

# Lecture 10

## Segmentation and clustering

# Administrative

A2 is out

- Due April 28th
- Date moved back

A3 is out

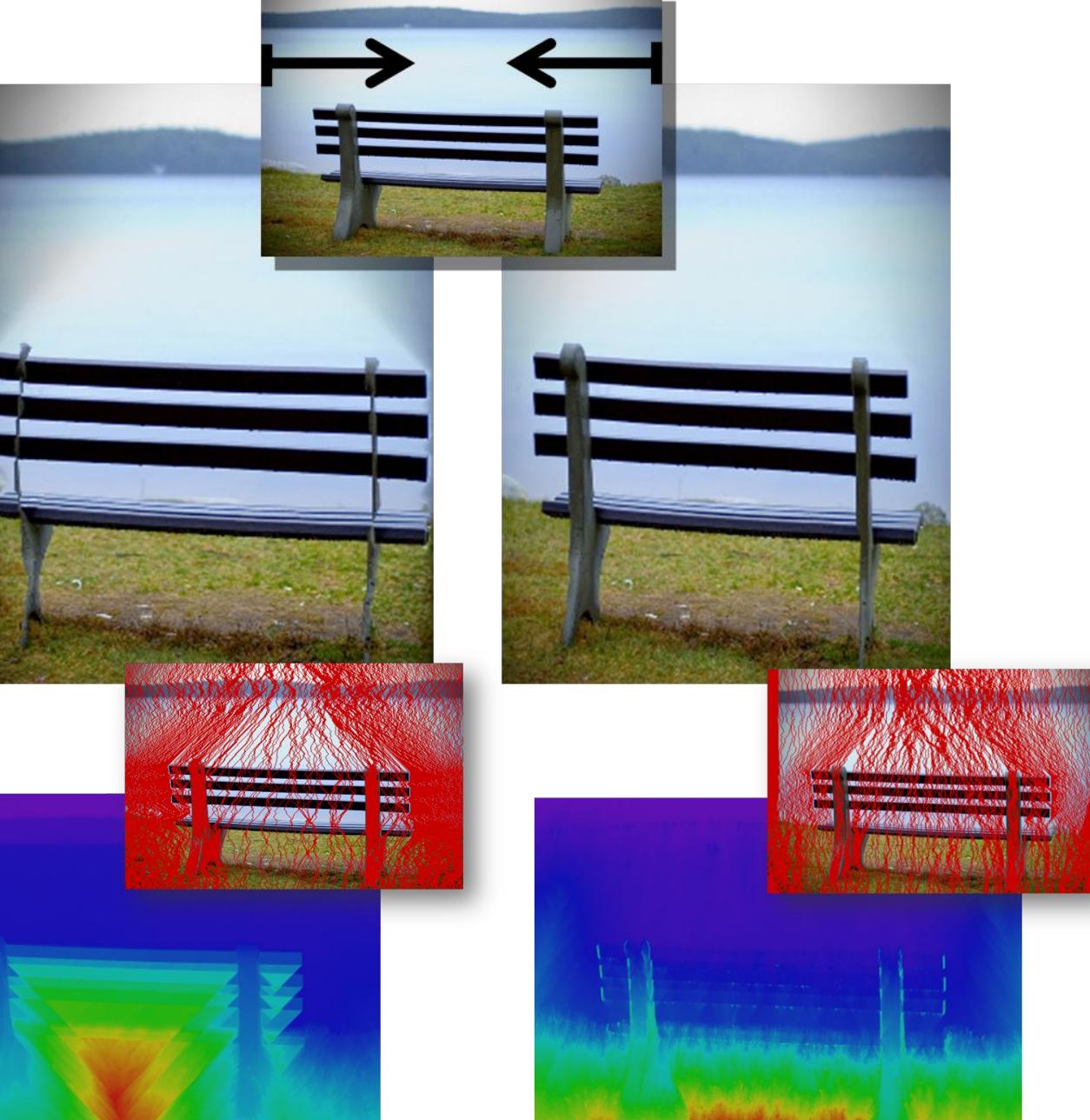
- Due May 9th

# Administrative

## Recitation

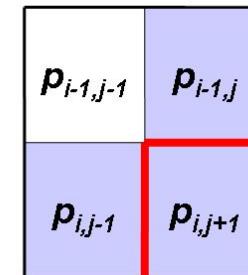
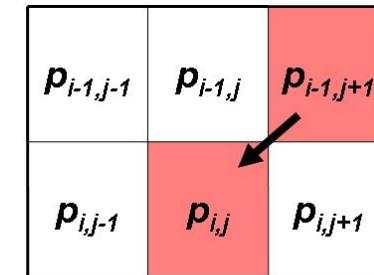
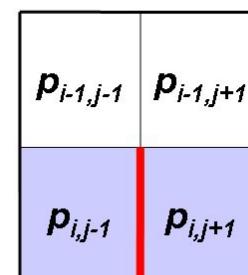
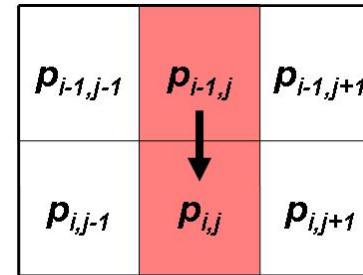
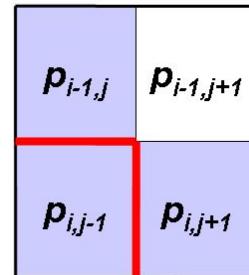
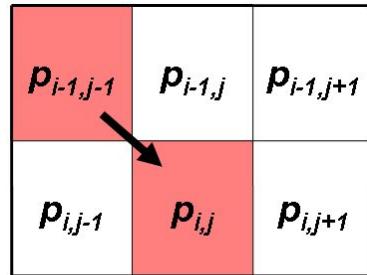
- Xiaojuan Wang
- Panorama (part of your A2)
- detector, descriptor, RANSAC recap

So far



# So far: Seam carving with pixel energies

$$M(i, j) = E(i, j) + \min \begin{cases} M(i - 1, j - 1) + C_L(i, j) \\ M(i - 1, j) + C_V(i, j) \\ M(i - 1, j + 1) + C_R(i, j) \end{cases}$$



# Today's agenda

- Introduction to segmentation and clustering
- Gestalt theory for perceptual grouping
- Graph-based oversegmentation
- Agglomerative clustering
- K-means
- Mean-shift

Reading:

Szeliski, 2<sup>nd</sup> edition, Chapter 7.5

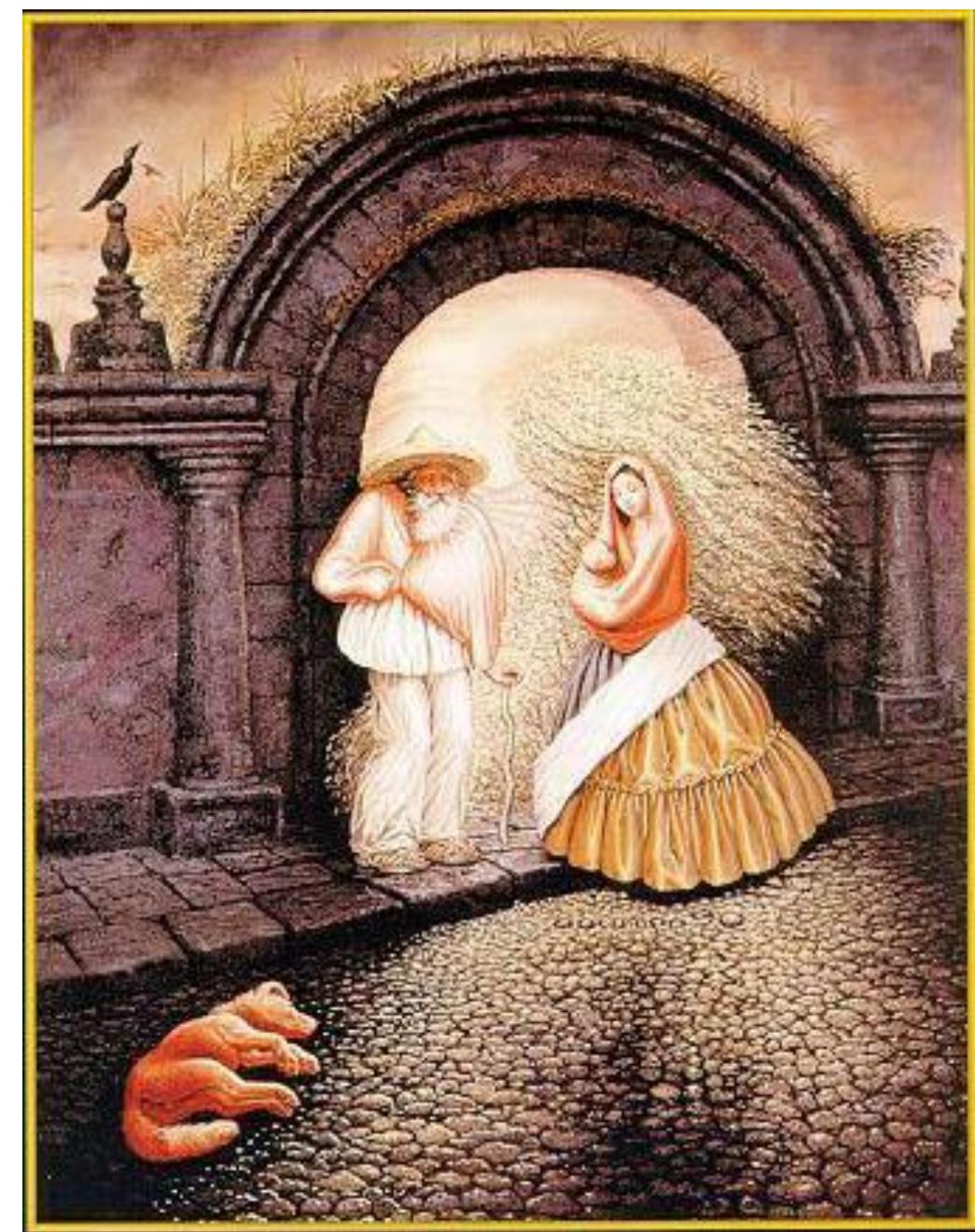
# Today's agenda

- Introduction to segmentation and clustering
- Gestalt theory for perceptual grouping
- Graph-based oversegmentation
- Agglomerative clustering
- K-means
- Mean-shift

Reading:

