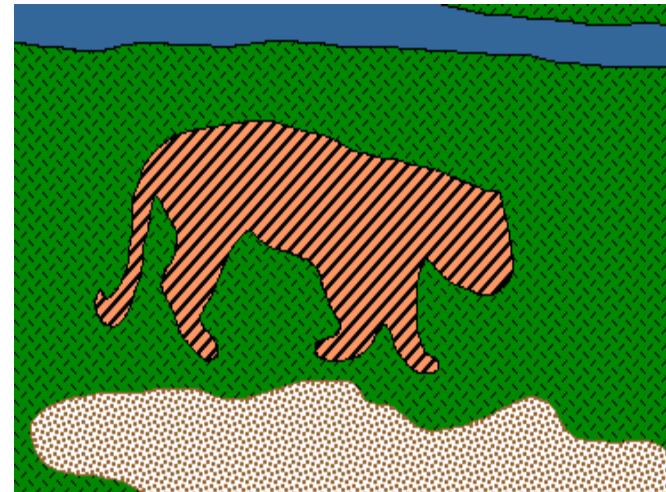
School of Computer Science and Engineering, Sun Yat-Sen University

# CS 131 Roadmap

Pixels	<b>Segments</b>	Images	Videos	Web
Convolutions Edges Descriptors	<b>Segmentation</b> <b>Clustering</b> Resizing	Recognition Detection Machine learning	Motion Tracking	Neural networks Convolutional neural networks

# Recap: Image Segmentation

- Goal: identify groups of pixels that go together



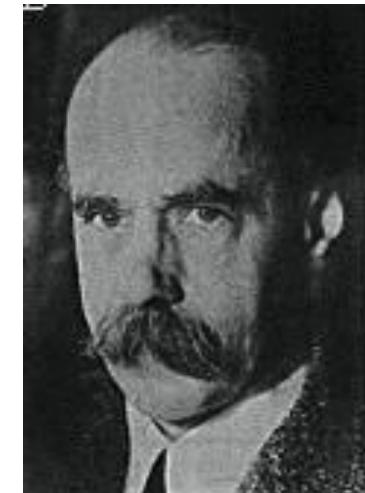
# Recap: Gestalt Theory

---

- Gestalt: whole or group
  - Whole is greater than sum of its parts
  - Relationships among parts can yield new properties/features
- Psychologists identified series of factors that predispose set of elements to be grouped (by human visual system)

*"I stand at the window and see a house, trees, sky.  
Theoretically I might say there were 327 brightnesses  
and nuances of colour. Do I have "327"? No. I have sky, house, and  
trees."*

Max Wertheimer  
(1880-1943)

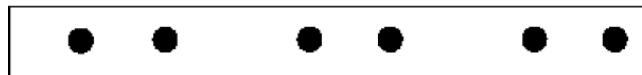


Untersuchungen zur Lehre von der Gestalt,  
*Psychologische Forschung*, Vol. 4, pp. 301-350, 1923  
<http://psy.ed.asu.edu/~classics/Wertheimer/Forms/forms.htm>

# Recap: Gestalt Factors



Not grouped



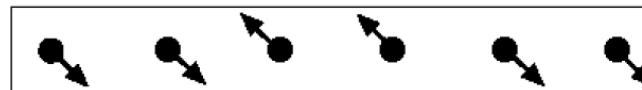
Proximity



Similarity



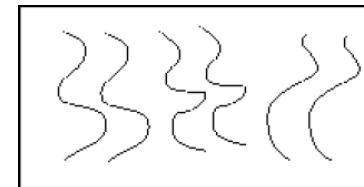
Similarity



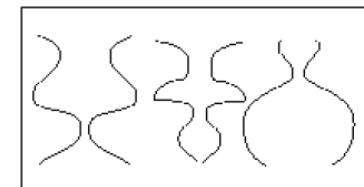
Common Fate



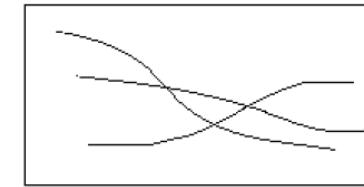
Common Region



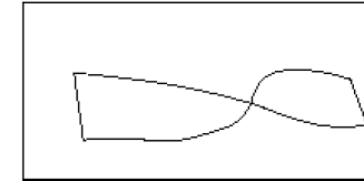
Parallelism



Symmetry



Continuity



Closure

- These factors make intuitive sense, but are very difficult to translate into algorithms.

# What will we learn today?

---

- K-means clustering
- Mean-shift clustering

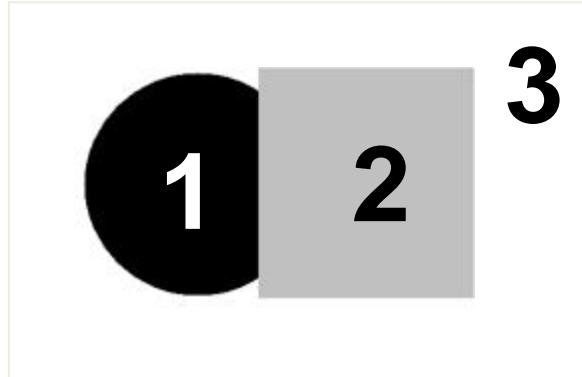
**Reading:** D. Comaniciu and P. Meer, [Mean Shift: A Robust Approach toward Feature Space Analysis](#), PAMI 2002.

# What will we learn today?

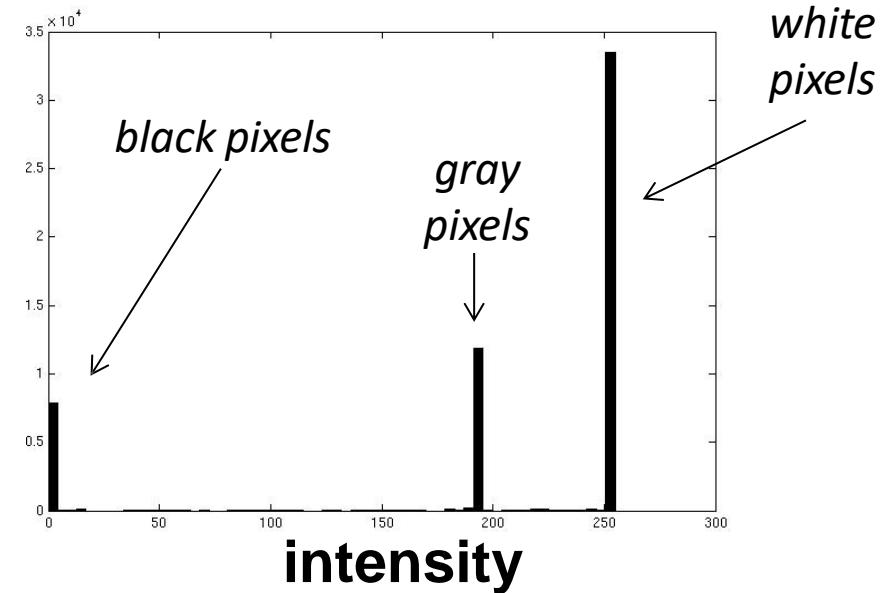
---

- K-means clustering
- Mean-shift clustering

# Image Segmentation: Toy Example

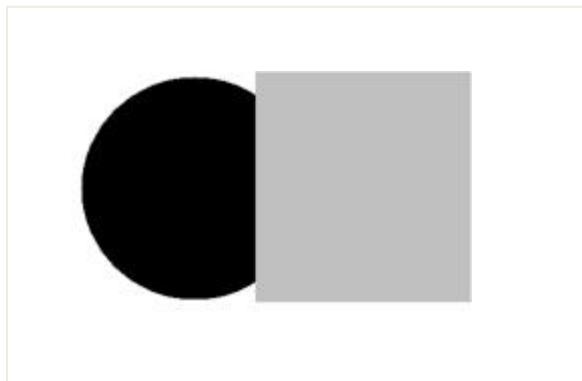


input image

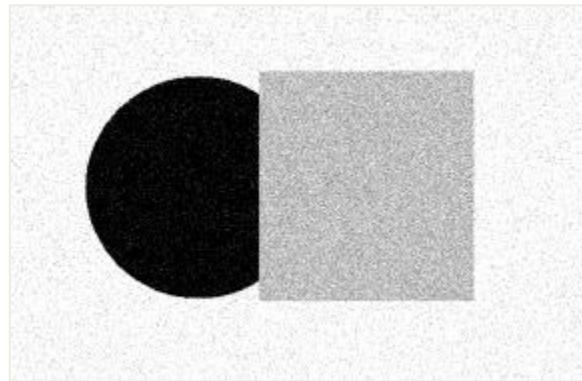
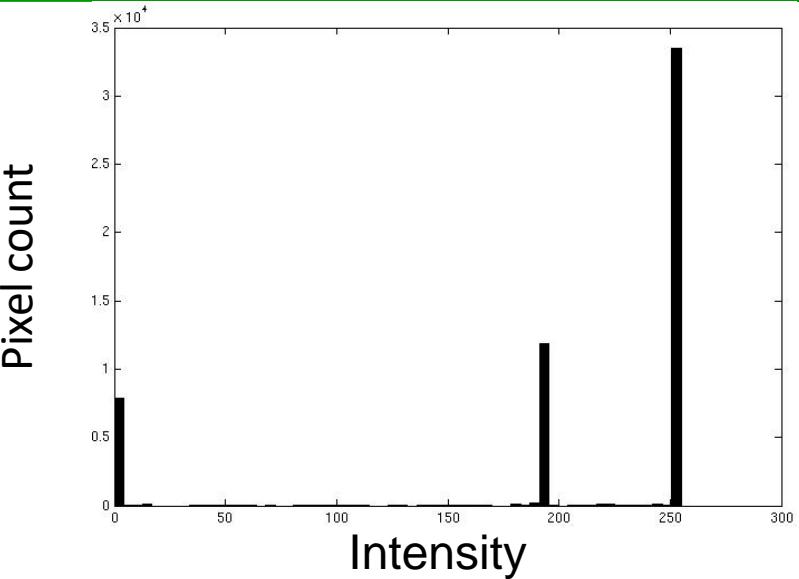


- These intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., segment the image based on the intensity feature.
- What if the image isn't quite so simple?