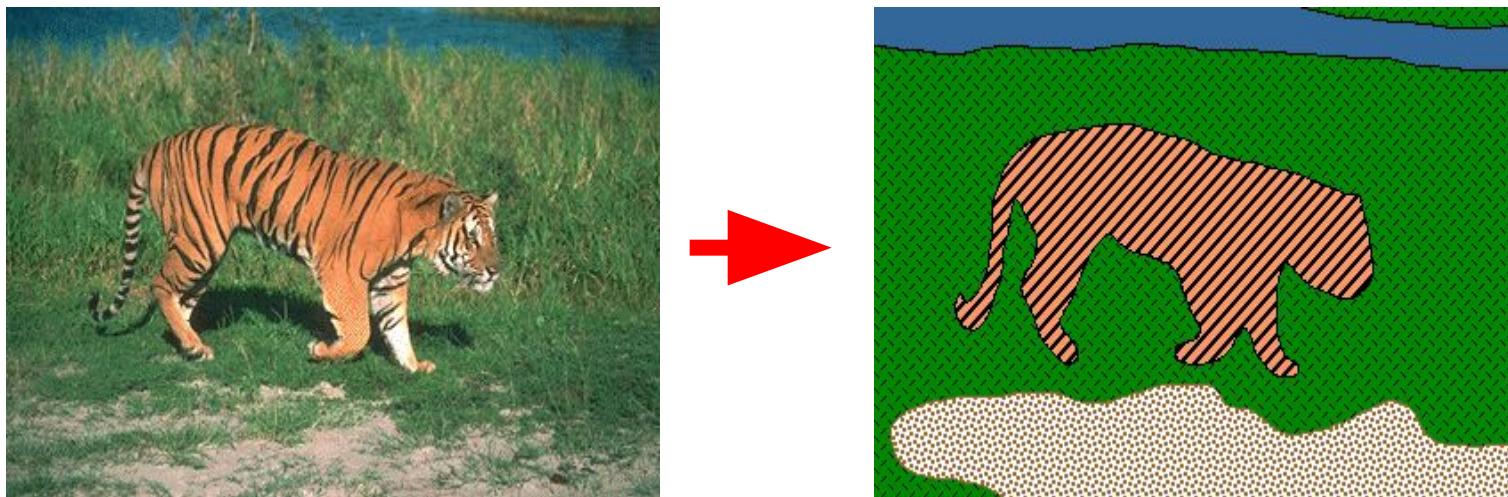
Szeliski, 2<sup>nd</sup> edition, Chapter 7.5

Q. What do you see?



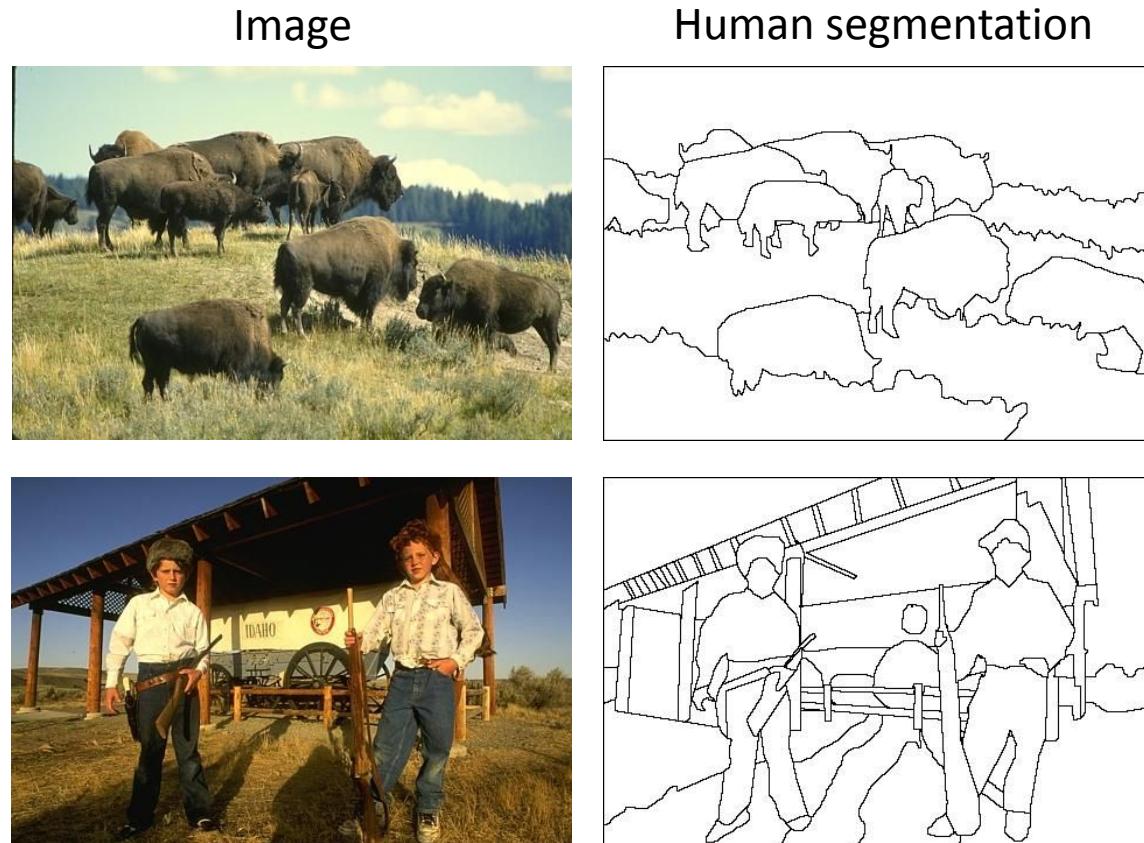
# Image Segmentation

- Goal: identify groups of pixels that go together



# The Goals of Segmentation

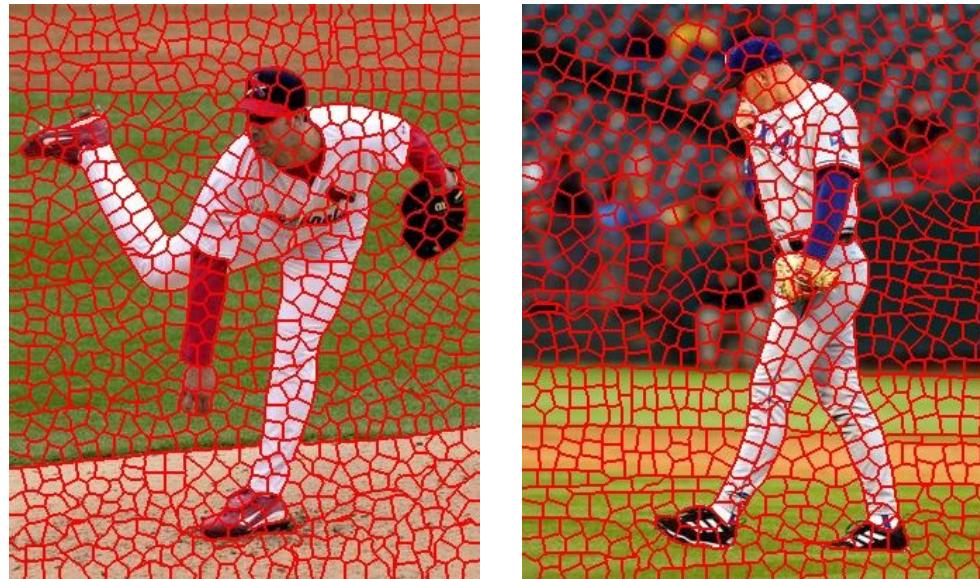
- Separate image into coherent “objects”



# The Goals of Segmentation

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

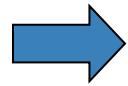
“superpixels”



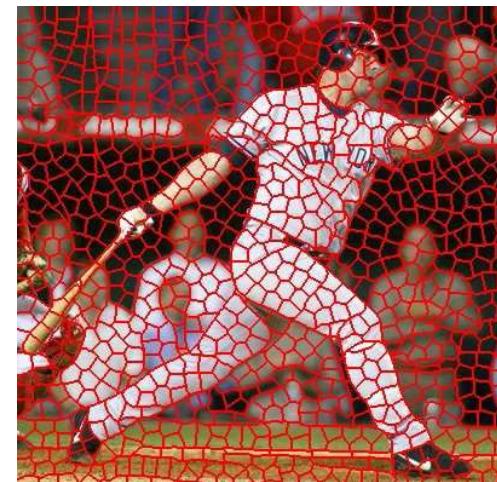
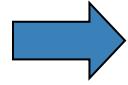
# Segmentation for feature support



# Segmentation for efficiency



[Felzenszwalb and Huttenlocher 2004]



[Hoiem et al. 2005, Mori 2005]

[Shi and Malik 2001]

Segmentation is used in Adobe  
photoshop to remove background

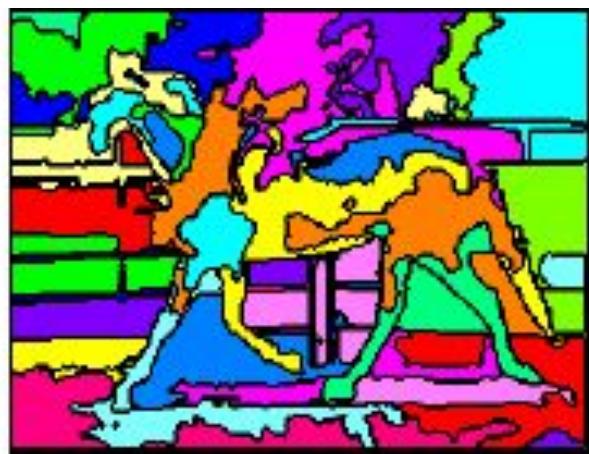


Rother et al. 2004

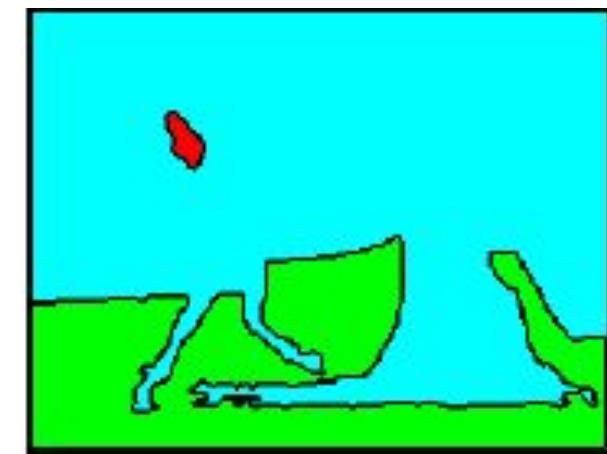
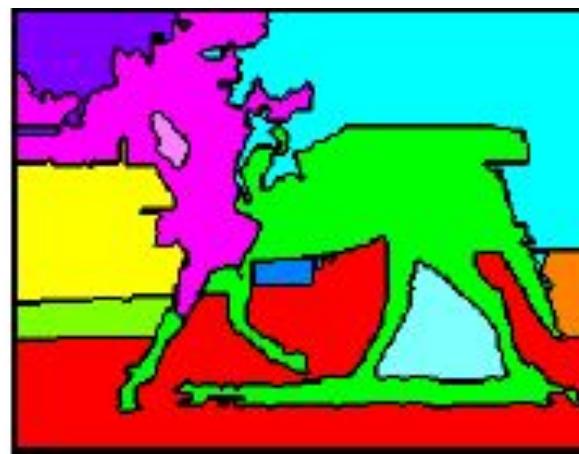
# Segment Anything [2023]



# Levels of segmentations



Over-segmentation



Under-segmentation

# 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
  - Find clusters of pixels
  - Represent each cluster of pixels with a HoG feature
- **Counting**
  - Histograms of texture, color, SIFT vectors
- **Foreground-background separation**
  - Separate the image into different regions
- **Prediction**
  - Images in the same cluster may have the same labels

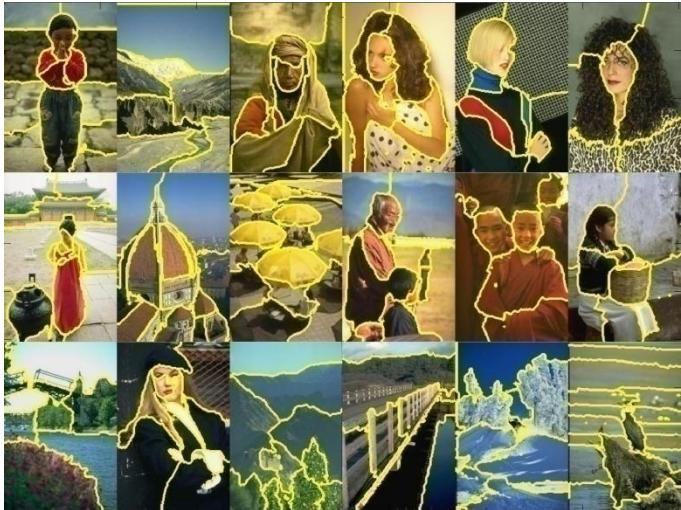
# How do we cluster?