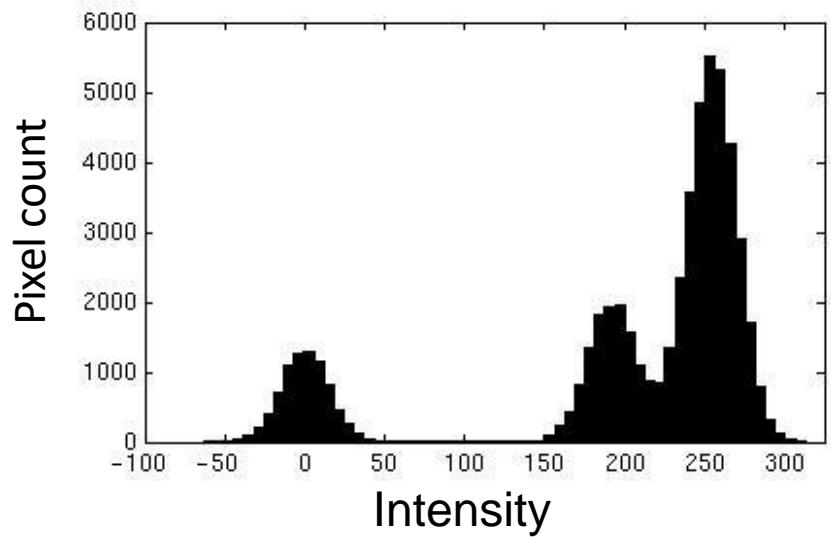
# Image Segmentation: Toy Example



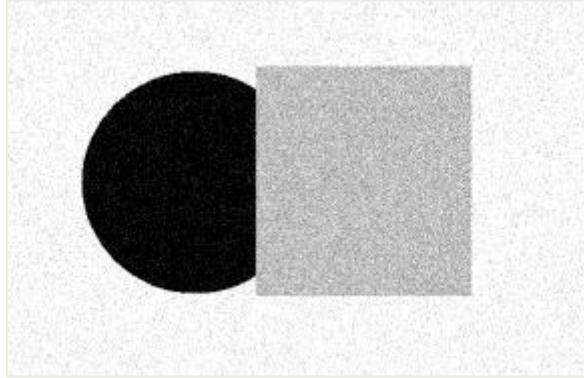
Input image



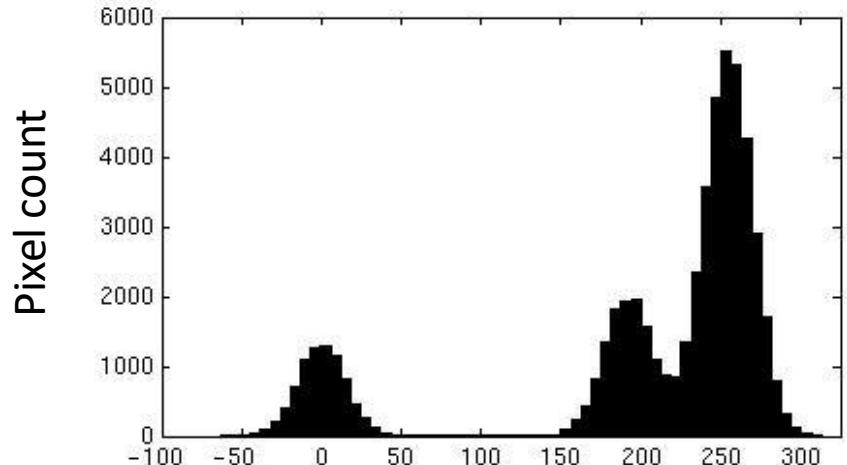
Input image



# Image Segmentation: Toy Example



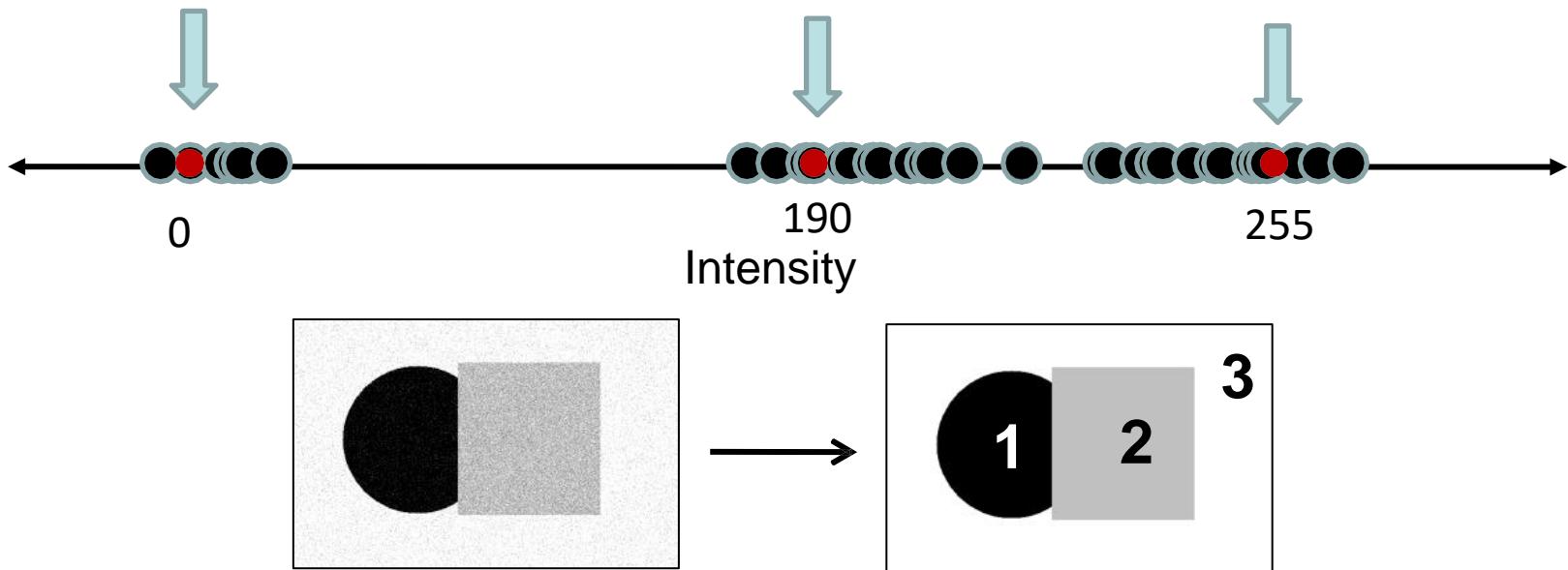
Input image



Intensity

- Now how to determine the three main intensities that define our groups?
- We need to cluster.

# Image Segmentation: Toy Example



- Goal: choose three “centers” as the representative intensities, and label every pixel according to which of these centers it is nearest to.
- Best cluster centers are those that minimize Sum of Square Distance (SSD) between all points and their nearest cluster center  $C_i$ :

$$SSD = \sum_{clusteri} \sum_{x \in clusteri} (x - c_i)^2$$

# Clustering for Summarization

Goal: cluster to minimize variance in data given clusters  
– Preserve information

$$c^*, \delta^* = \arg \min_{c, \delta} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij} (c_i - x_j)^2$$

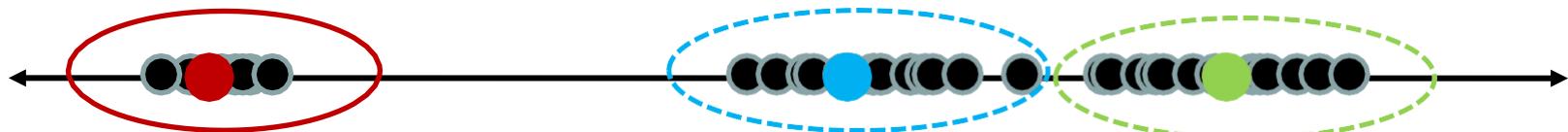
Cluster center

Data

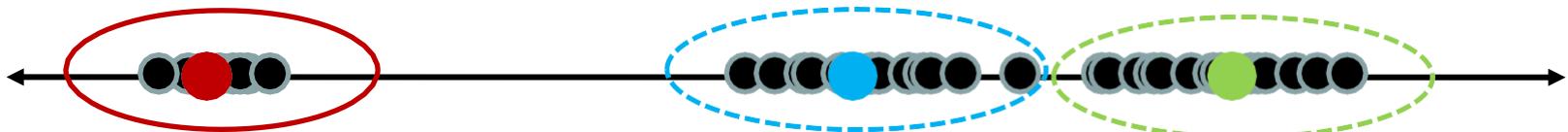
Whether  $x_j$  is assigned to  $c_i$

# Clustering

- With this objective, it is a “chicken and egg” problem:
  - If we knew the ***cluster centers***, we could allocate points to groups by assigning each to its closest center.



- If we knew the ***group memberships***, we could get the centers by computing the mean per group.



# K-means clustering

---

1. Initialize( $t=0$ ): cluster centers  $c_1, \dots, c_K$

2. Compute  $\delta^t$ : assign each point to the closest center

-  $\delta^t$  denotes the set of assignment for each  $x_j$  to cluster  $c_i$  at iteration  $t$

$$\delta^t = \operatorname{argmin}_{\delta} \frac{1}{N} \sum_j \sum_i \delta_{ij}^{t-1} (c_i^{t-1} - x_j)^2$$

3. Compute  $c^t$ : update cluster centers as the mean of the points

$$c^t = \operatorname{argmin}_c \frac{1}{N} \sum_j \sum_i \delta_{ij}^t (c_i^{t-1} - x_j)^2$$

4. Update  $t=t+1$ , Repeat Step 2-3 till stopped



# K-means clustering

---

1. Initialize( $t=0$ ): cluster centers  $c_1, \dots, c_K$

- Commonly used: random initialization
- Or greedily choose K to minimize residual

2. Compute  $\delta^t$ : assign each point to the closest center

- Typical distance measure:

- Euclidean  $sim(x, x!) = x^T x!$

- Cosine  $sim(x, x!) = x^T x! / (\|x\| \cdot \|x!\|)$

- Others

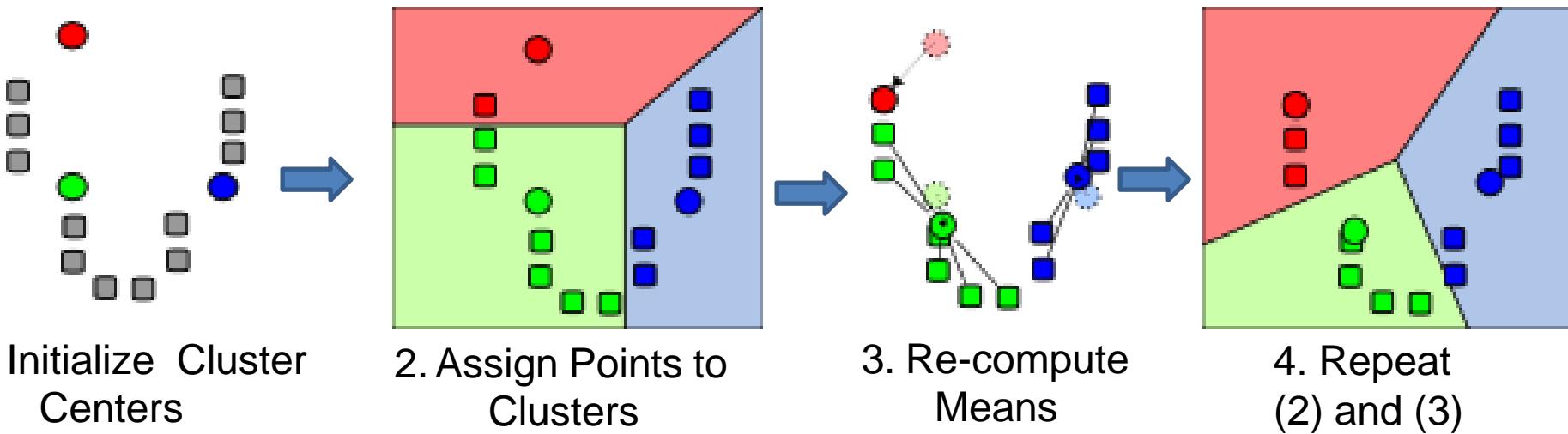
3. Compute  $c^t$ : update cluster centers as the mean of the points

$$c^t = \operatorname{argmin}_c \frac{1}{N} \sum_j \sum_i \delta_{ij}^t (c_i^{t-1} - x_j)^2$$

4. Update  $t=t+1$ , Repeat Step 2-3 till stopped

- $c^t$  doesn't change anymore.

# K-means clustering

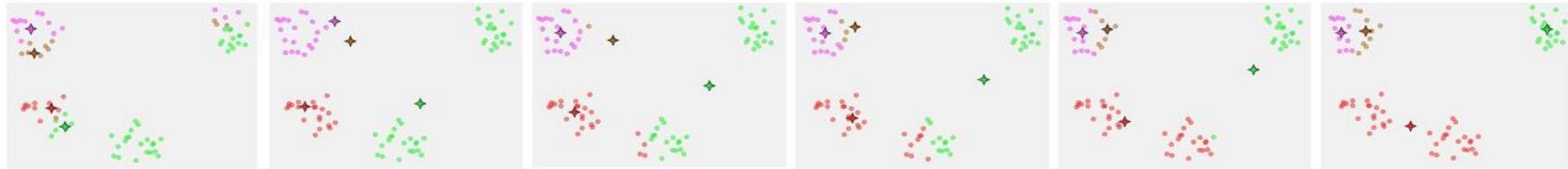


- Java demo:

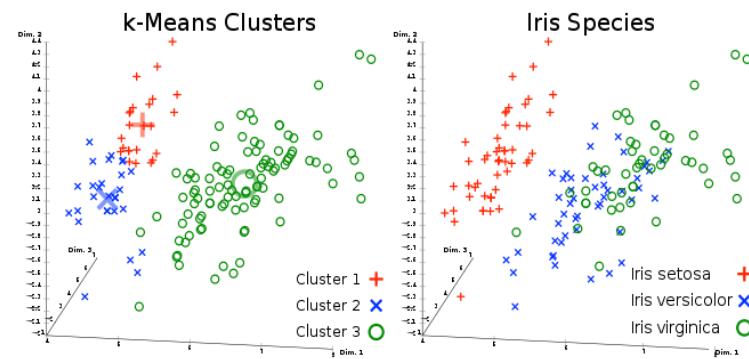
<https://stanford.edu/class/engr108/visualizations/kmeans/kmeans.html>

# K-means clustering

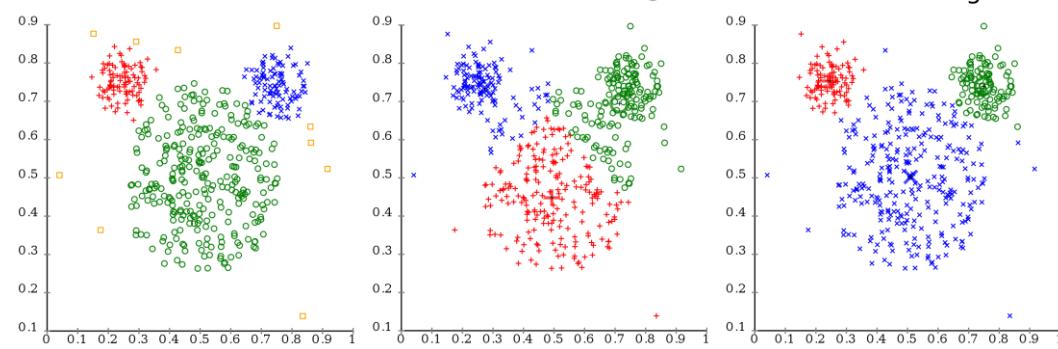
- Converges to a *local minimum* solution
  - Initialize multiple runs



- Better fit for spherical data, expect clusters to be of similar size



Different cluster analysis results on "mouse" data set:  
Original Data



- Need to pick K (# of clusters)

# Segmentation as Clustering

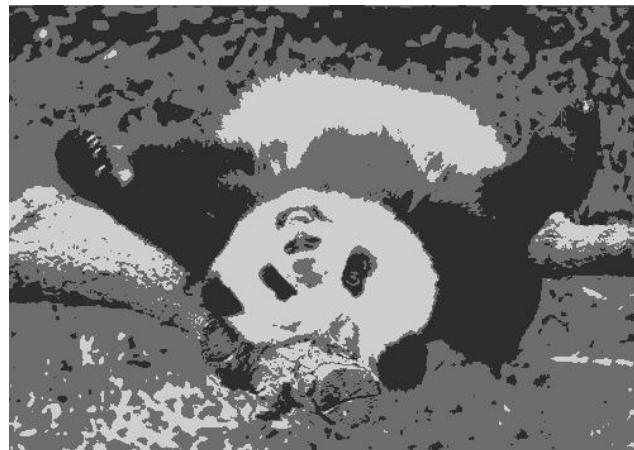
---



Original image



2 clusters



3 clusters

# Feature Space

---

- Depending on what we choose as the *feature space*, we can group pixels in different ways.

- Grouping pixels based on  
**intensity** similarity

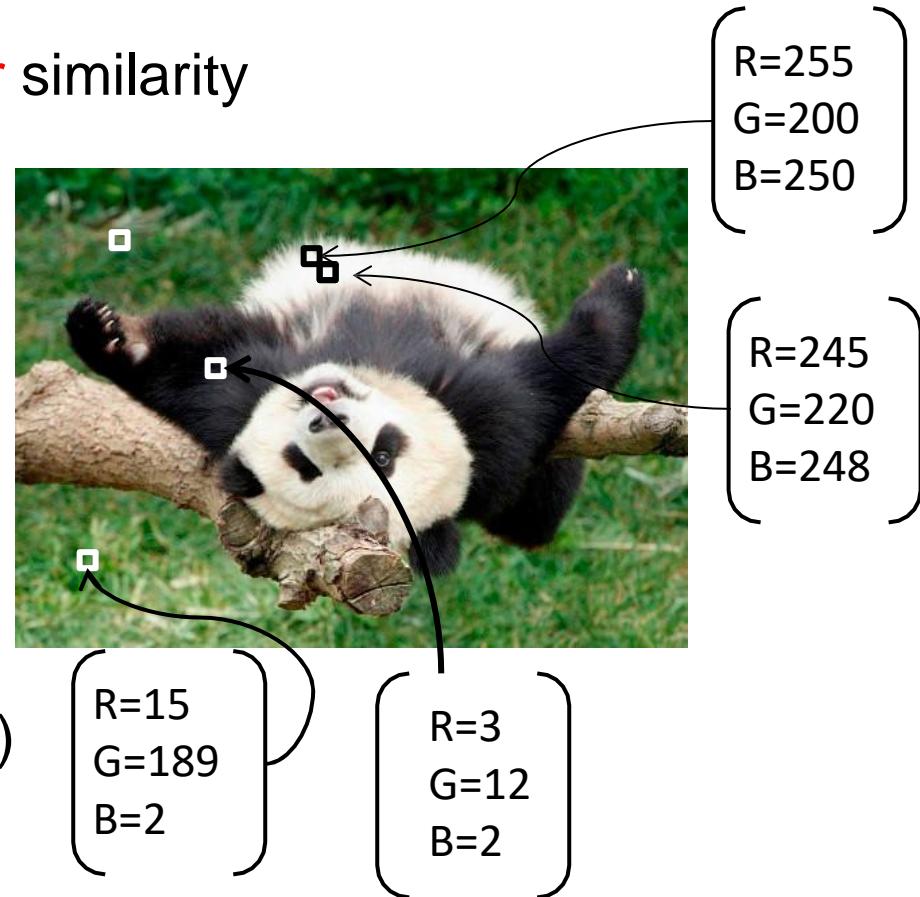
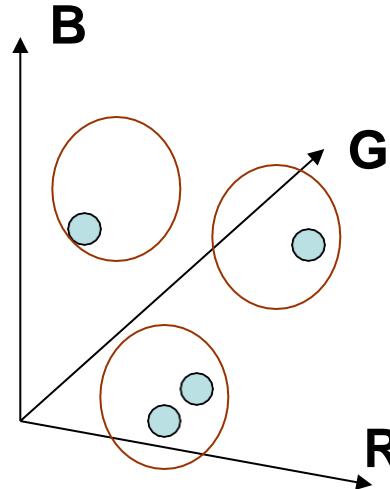


- Feature space: intensity value (1D)



# Feature Space

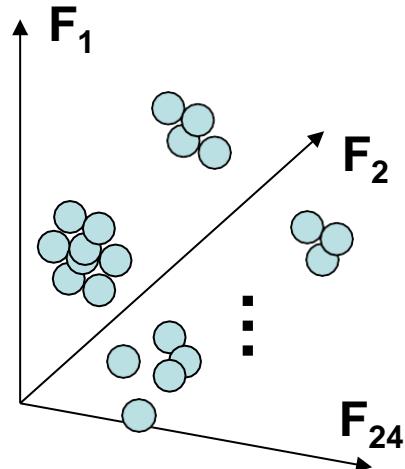
- Depending on what we choose as the *feature space*, we can group pixels in different ways.
- Grouping pixels based on **color** similarity



- Feature space: color value (3D)

# Feature Space

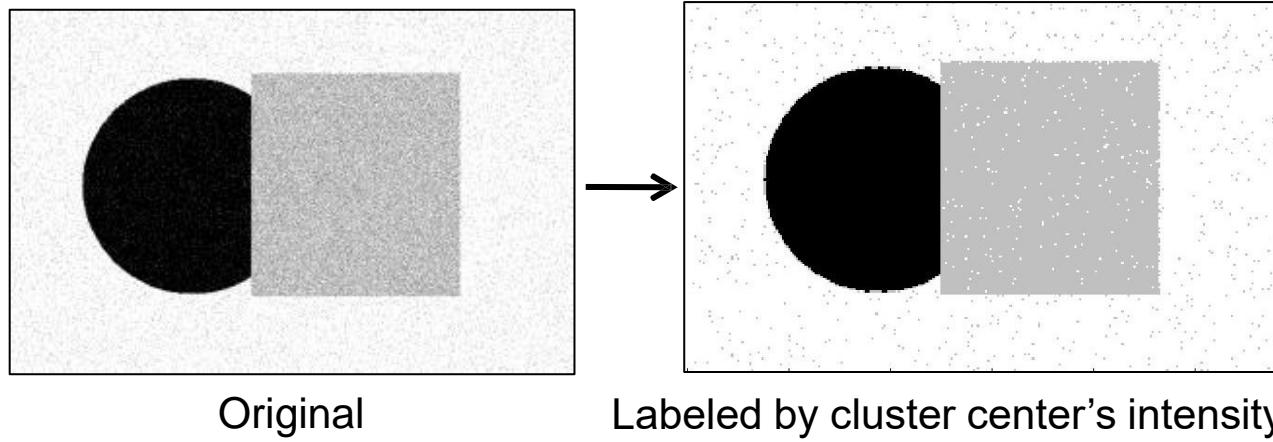
- Depending on what we choose as the *feature space*, we can group pixels in different ways.
- Grouping pixels based on **texture** similarity



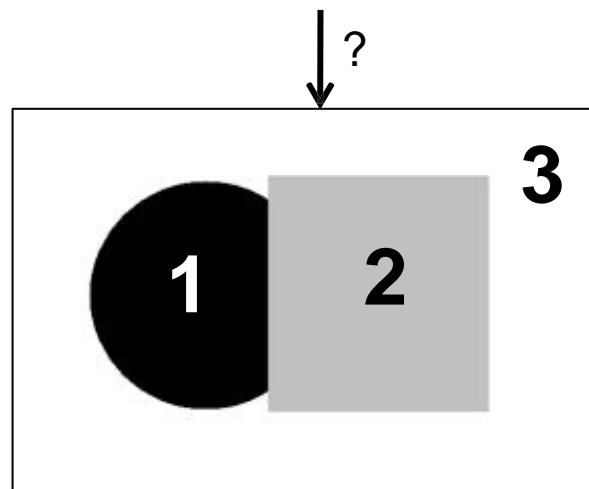
- Feature space: filter bank responses (e.g., 24D)

# Smoothing Out Cluster Assignments

- Assigning a cluster label per pixel may yield outliers:

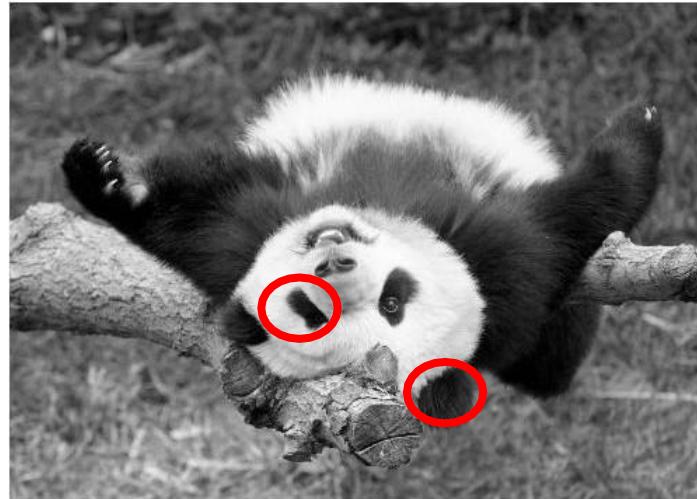
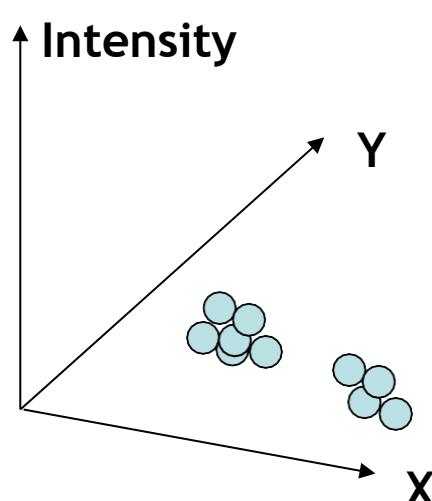


- How can we ensure they are spatially smooth?



# Segmentation as Clustering

- Depending on what we choose as the *feature space*, we can group pixels in different ways.
- Grouping pixels based on *intensity + position* similarity



⇒ Way to encode both *similarity* and *proximity*.

# K-Means Clustering Results

- 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



Image source: Forsyth & Ponce

# K-Means Clustering Results

- K-means clustering based on intensity or color is essentially vector quantization of the image attributes
  - Clusters don't have to be spatially coherent
- Clustering based on (r,g,b,x,y) values enforces more spatial coherence



Image source: Forsyth & Ponce

# How to evaluate clusters?

---

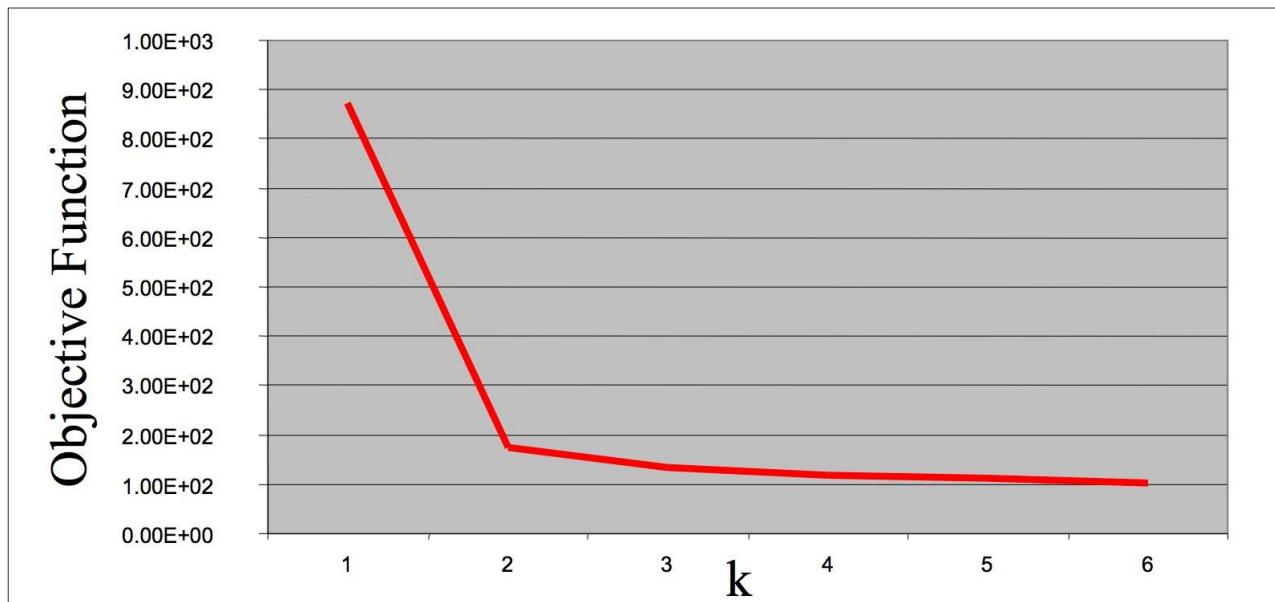
- Generative
  - How well are points reconstructed from the clusters?
- Discriminative
  - How well do the clusters correspond to labels?
    - Can we correctly classify which pixels belong to the panda?
  - Note: unsupervised clustering does not aim to be discriminative as we don't have the labels.