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

# General ideas

- **Tokens**
  - Things that can be grouped together
  - (e.g. pixels, points, surface elements, etc., 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



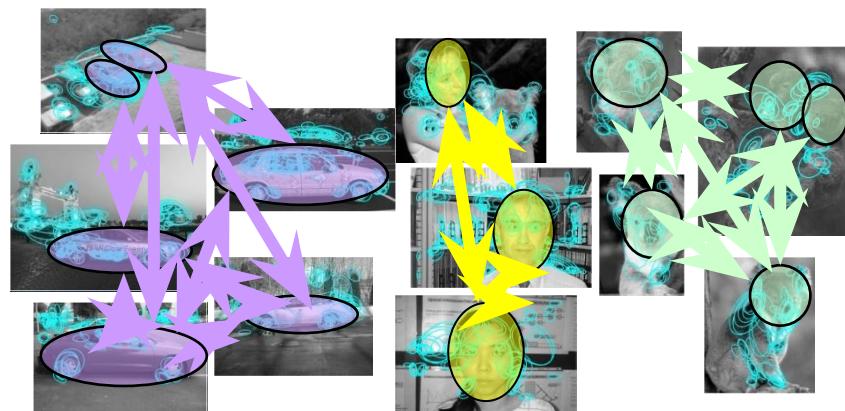
Determining image regions

What things should  
be grouped?

What cues  
indicate groups?



Grouping video frames into shots



Object-level grouping

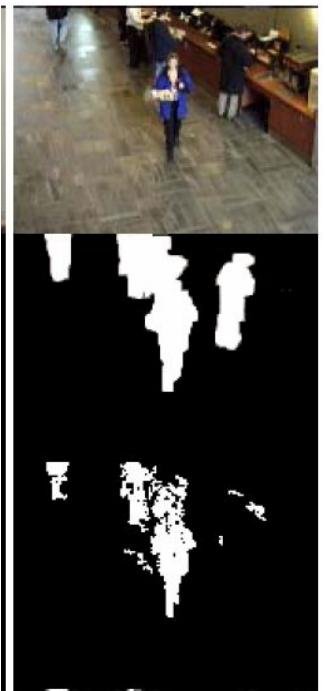
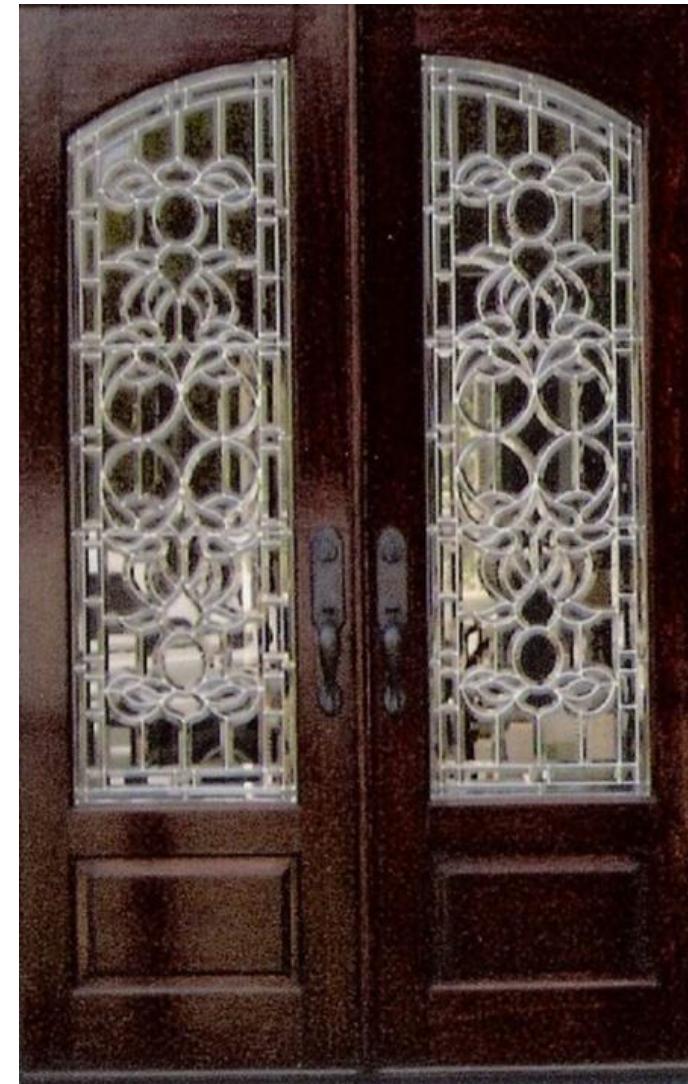
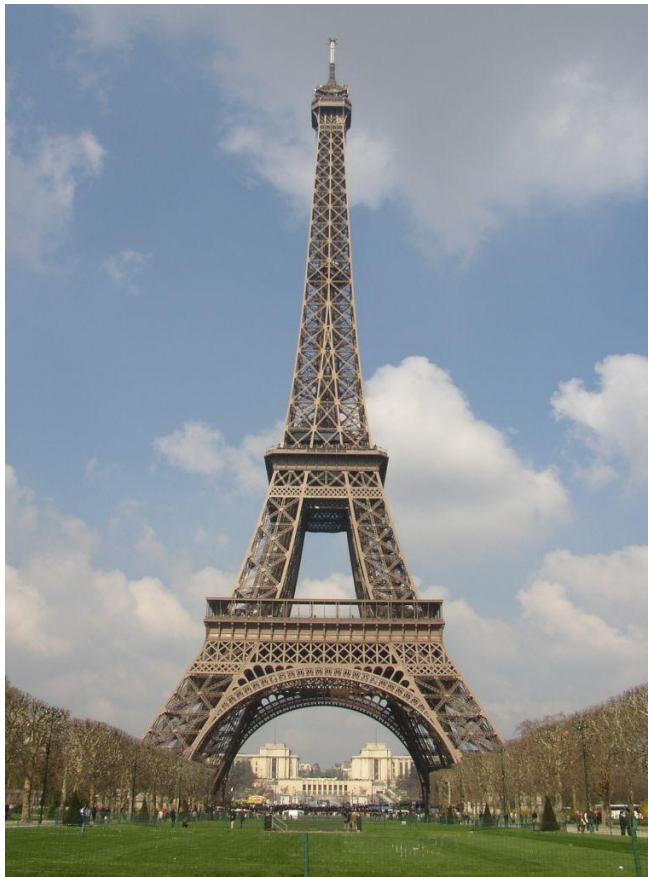


Figure-ground

# Similarity



# Symmetry



Slide credit: Kristen Grauman

Ranjay Krishna, Jieyu Zhang

Lecture 10 - 23

April 25, 2024

# Common Fate



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



(c) 2005 Heiko Burkhardt, lillano.com

# Proximity

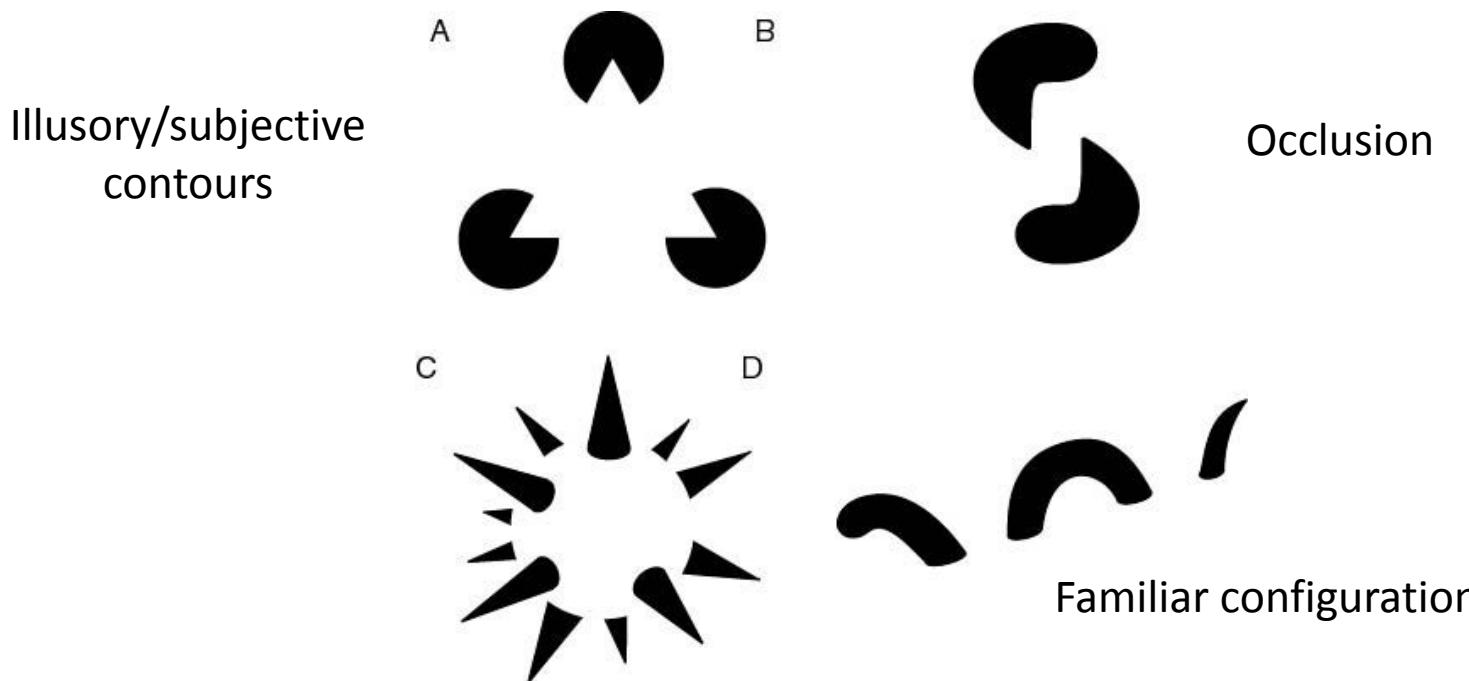


# What will we learn today?

- Introduction to segmentation and clustering
- **Gestalt theory for perceptual grouping**
- Graph-based oversegmentation
- Agglomerative clustering
- K-means
- Mean-shift

# The Gestalt School

- Grouping is key to visual perception
- Elements in a collection can have properties that result from different **relationships (space, affordance, etc.)**
  - “The whole is greater than the sum of its parts”