# How to choose the number of clusters?

Try different numbers of clusters in a validation set and look at performance.

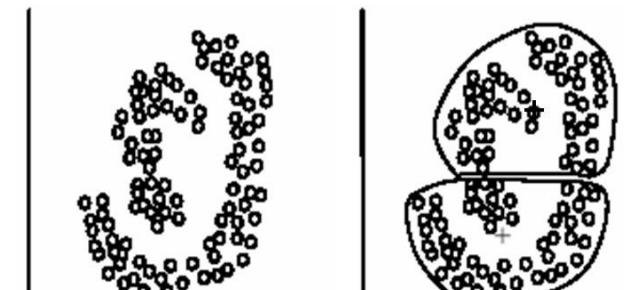
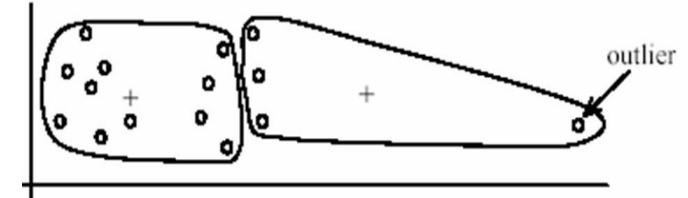
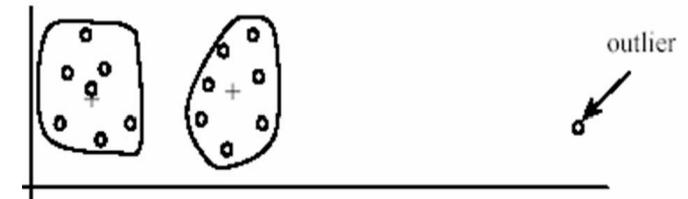
We can plot the objective function values for  $k$  equals 1 to 6...

The abrupt change at  $k = 2$ , is highly suggestive of two clusters in the data. This technique for determining the number of clusters is known as “knee finding” or “elbow finding”.



# K-Means pros and cons

- Pros
  - Finds cluster centers that minimize conditional variance (good representation of data)
  - Simple and fast, Easy to implement
- Cons
  - Need to choose K
  - Sensitive to outliers
  - Prone to local minima
  - Spherical cluster assumption
  - All clusters have the same parameters (e.g., distance measure is non-adaptive)
- Usage
  - Unsupervised clustering
  - Rarely used for pixel segmentation



(A): Two natural clusters

# What will we learn today?

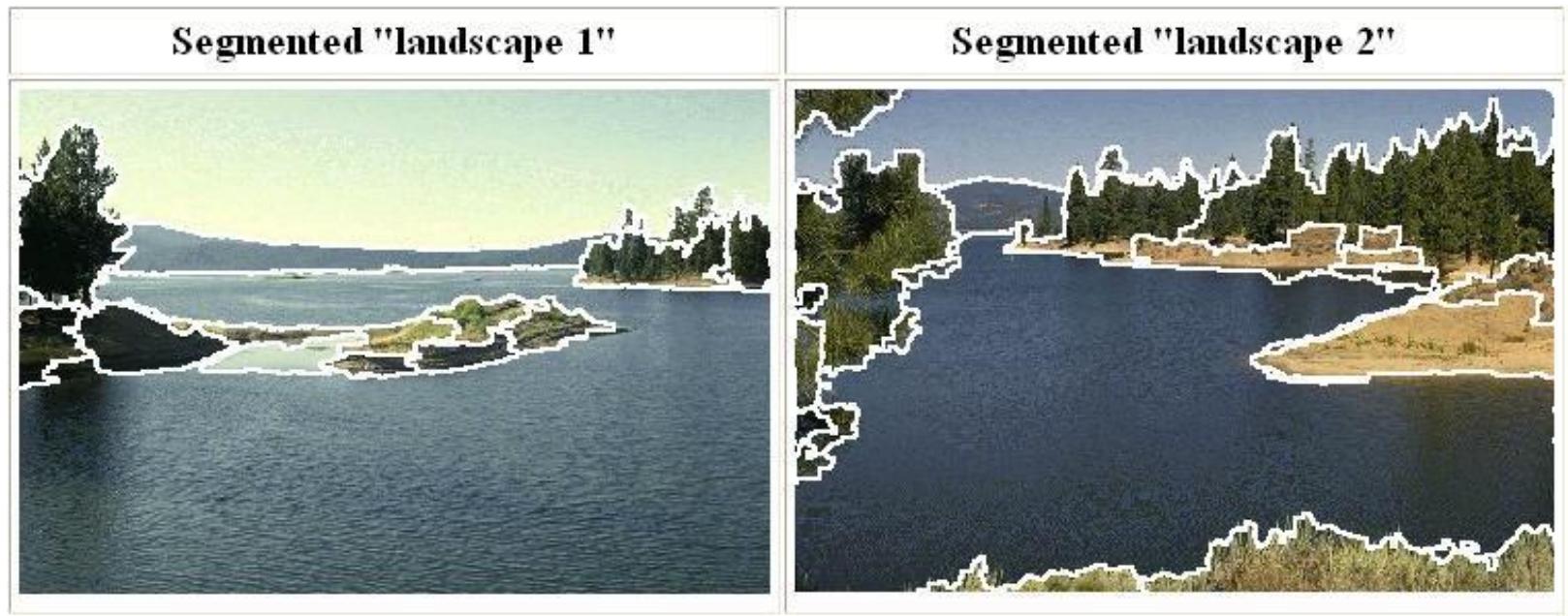
---

- K-means clustering
- Mean-shift clustering

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

# Mean-Shift Segmentation

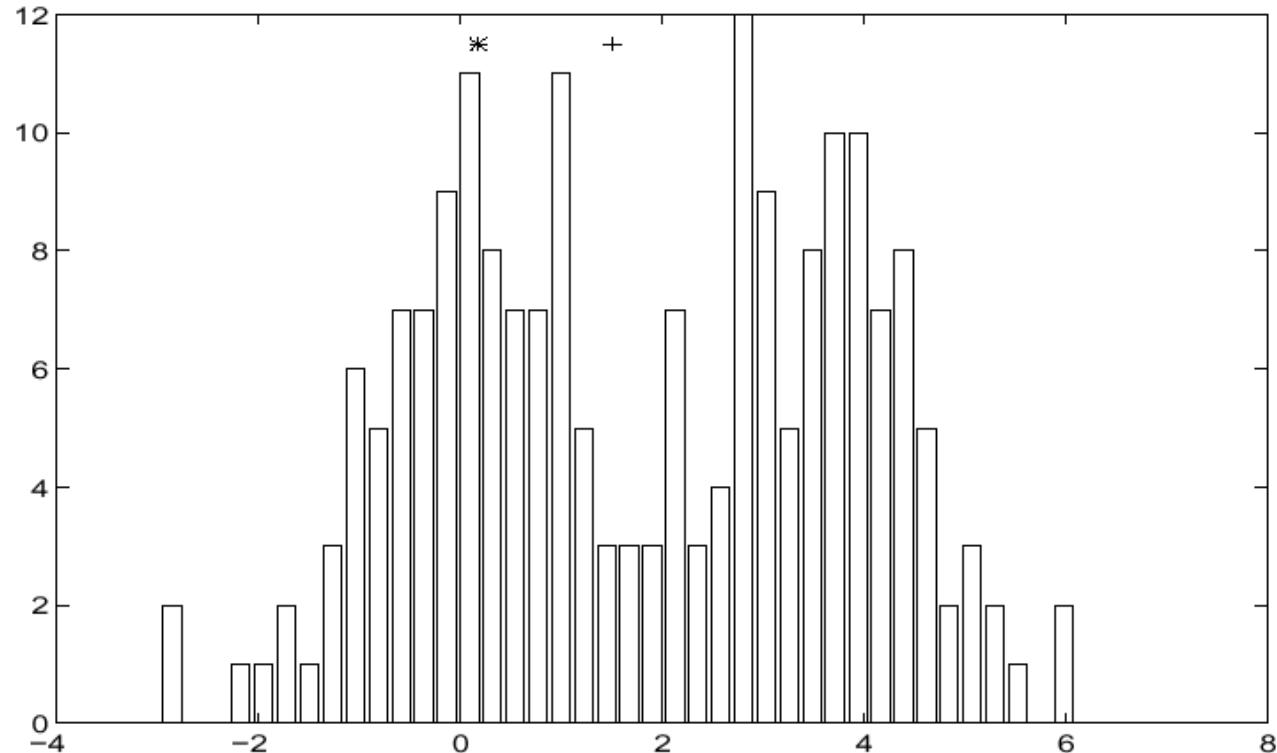
- An advanced and versatile technique for clustering-based segmentation



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

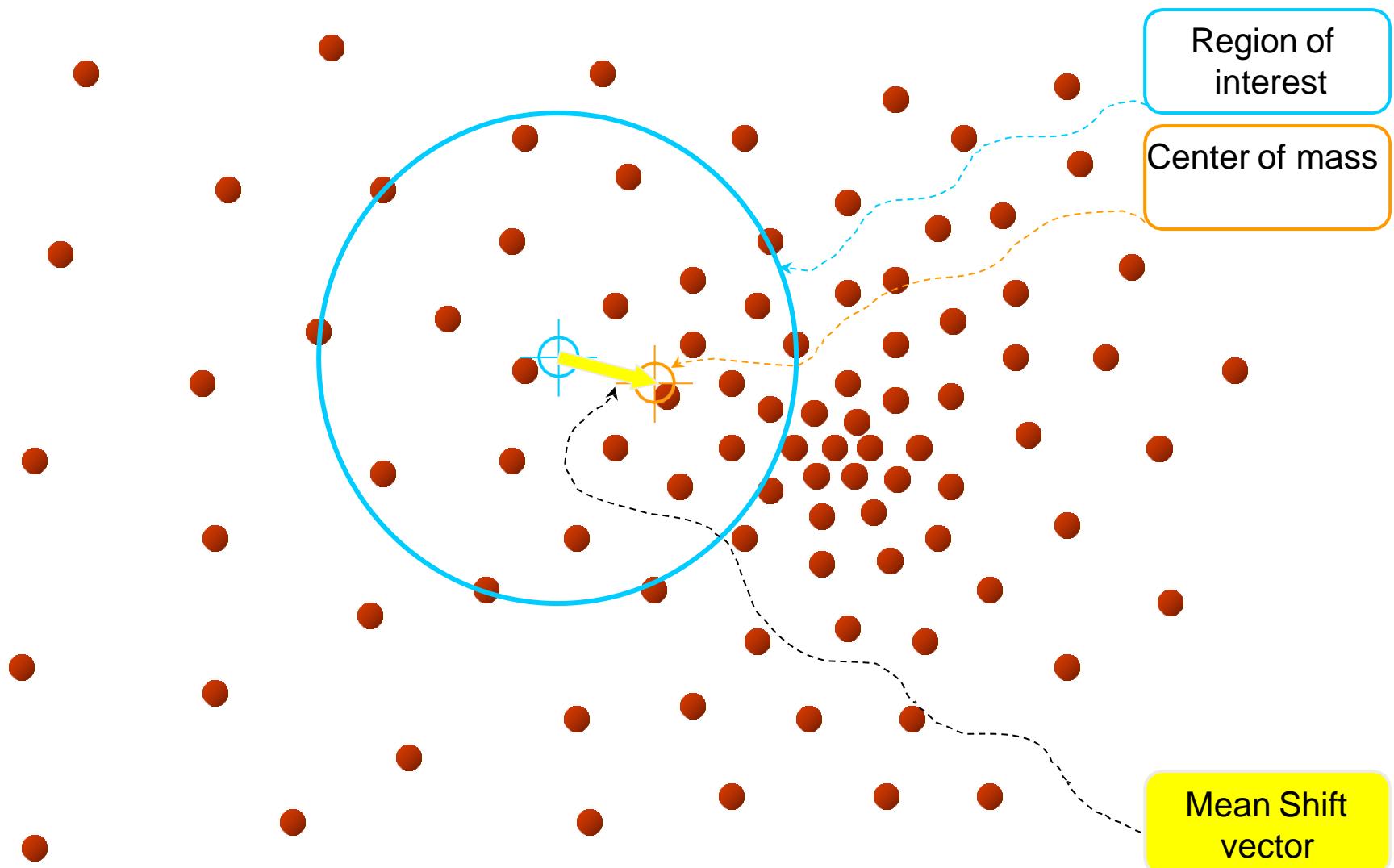
D. Comaniciu and P. Meer, [Mean Shift: A Robust Approach toward Feature Space Analysis](#), PAMI 2002.