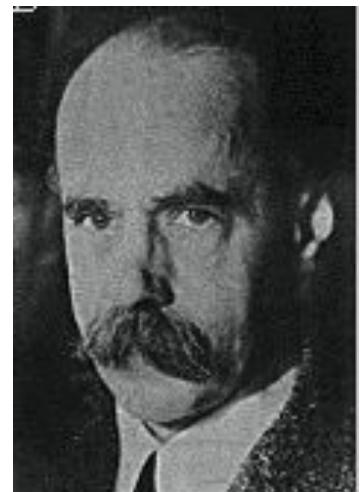


# Gestalt Theory

- 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)

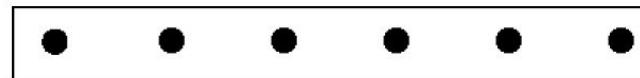
*"I stand at the window and see a house, trees, sky.  
Theoretically I might say there were 327  
brightnesses and nuances of colour. Do I have "327"?  
No. I have sky, house, and trees."*

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



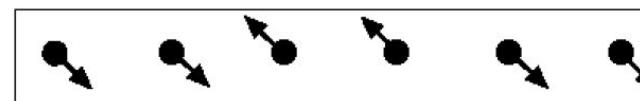
Proximity



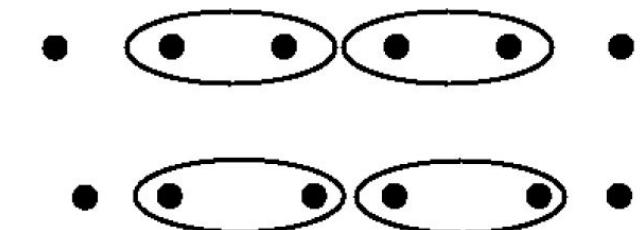
Similarity



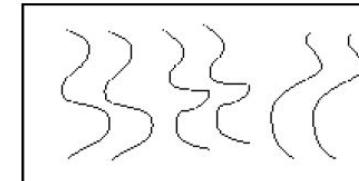
Similarity



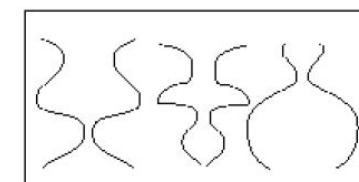
Common Fate



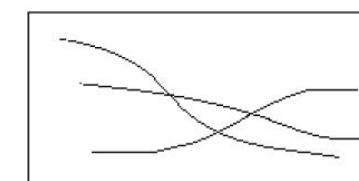
Common Region



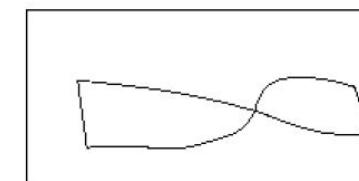
Parallelism



Symmetry



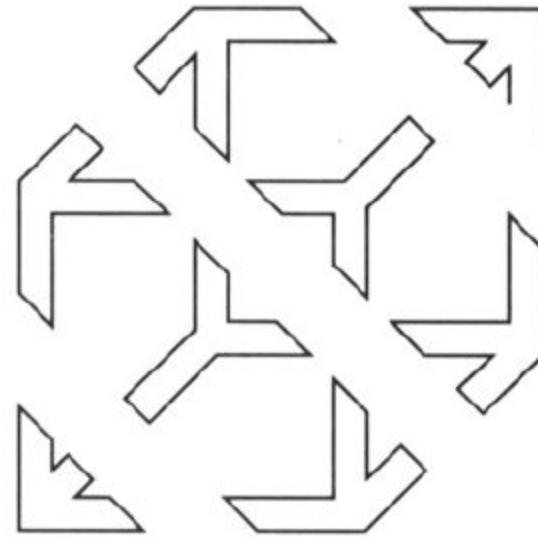
Continuity



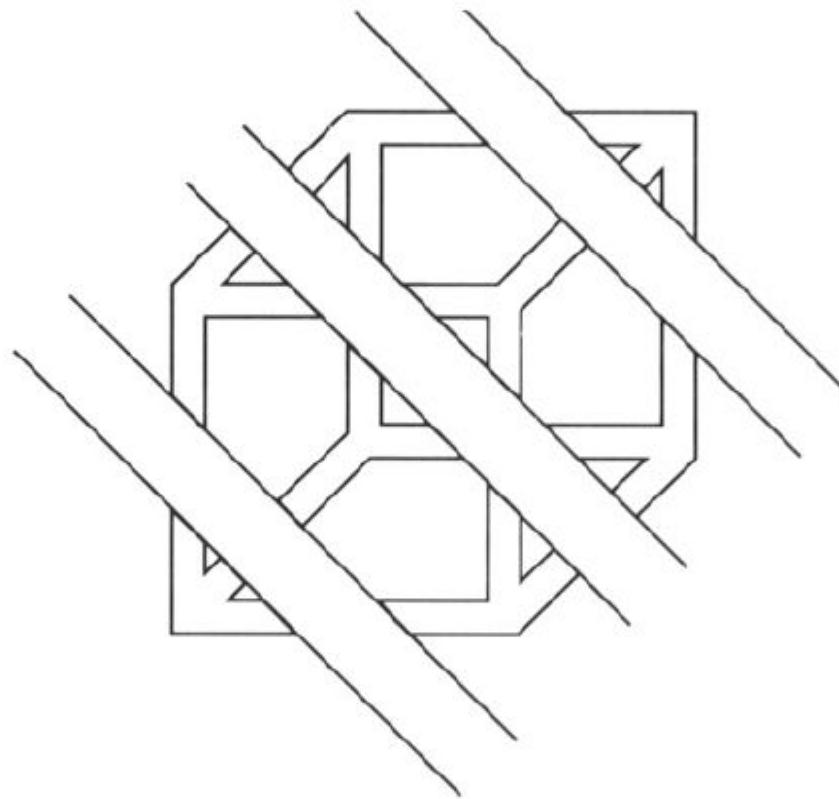
Closure

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

# Continuity through Occlusion Cues

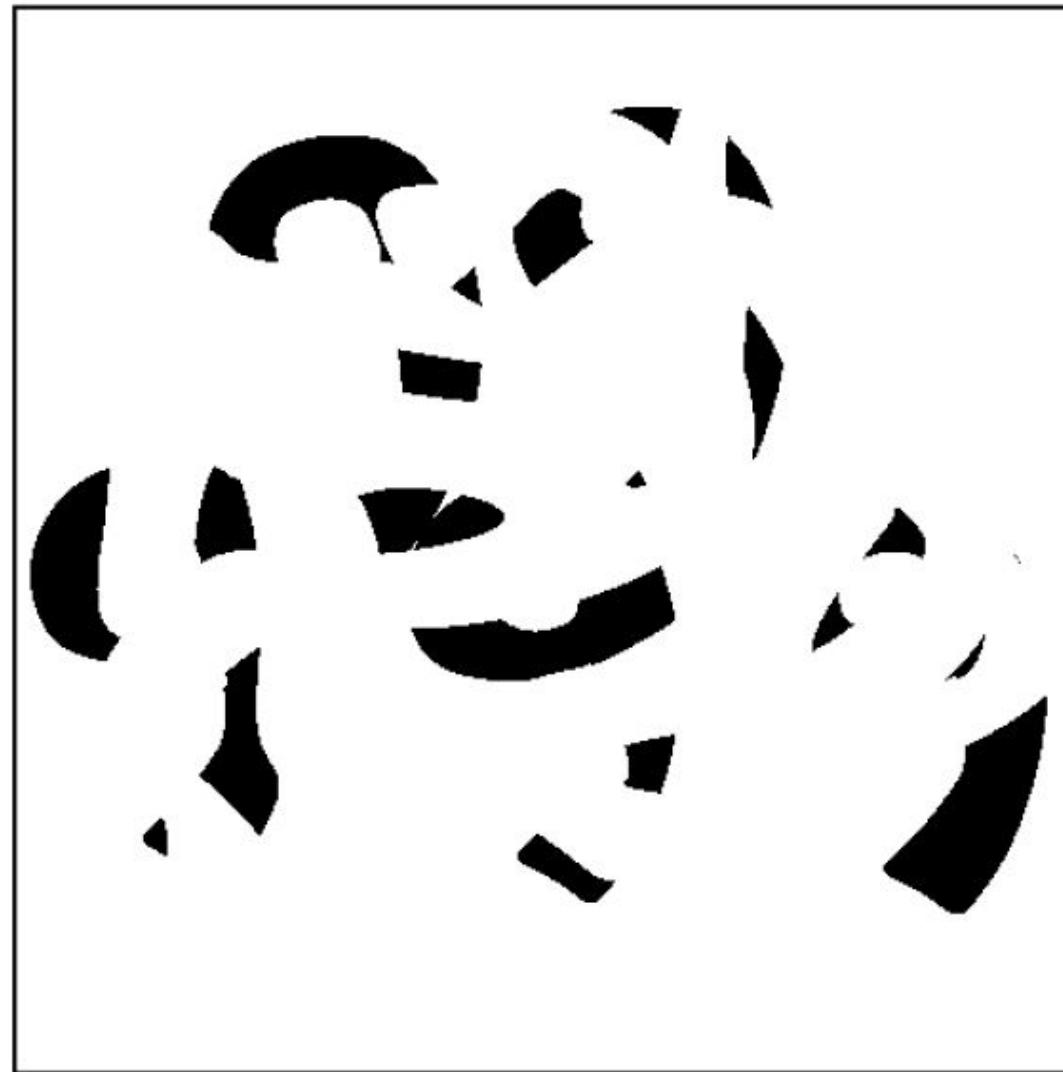


# Continuity through Occlusion Cues



Continuity, explanation by occlusion

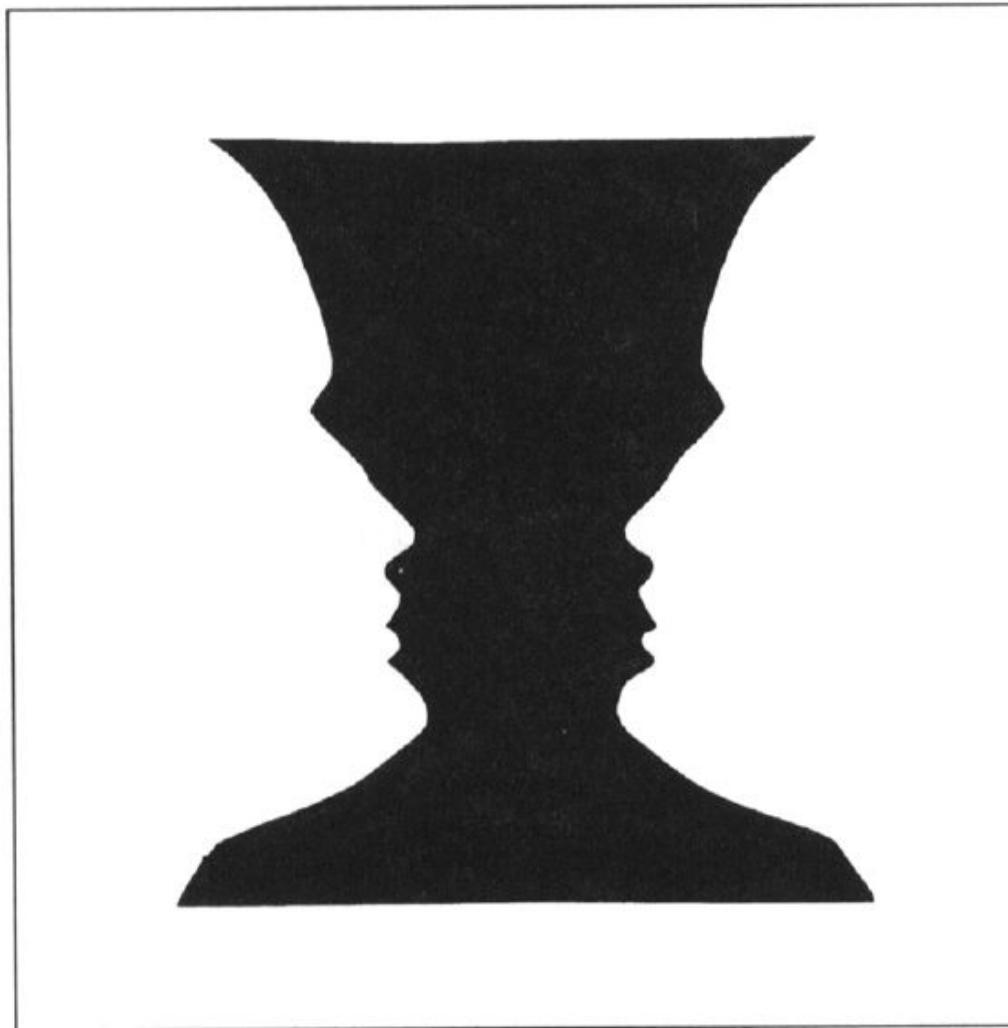
# Continuity through Occlusion Cues



# Continuity through Occlusion Cues



# Figure-Ground Discrimination



# The Ultimate Gestalt?



# What will we learn today?

- Introduction to segmentation and clustering
- Gestalt theory for perceptual grouping
- **Graph-based oversegmentation**
- Agglomerative clustering
- K-means
- Mean-shift



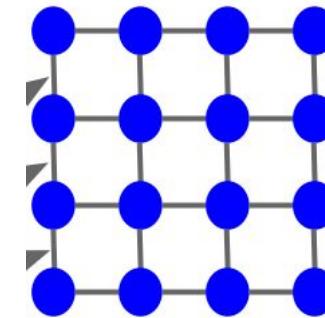
# Over-segmenting images

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



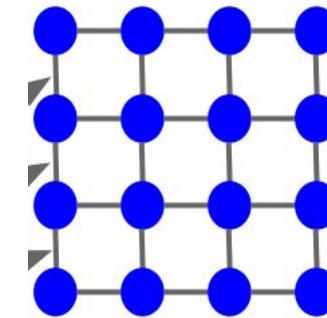
# Image as a Graph - Features and weights

- Every pixel is connected to its **8 neighboring pixels**