# Mean-Shift Algorithm

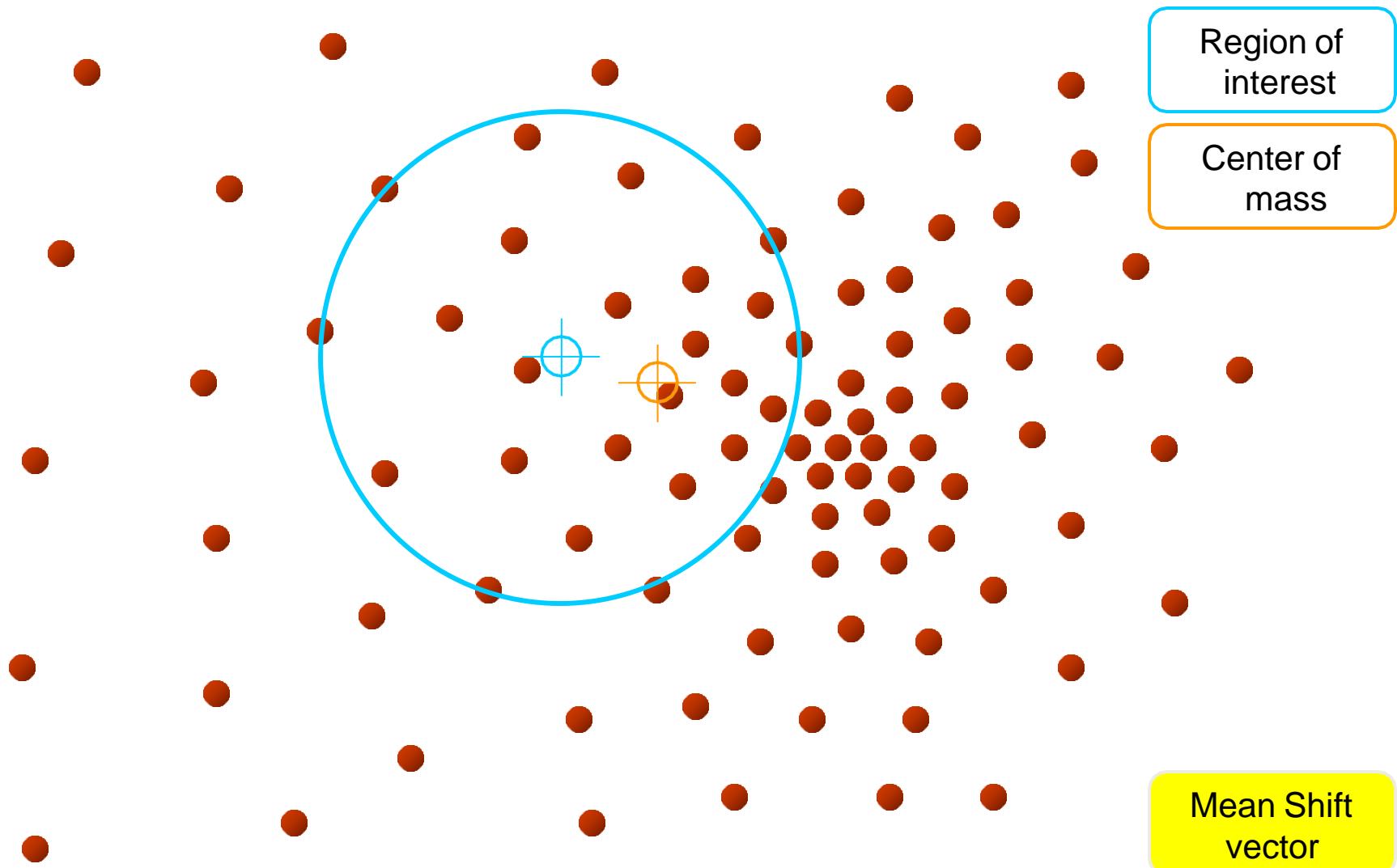


- Iterative Mode Search
  1. Initialize random seed, and window  $W$
  2. Calculate center of gravity (the “mean”) of  $W$ :  $\sum_{x \in W} xH(x)$
  3. Shift the search window to the mean
  4. Repeat Step 2 until convergence