- The edges between neighbors have **weights** that are determined by the distance between them.
- Edge weights between pixels are determined using  $\text{dist}(x, x')$  distance in feature space.
  - where  $p$  and  $p'$  are two neighboring pixels
- Q. What is a good feature space?



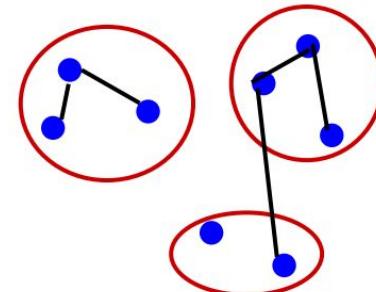
# What are good pixel features?

- Use **RGB values**?
  - $v = [r, g, b]$
  - It is 3-dimensional
- Use **location**?
  - $v = [x, y]$
  - 2-dim
- Use **RGB + location**?
  - $v = [x, y, r, g, b]$
  - 5-dim
- Use **gradient magnitude**?
  - $v = [df/dx, df/dy]$
  - 2-d



# 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
- $\text{dist}(v_i, v_j)$  is the weight/distance of the edge between nodes  $v_i$  and  $v_j$ .
- $S$  is a segmentation of a graph  $G$  such that  $G' = (V, E')$  where  $E' \subset E$ .
  - That is, **we keep all vertices**, but **select a subset  $E'$**  from all initial edges  $E$ .
- $S$  divides  $G$  into  $G'$  such that it contains distinct clusters  $C$ .



# Weights of edges: distance measure

Clustering is an unsupervised learning method. Given items  $v_1, v_2, \dots, v_n \in \mathcal{R}^D$ , 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 (or similarity) between  $x$  and  $x'$  is a real number:

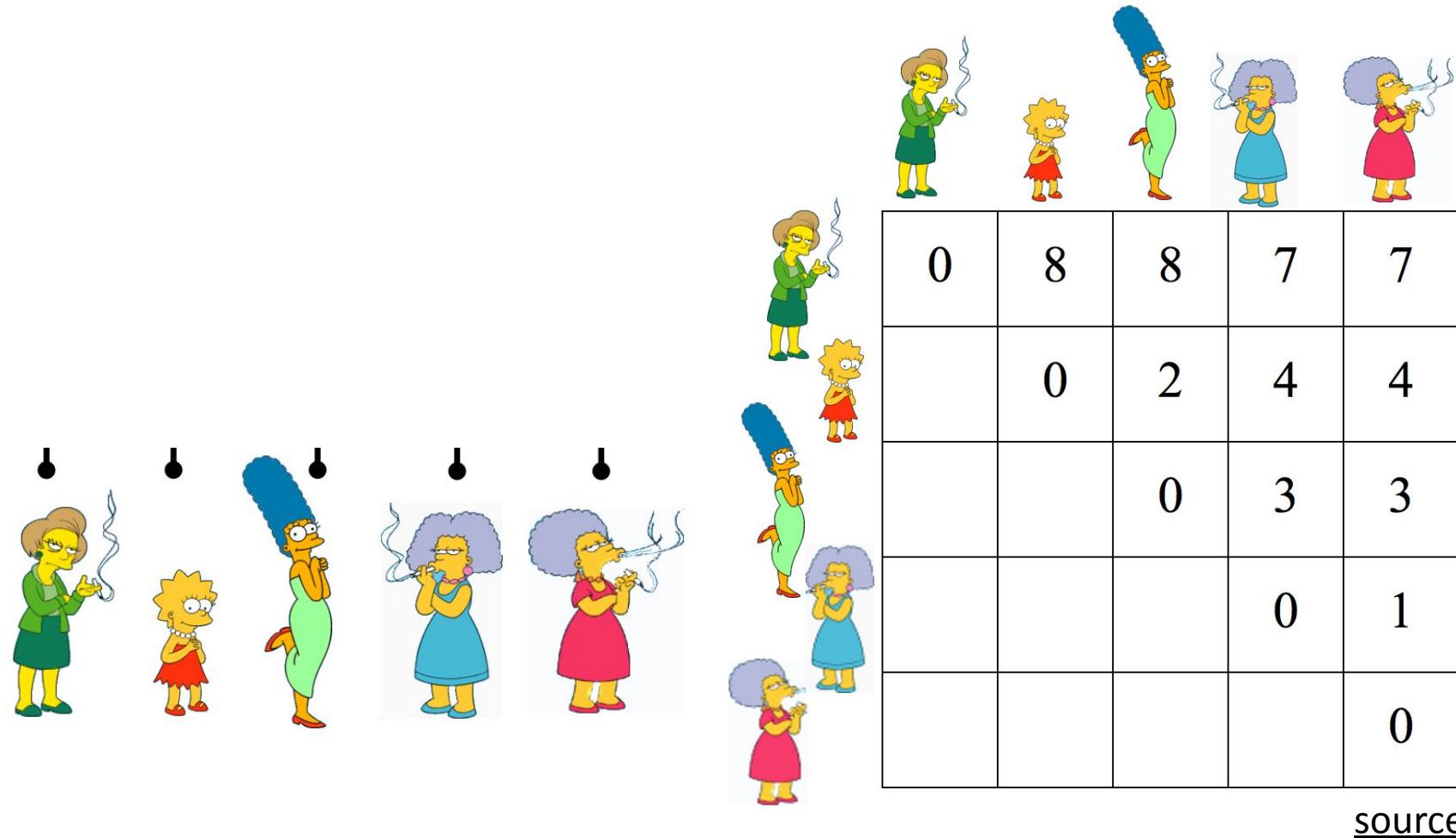
- The Euclidean distance is defined as  $dist(v_1, v_2) = \sqrt{\sum_i (v_{1i} - v_{2i})^2}$
- In contrast, the cosine similarity measure would be

$$\begin{aligned} dist(v_1, v_2) &= 1 - \cos(v_1, v_2) \\ &= 1 - \frac{v_1^T v_2}{\|v_1\| \cdot \|v_2\|} \end{aligned}$$

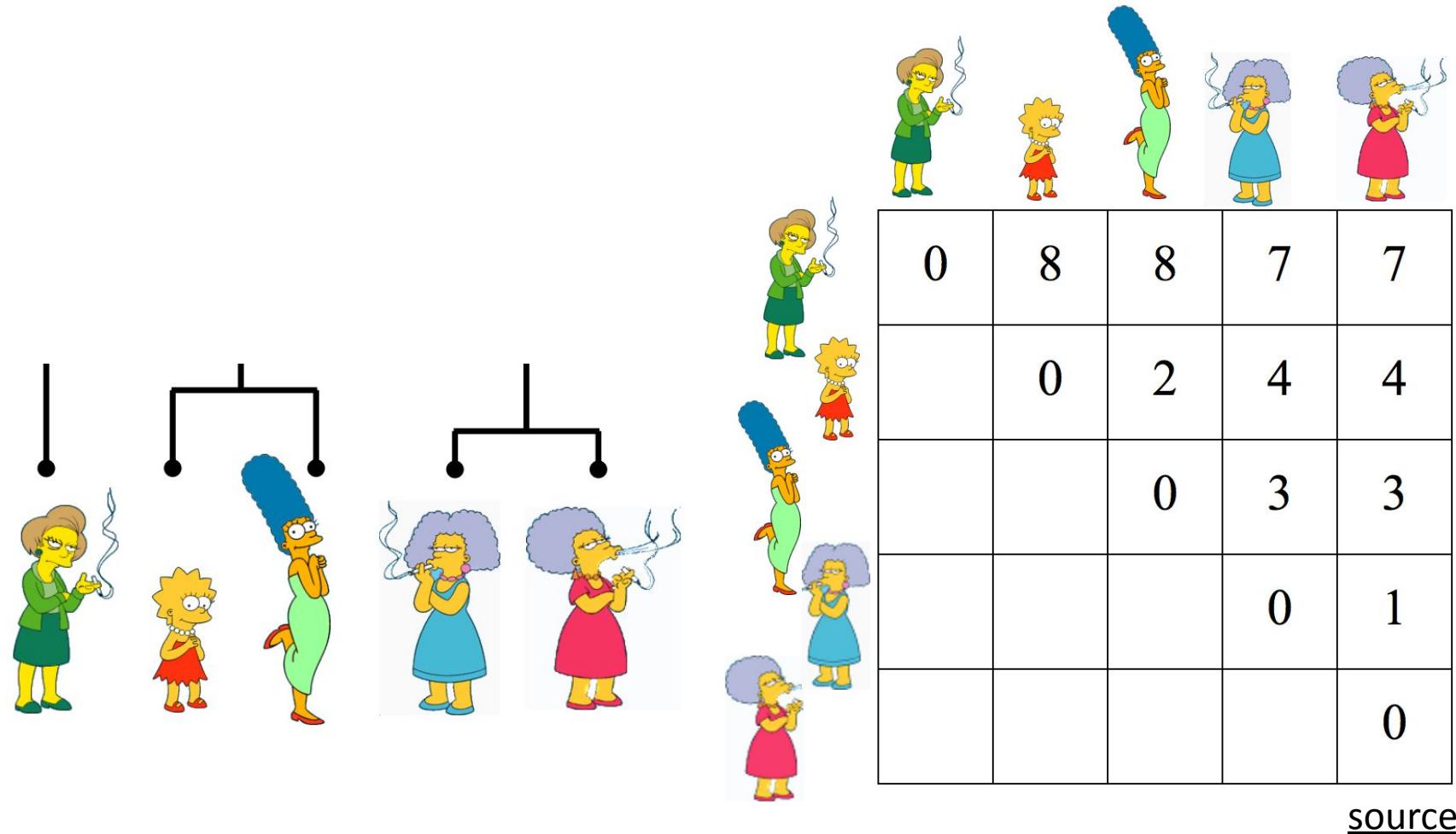
# What will we learn today?