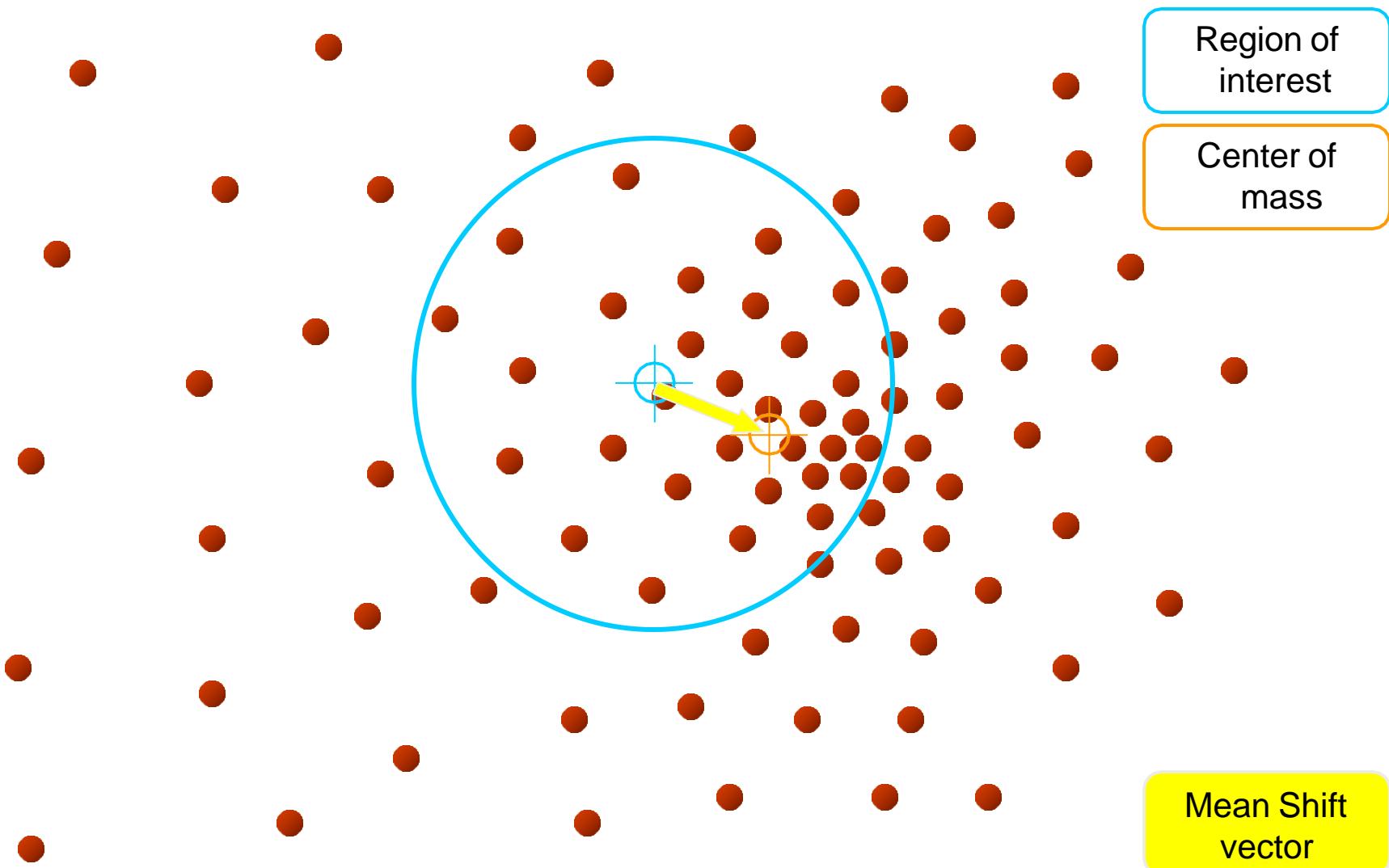
# Mean-Shift



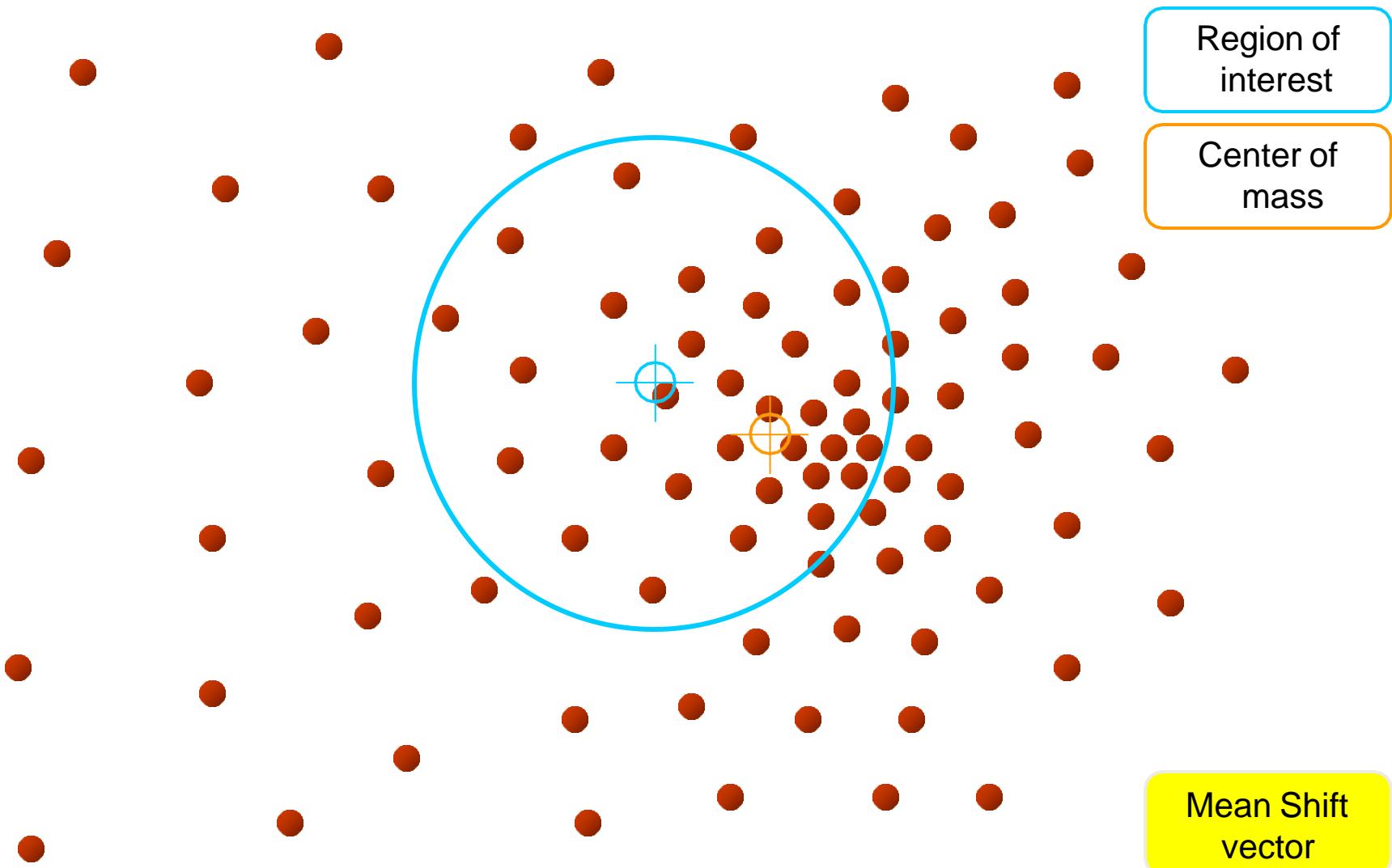
# Mean-Shift



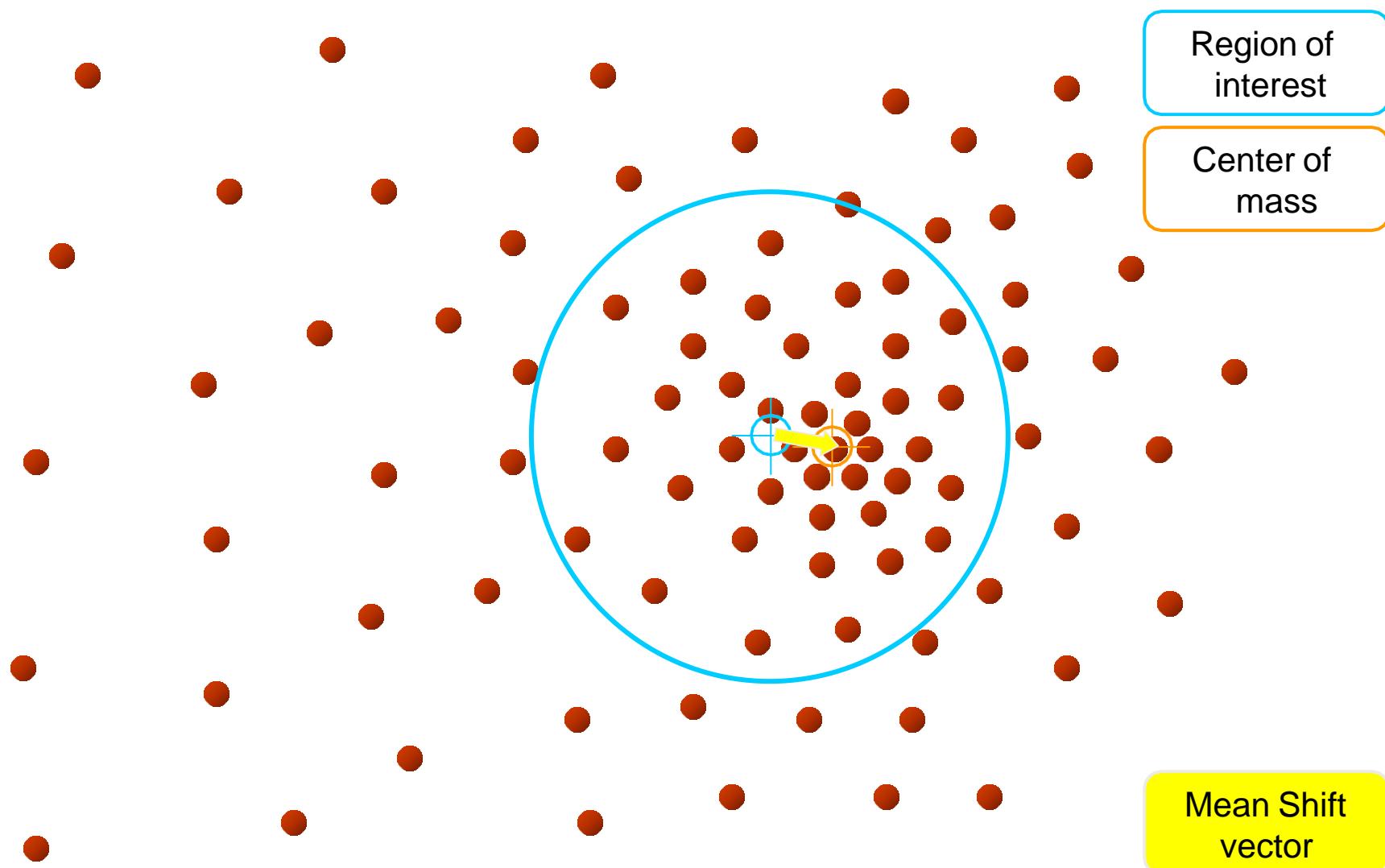
# Mean-Shift



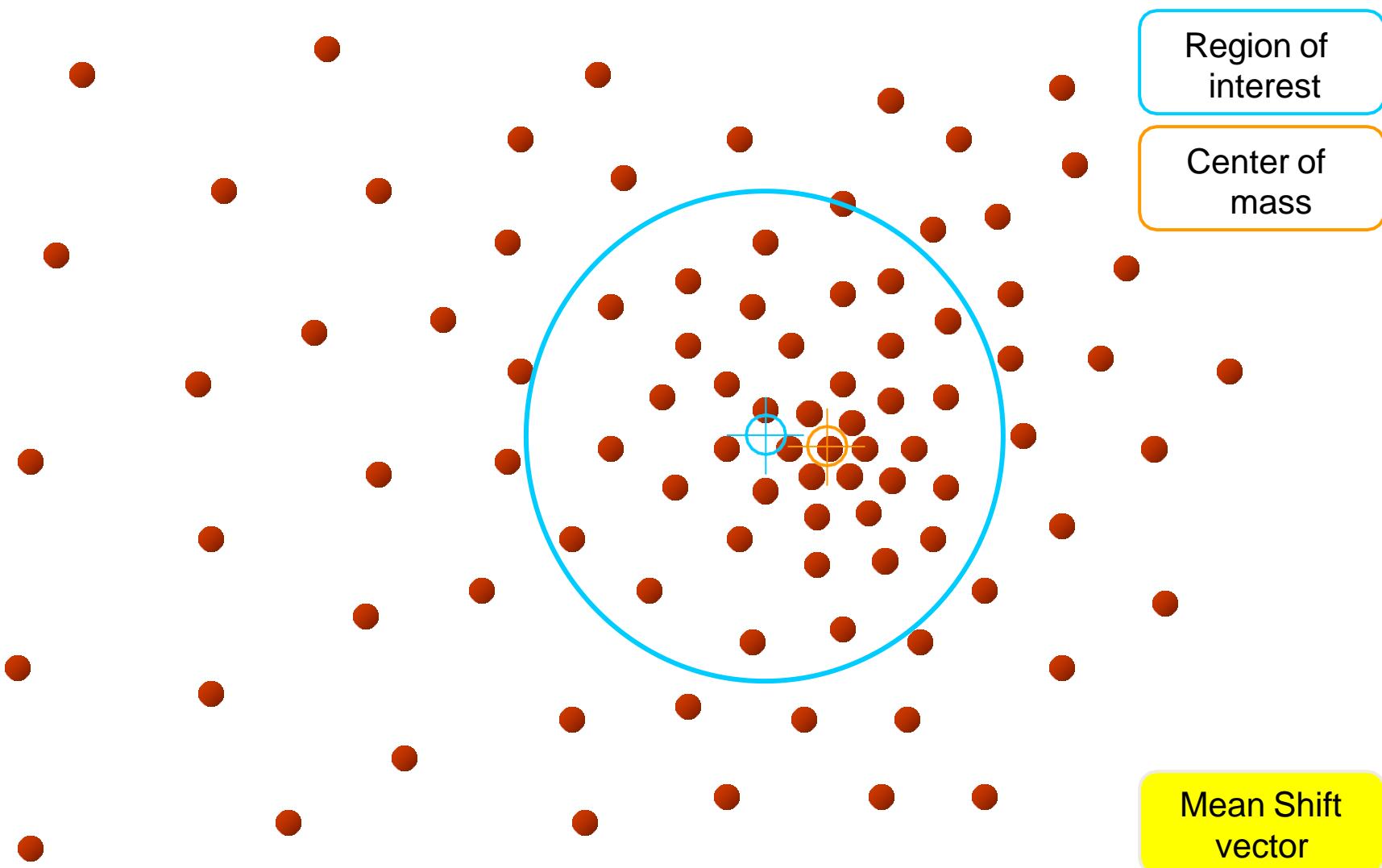
# Mean-Shift



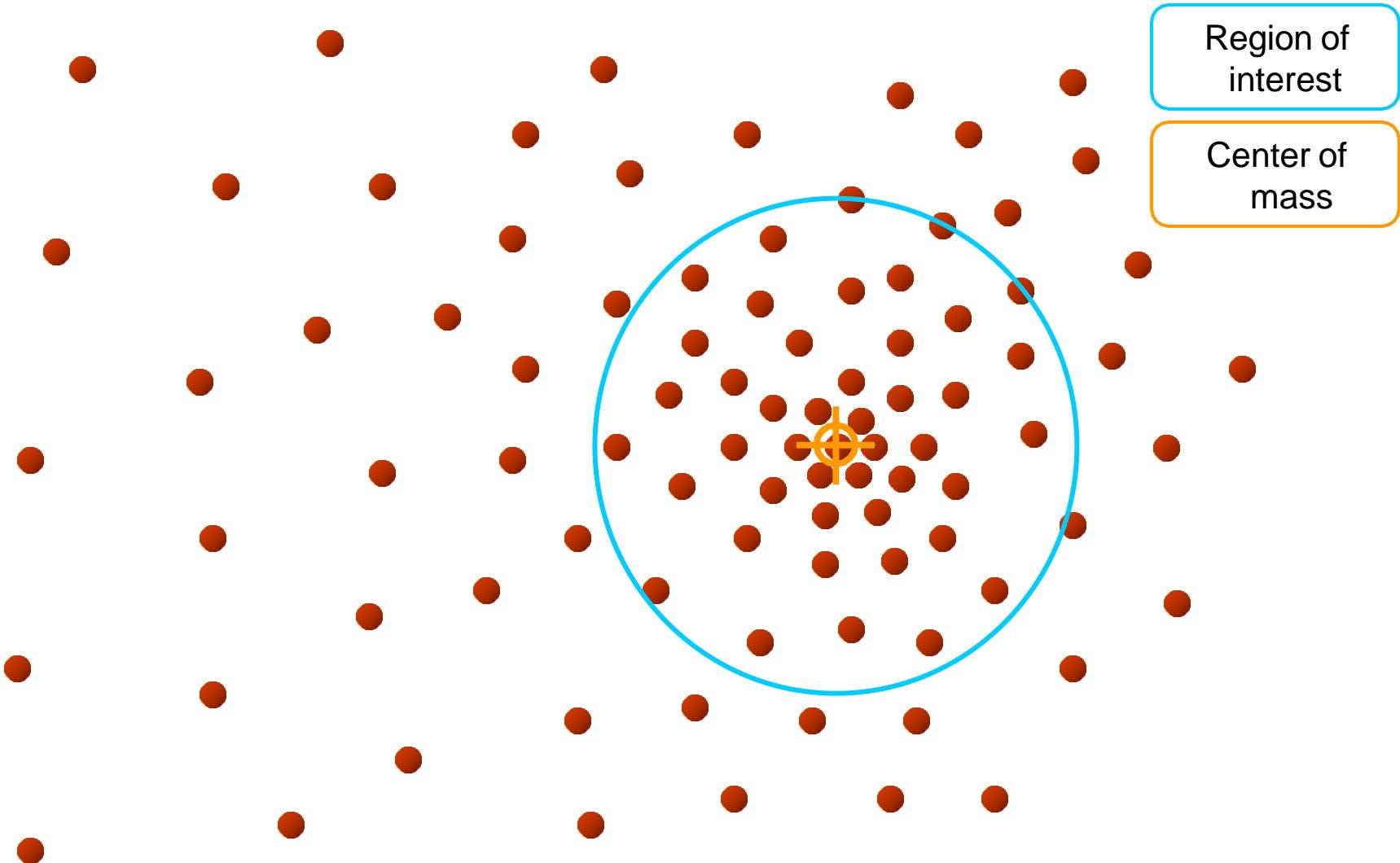
# Mean-Shift



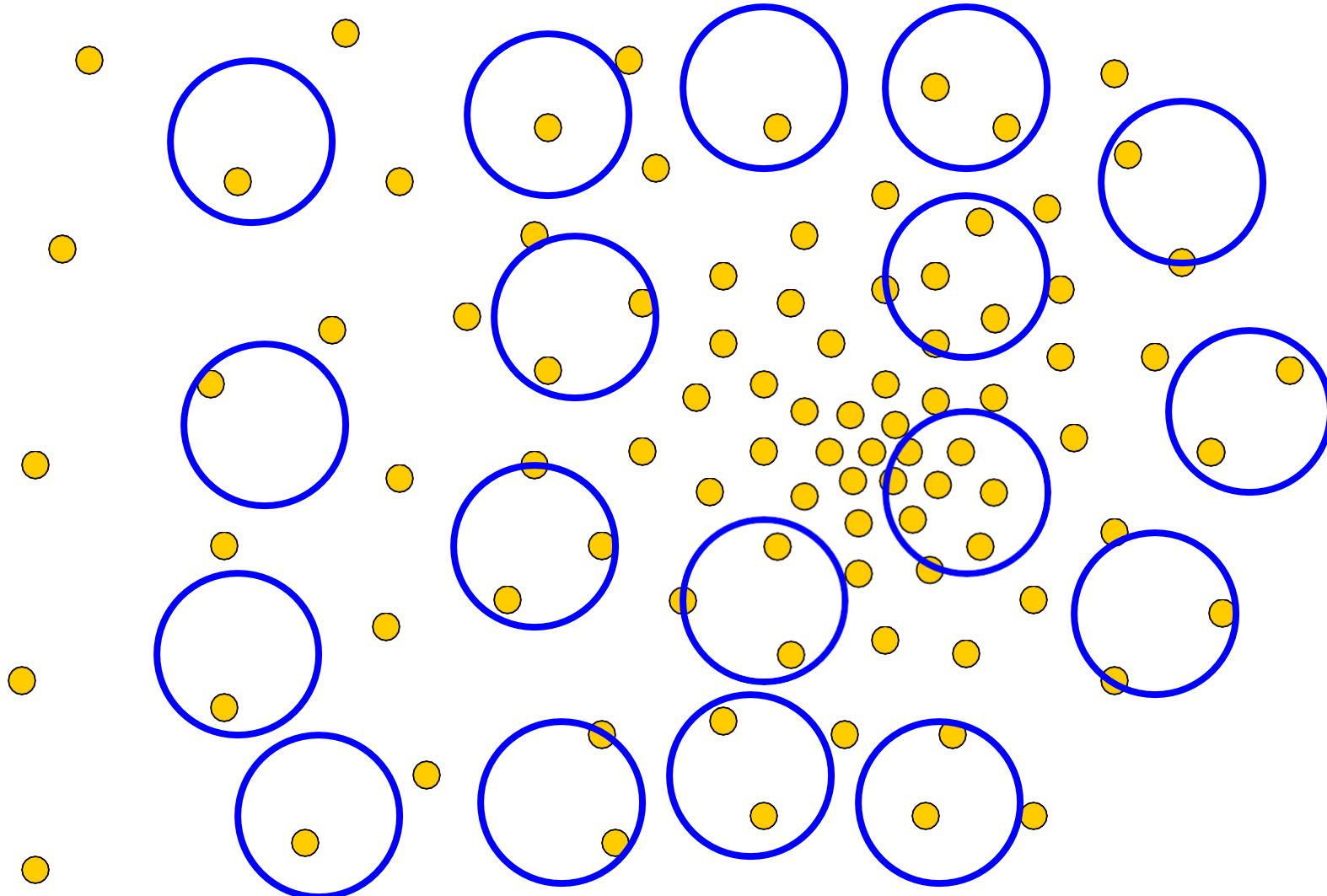
# Mean-Shift



# Mean-Shift



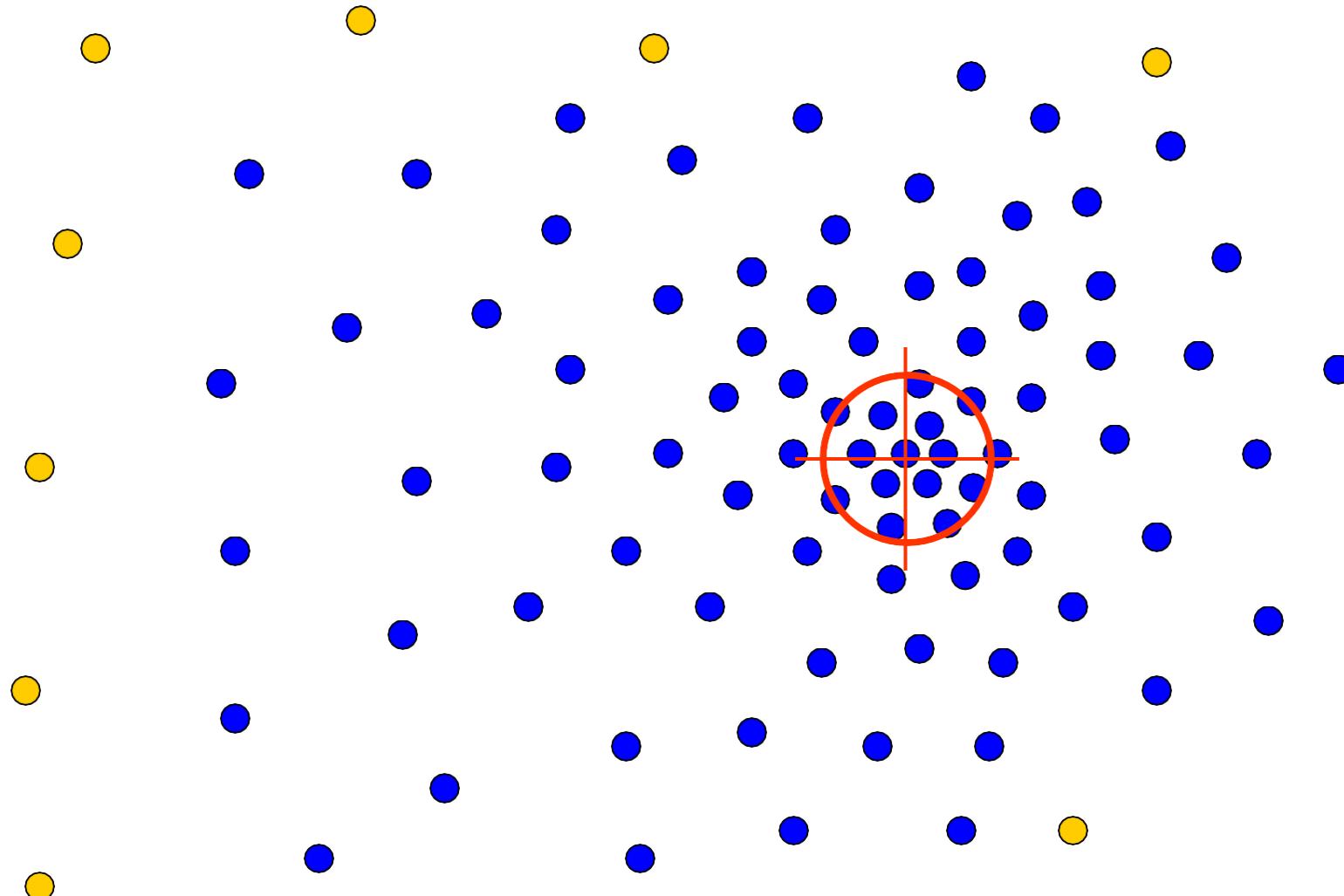
# Real Modality Analysis



Tessellate the space with windows