- Introduction to segmentation and clustering
- Gestalt theory for perceptual grouping
- Graph-based oversegmentation
- **Agglomerative clustering**
- K-means
- Mean-shift

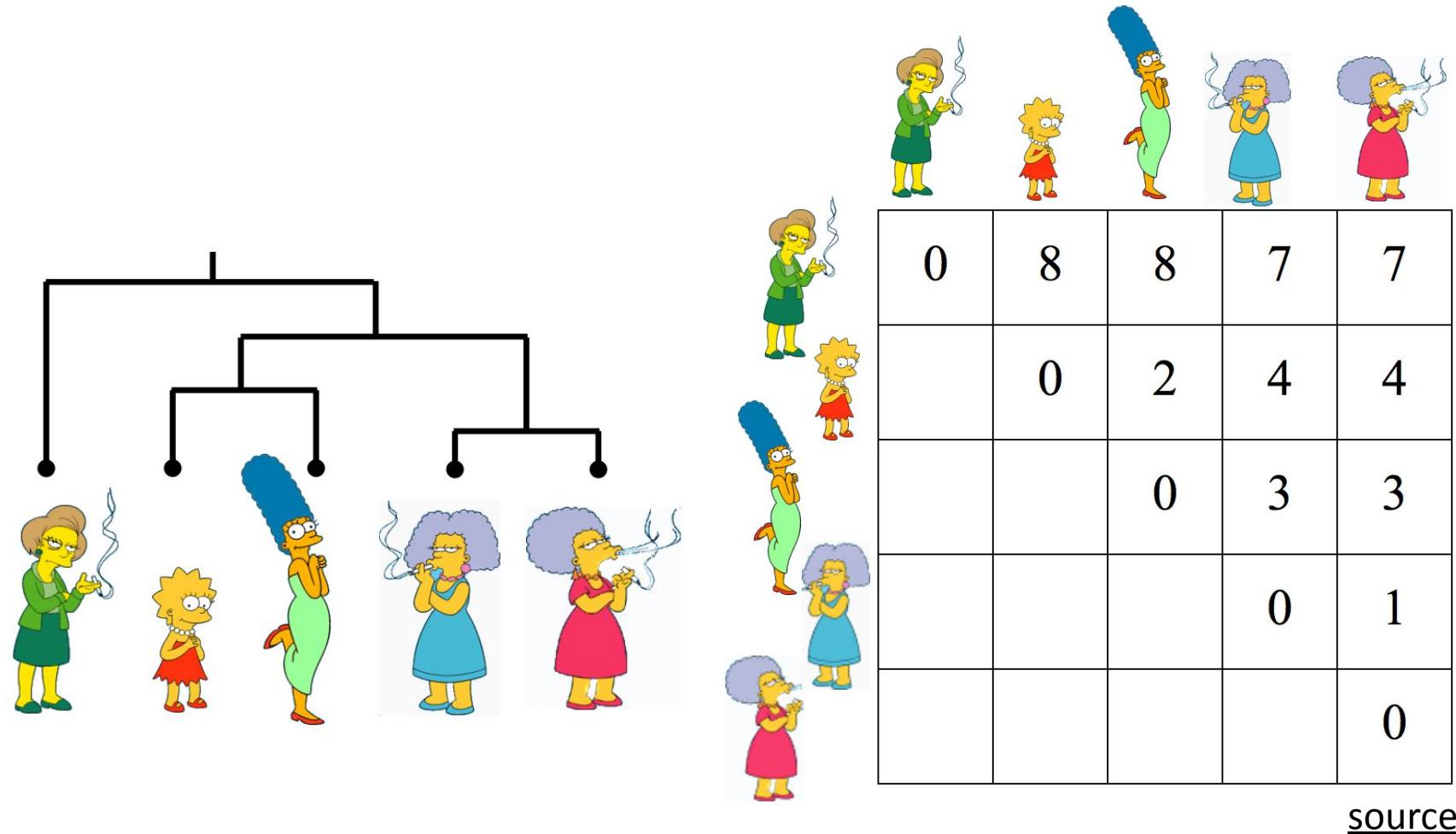
# Animated example



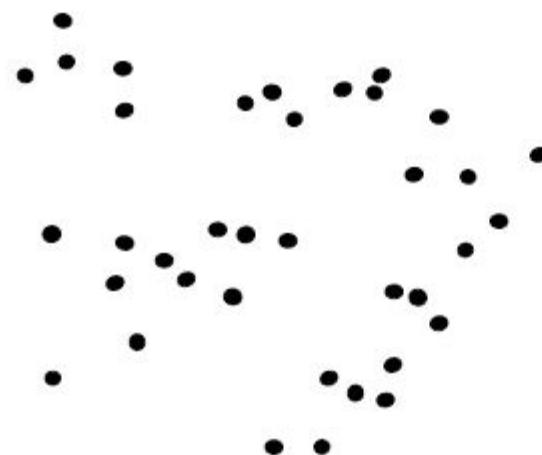
# Animated example



# Animated example



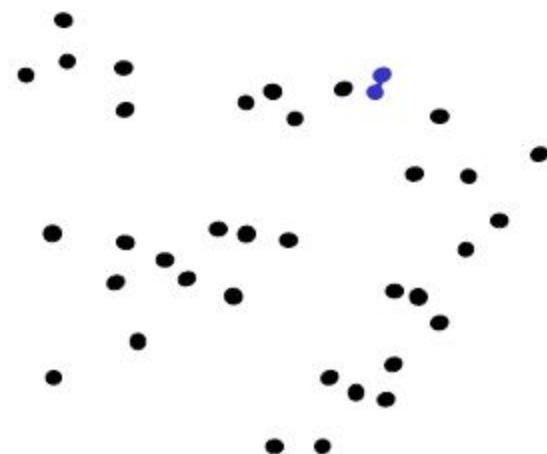
# Agglomerative clustering



1. Say "Every point is its own cluster"

Slide credit: Andrew Moore

# Agglomerative clustering

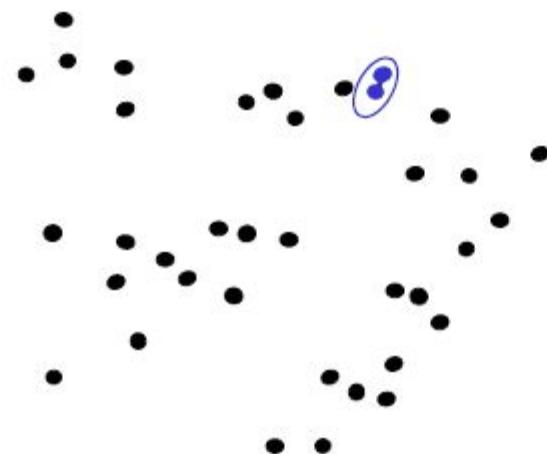


1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters



Slide credit: Andrew Moore

# Agglomerative clustering

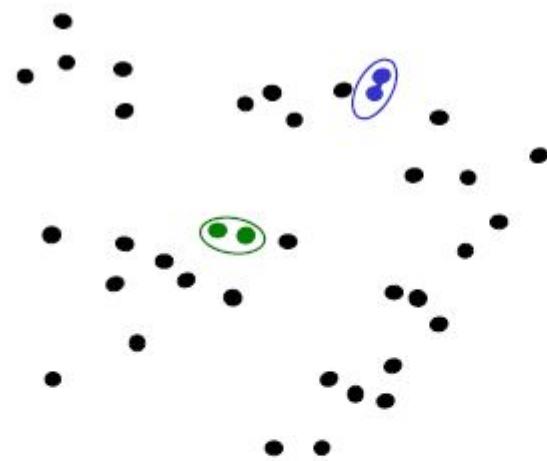


1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster



Slide credit: Andrew Moore

# Agglomerative clustering

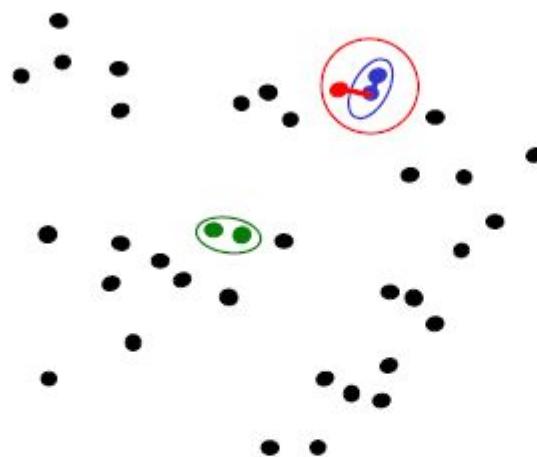


1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

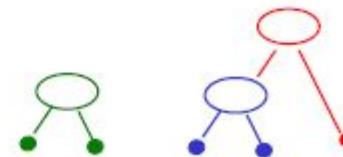


Slide credit: Andrew Moore

# 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

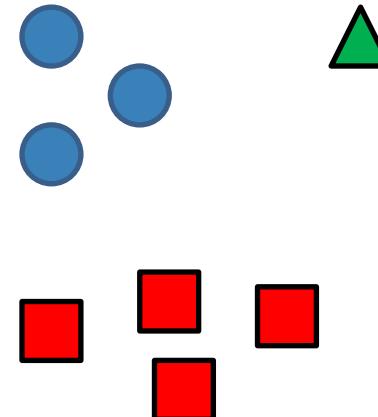


Slide credit: Andrew Moore

# Agglomerative clustering

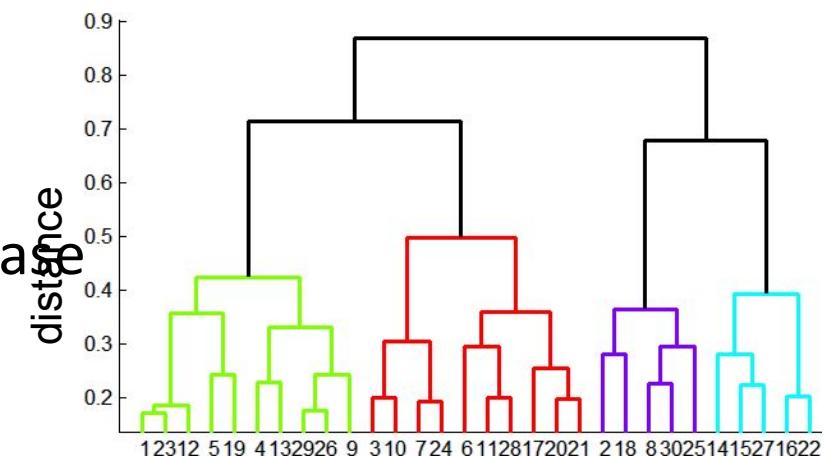
How to define cluster similarity?

- Average distance between all pixels between the two cluster?
- Maximum distance?
- Minimum distance?
- Distance between means?



How many clusters?

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



# Agglomerative Hierarchical Clustering - Algorithm

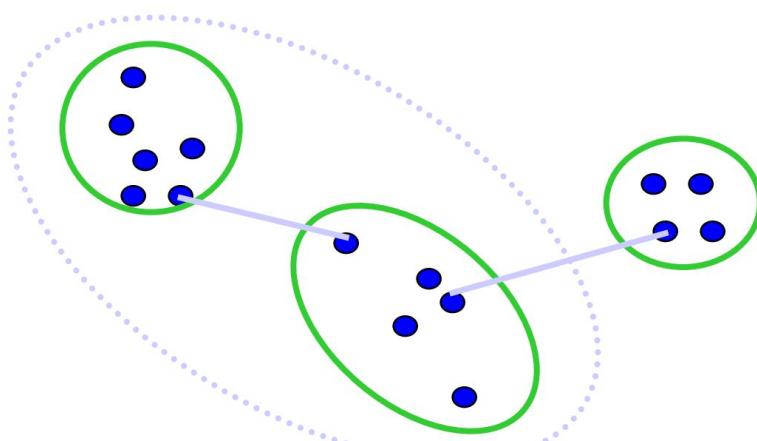
## Inputs:

- An input image
  - Feature representation for each pixel
  - Distance metric  $\text{dist}(\cdot, \cdot)$
- Initially, each pixel  $v_1, \dots, v_n$  is its own cluster  $C_1, \dots, C_n$
- While True:
  - Find two nearest clusters according to  $\text{dist}(C_i, C_j)$
  - Merge  $C = (C_i, C_j)$
  - If only 1 cluster is left:
    - break

# Different measures of nearest clusters

**Single Linkage:**  $dist(C_i, C_j) = \min_{v_i \in C_i, v_j \in C_j, (C_i, C_j) \in E} dist(v_i, v_j)$

Connects the clusters based on the distance of their closest pixels  
It produces “long” clusters.

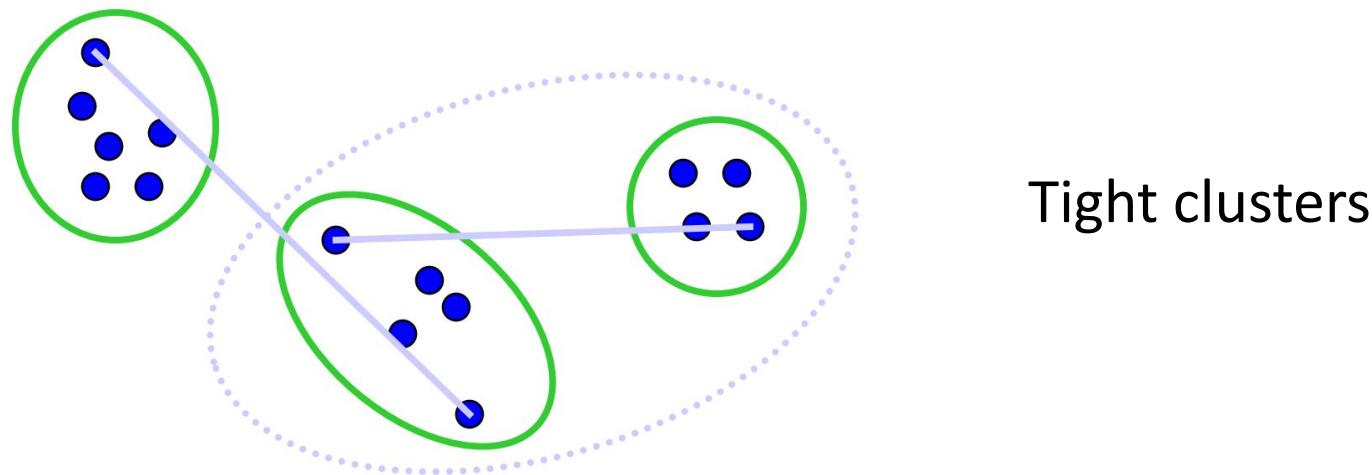


Long, skinny clusters

# Different measures of nearest clusters

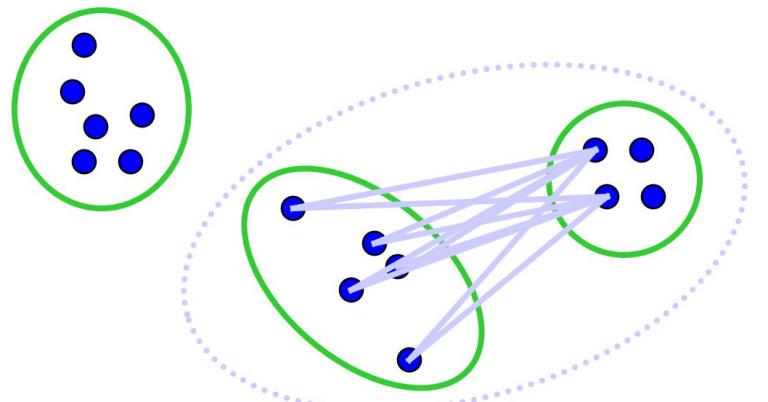
**Complete Link:**  $dist(C_i, C_j) = \max_{v_i \in C_i, v_j \in C_j, (C_i, C_j) \in E} dist(v_i, v_j)$

Produces compact clusters that are similar in diameter



# Different measures of nearest clusters

**Average Link:**  $dist(C_i, C_j) = \frac{\sum_{v_i \in C_i, v_j \in C_j, (C_i, C_j) \in E} dist(v_i, v_j)}{|C_i||C_j|}$



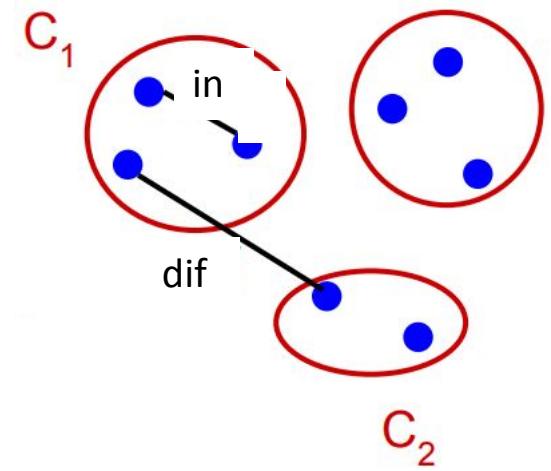
Robust against noise.

# Different measures of nearest clusters

**Inlier-outlier linkage:**  $Merge(C_1, C_2) = \begin{cases} True & \text{if } dif(C_1, C_2) < in(C_1, C_2) \\ False & \text{otherwise} \end{cases}$

Where

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



# Different measures of nearest clusters

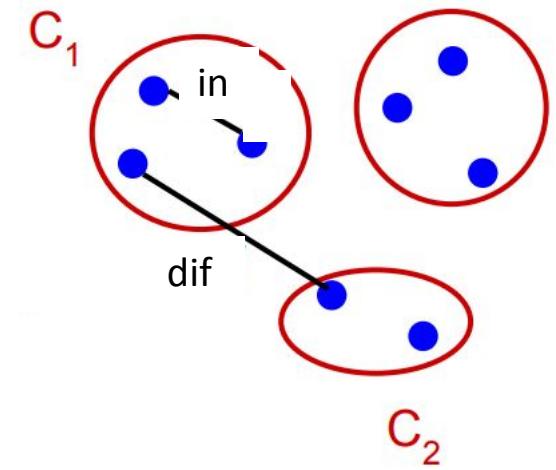
**Inlier-outlier linkage:**

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

$$\text{dif}(C_i, C_j) = \min_{v_i \in C_i, v_j \in C_j, (C_i, C_j) \in E} \text{dist}(v_i, v_j)$$

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  $C_2$



# Different measures of nearest clusters

**Inlier-outlier linkage:**

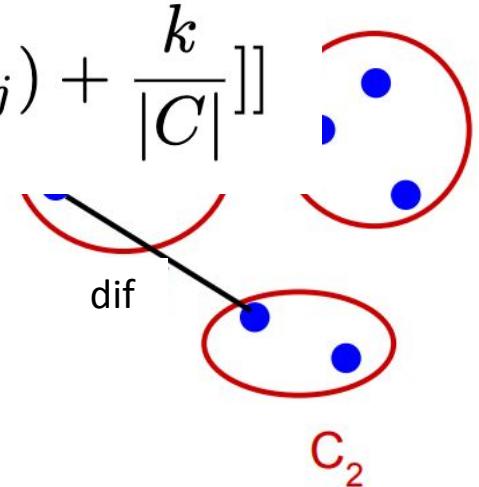
$$\text{Merge}(C_1, C_2) = \begin{cases} \text{True} & \text{if } \text{dif}(C_1, C_2) < \text{in}(C_1, C_2) \\ \text{False} & \text{otherwise} \end{cases}$$

$$\text{dif}(C_i, C_j) = \min_{v_i \in C_i, v_j \in C_j, (C_i, C_j) \in E} \text{dist}(v_i, v_j)$$

$$\text{in}(C_i, C_j) = \min_{C \in \{C_i, C_j\}} \left[ \max_{v_i, v_j \in C} [\text{dist}(v_i, v_j) + \frac{k}{|C|}] \right]$$

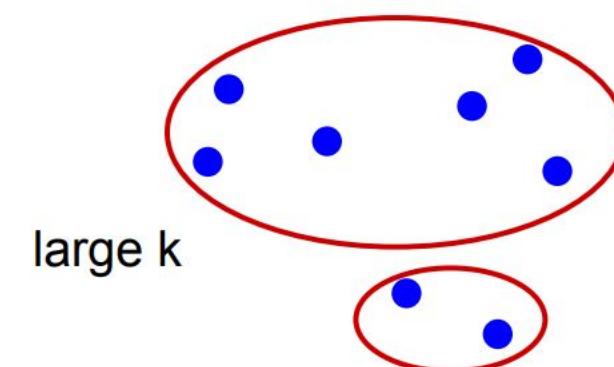
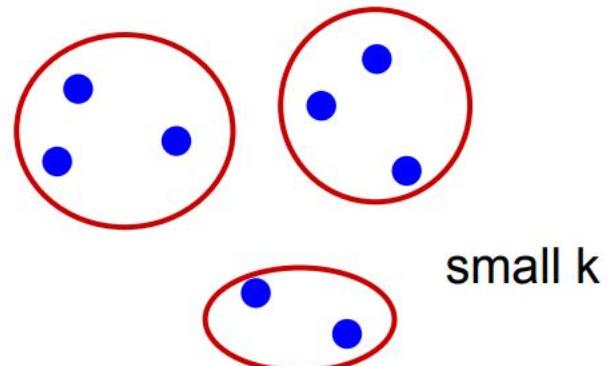
Where