Run the procedure in parallel

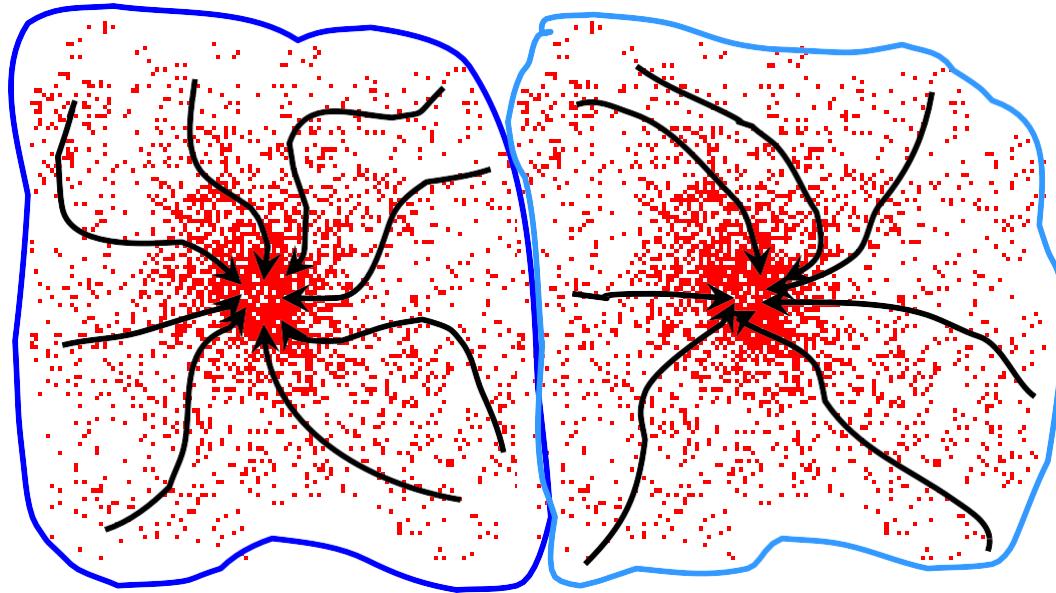
# Real Modality Analysis



The **blue** data points were traversed by the windows towards the mode.

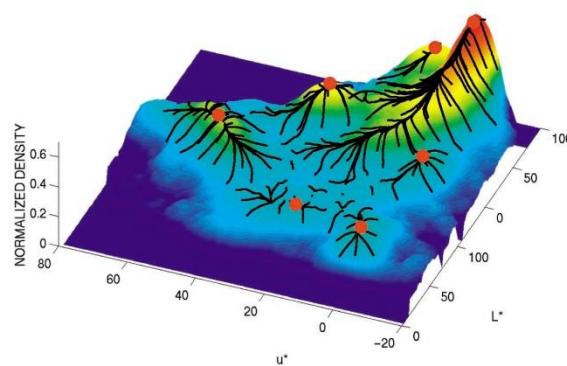
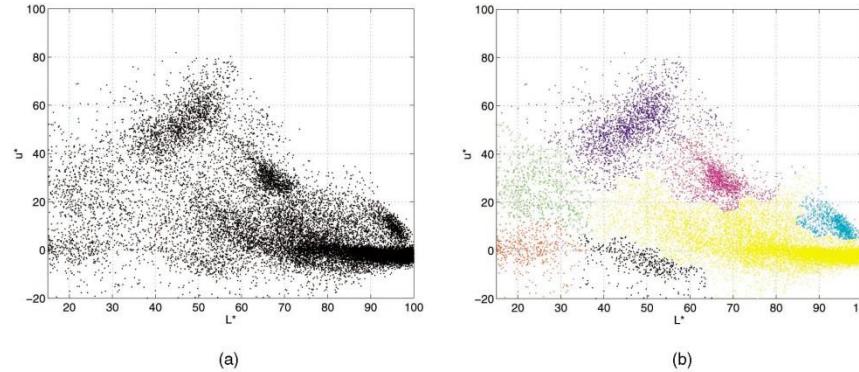
# Mean-Shift Clustering

- 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 pixel locations
- Perform mean shift for each window until convergence
- Merge windows that end up near the same “peak” or mode



# Mean-Shift Segmentation Results

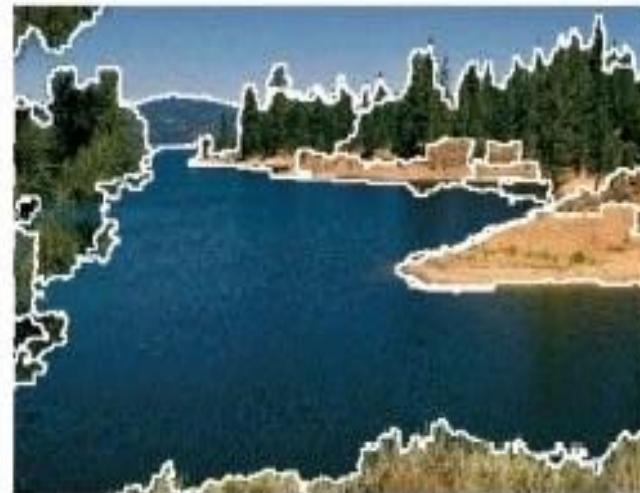
---



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

# More Results

---

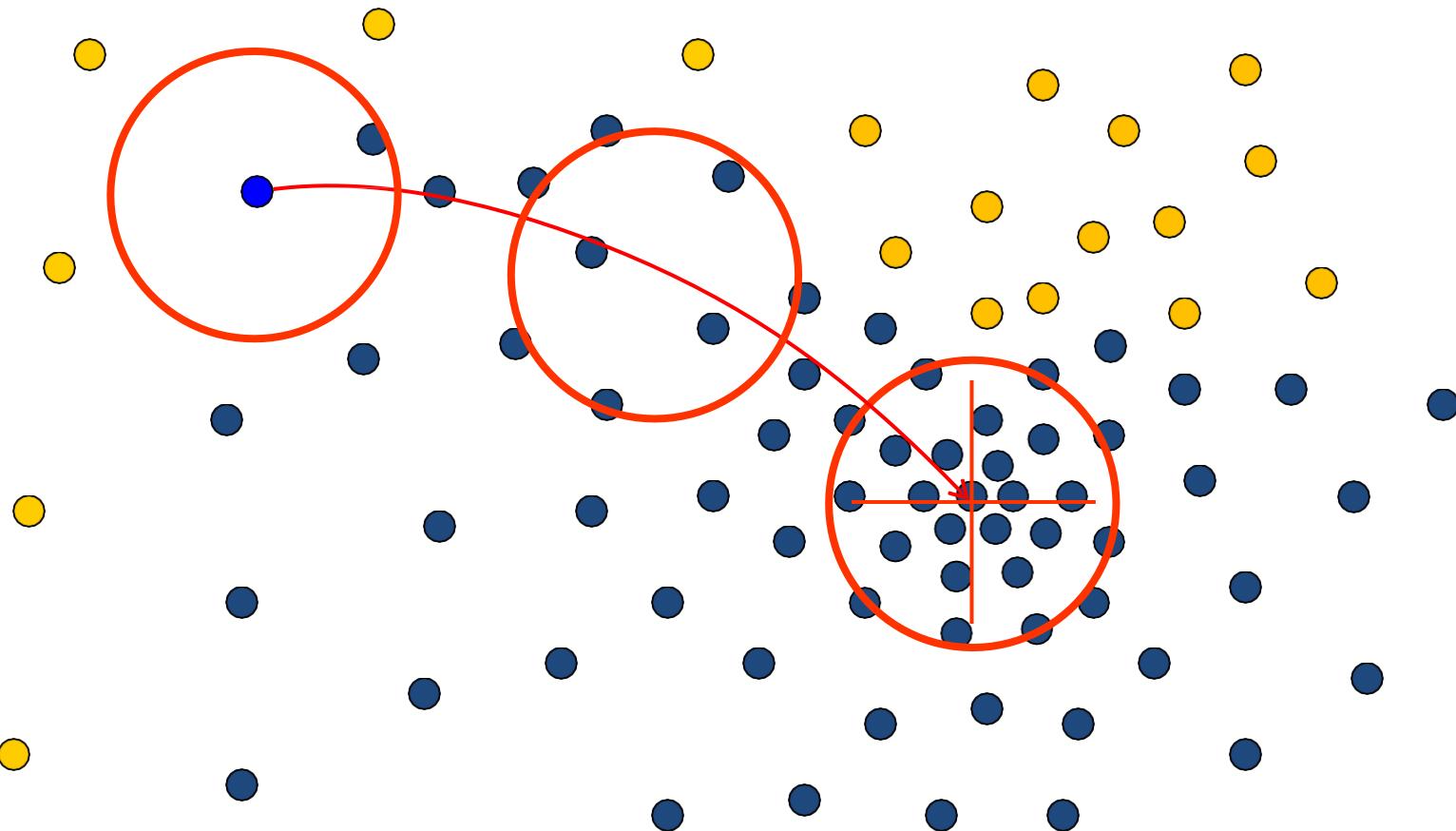


# More Results

---

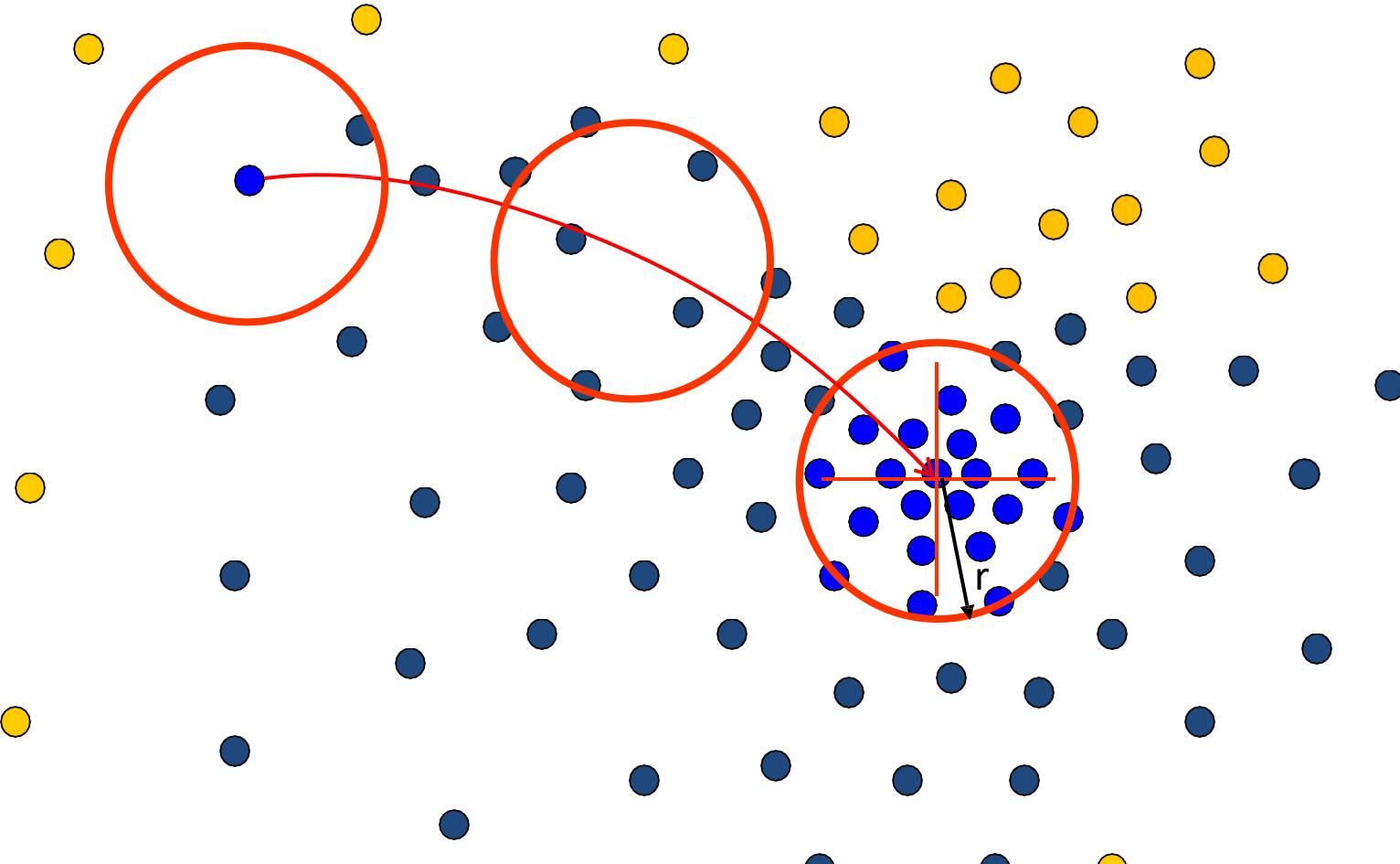


# Problem: Computational Complexity



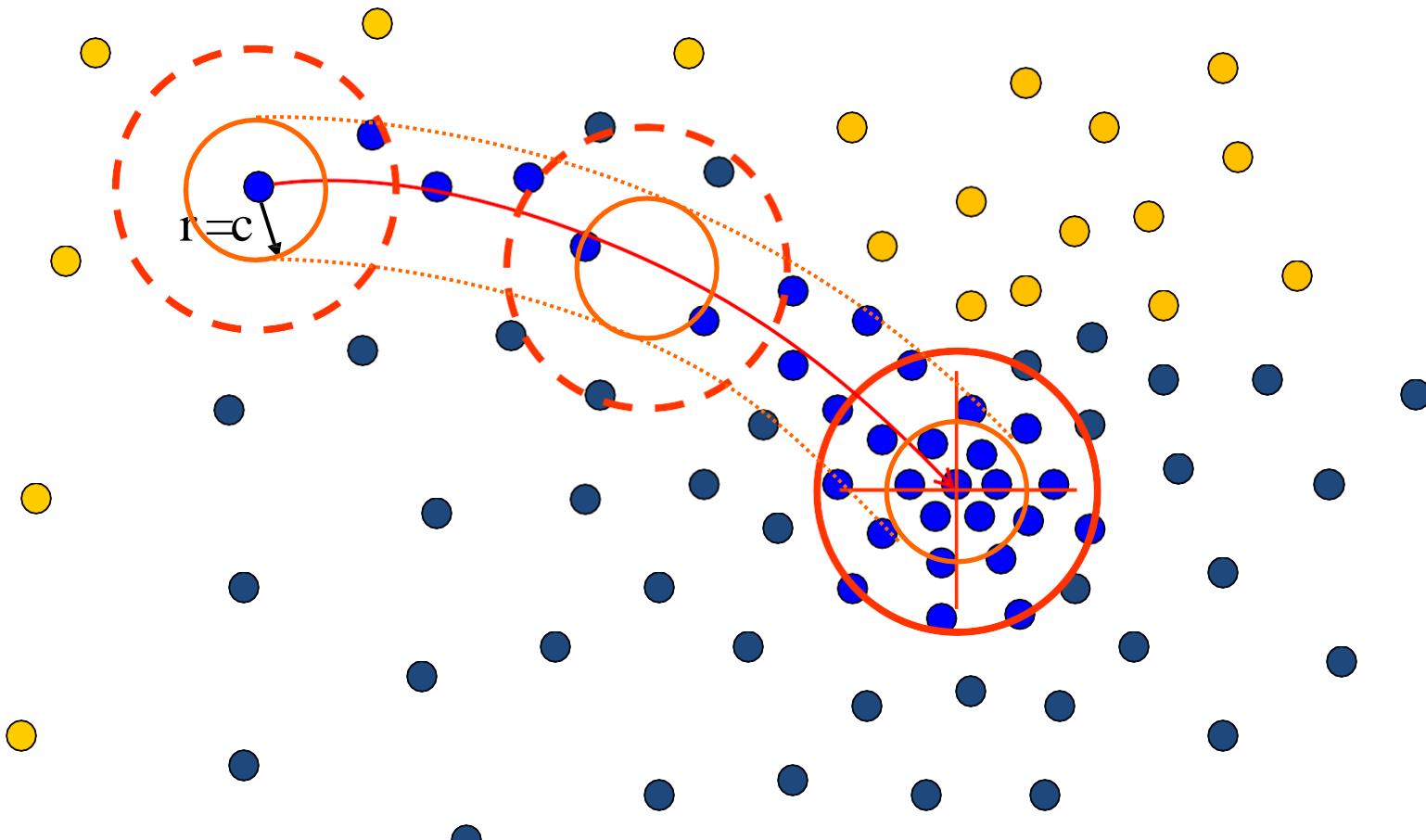
- Need to shift many windows...
- Many computations will be redundant.

# Speedups: Basin of Attraction



1. Assign all points within radius  $r$  of end point to the mode.

# Speedups



2. Assign all points within radius  $r/c$  of the search path to the mode  $\rightarrow$  reduce the number of data points to search.

# Technical Details

---

Given  $n$  data points  $\mathbf{x}_i \in \mathbb{R}^d$ , the multivariate kernel density estimate using a radially symmetric kernel<sup>1</sup> (e.g., Epanechnikov and Gaussian kernels),  $K(\mathbf{x})$ , is given by,

$$\hat{f}_K = \frac{1}{nh^d} \sum_{i=1}^n K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right), \quad (1)$$

where  $h$  (termed the *bandwidth* parameter) defines the radius of kernel. The radially symmetric kernel is defined as,

$$K(\mathbf{x}) = c_k k(\|\mathbf{x}\|^2), \quad (2)$$

where  $c_k$  represents a normalization constant.

Comaniciu & Meer, 2002

# Other Kernels

---

A kernel is a function that satisfies the following requirements:

1.  $\int_{R^d} \phi(x) = 1$

2.  $\phi(x) \geq 0$

Some examples of kernels include:

1. Rectangular

$$\phi(x) = \begin{cases} 1 & a \leq x \leq b \\ 0 & \text{else} \end{cases}$$

2. Gaussian

$$\phi(x) = e^{-\frac{x^2}{2\sigma^2}}$$

3. Epanechnikov

$$\phi(x) = \begin{cases} \frac{3}{4}(1 - x^2) & \text{if } |x| \leq 1 \\ 0 & \text{else} \end{cases}$$

# Technical Details

---

Taking the derivative of:

$$\hat{f}_K = \frac{1}{nh^d} \sum_{i=1}^n K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right)$$

$$\nabla \hat{f}(\mathbf{x}) = \underbrace{\frac{2c_{k,d}}{nh^{d+2}} \left[ \sum_{i=1}^n g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right) \right]}_{\text{term 1}} \underbrace{\left[ \frac{\sum_{i=1}^n \mathbf{x}_i g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^n g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right)} - \mathbf{x} \right]}_{\text{term 2}}, \quad (3)$$

where  $g(x) = -k'(x)$  denotes the derivative of the selected kernel profile.

- Term1: this is proportional to the density estimate at  $\mathbf{x}$  (similar to equation 1 from two slides ago).
- Term2: this is the mean-shift vector that points towards the direction of maximum density.

Comaniciu & Meer, 2002

# Technical Details

---

Finally, the mean shift procedure from a given point  $\mathbf{x}_t$  is:

1. Compute the mean shift vector  $\mathbf{m}$ :

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

2. Translate the density window:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{m}(\mathbf{x}_i^t).$$

3. Iterate steps 1 and 2 until convergence.

$$\nabla f(\mathbf{x}_i) = 0.$$

# Summary Mean-Shift

---

- Pros
  - General, application-independent tool
  - Model-free, does not assume any prior shape (spherical, elliptical, etc.) on data clusters
  - Just a single parameter (window size  $h$ )
    - $h$  has a physical meaning (unlike k-means)
  - Finds variable number of modes
  - Robust to outliers
- Cons
  - Output depends on window size
  - Window size (bandwidth) selection is not trivial
  - Computationally (relatively) expensive (~2s/image)
  - Does not scale well with dimension of feature space



*Next time:*

## ***Dimensionality Reduction***

**Pattern Recognition and Computer Vision**

Guanbin Li,

School of Computer Science and Engineering, Sun Yat-Sen University