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

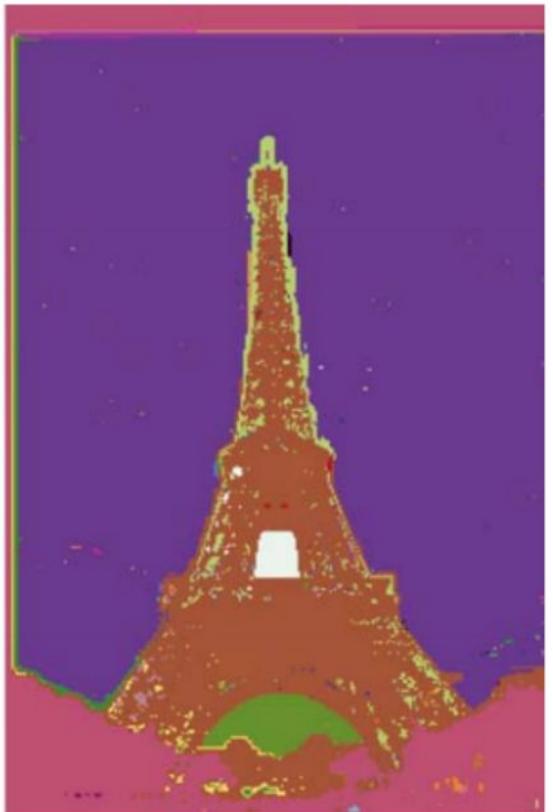
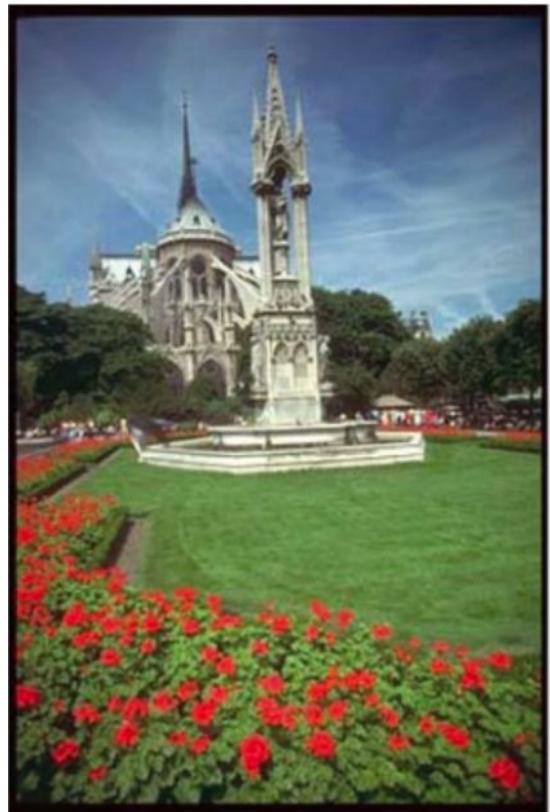


# inlier-outlier linkage for Segmentation

- $k/|C|$  sets the threshold by which the clusters need to be different from the internal pixels in a cluster.
- Effect of  $k$ :
  - If  $k$  is large, it causes a preference for larger objects.



# Results



# Conclusions: Agglomerative Clustering

## Pros:

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

## Cons:

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

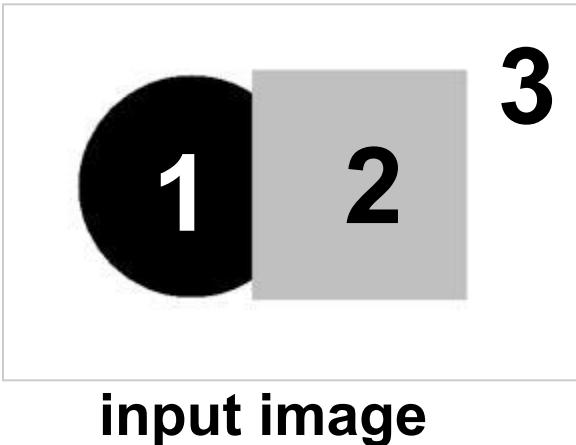
# Today's agenda

- Introduction to segmentation and clustering
- Gestalt theory for perceptual grouping
- Graph-based oversegmentation
- Agglomerative clustering
- **K-means clustering**
- Mean-shift clustering

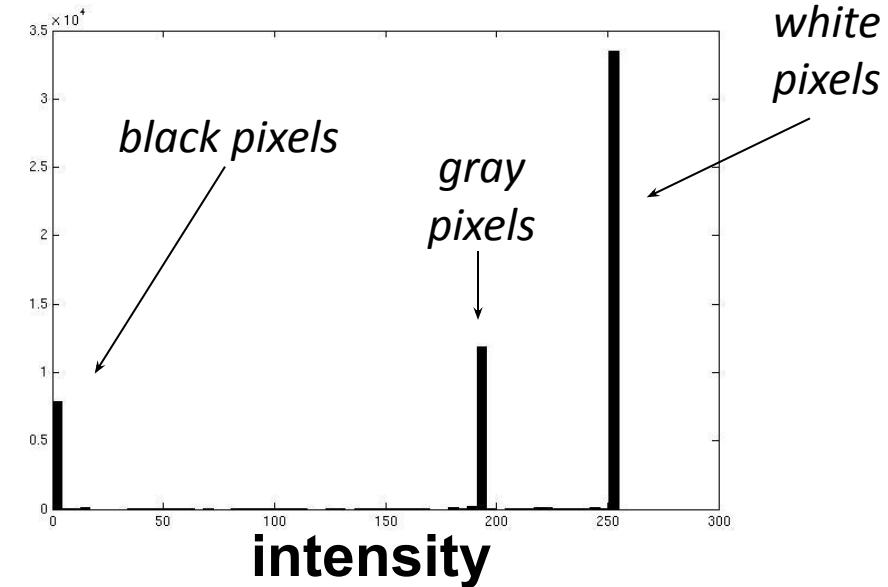
**Reading:** Szeliski Chapters: [5.2.2](#), 7.5.2

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

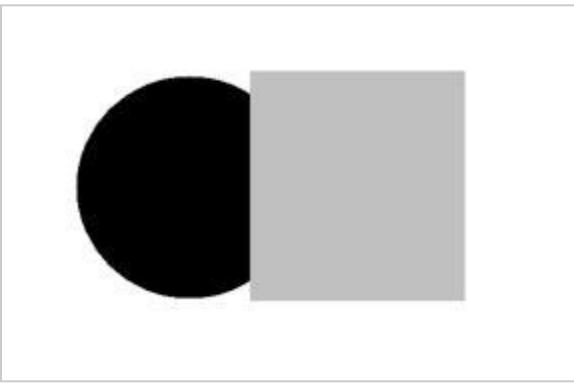
# Image Segmentation: Binary image Example



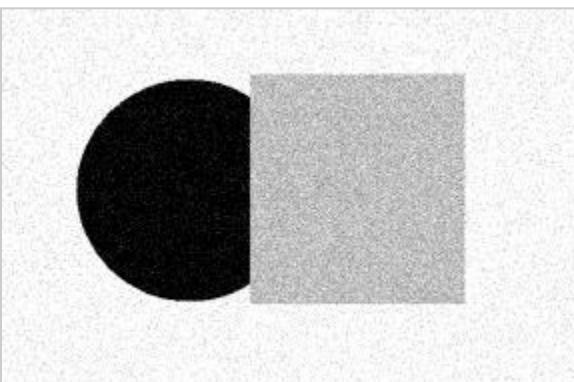
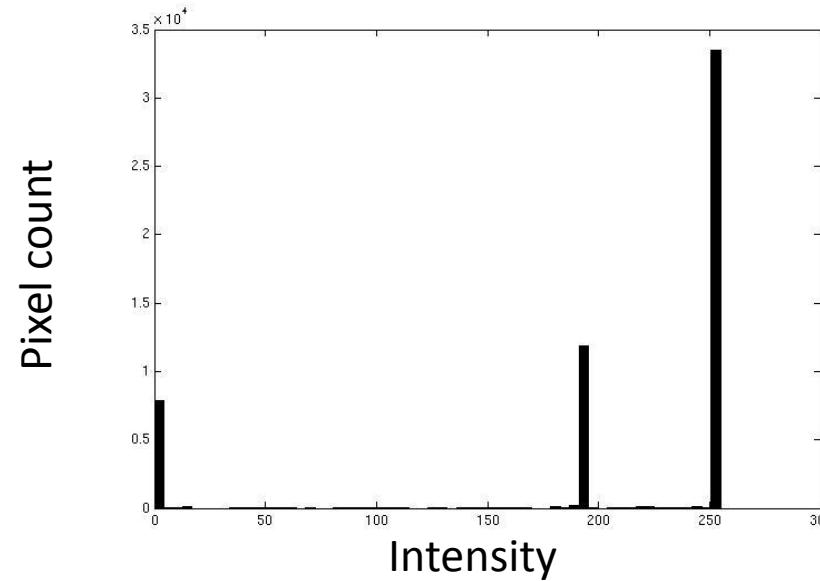
input image



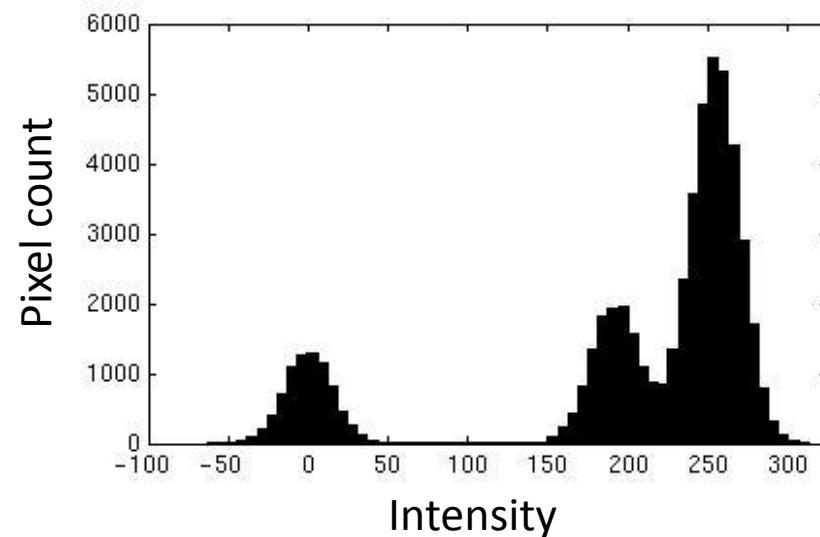
- These pixel values show that there are three things in the image.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., segment the image based on the intensity feature.
- What if the image isn't quite so simple?

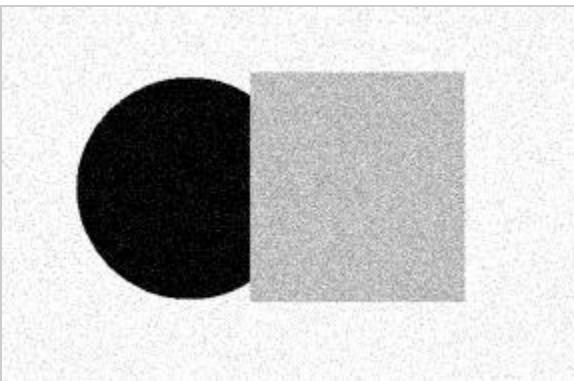


Input image

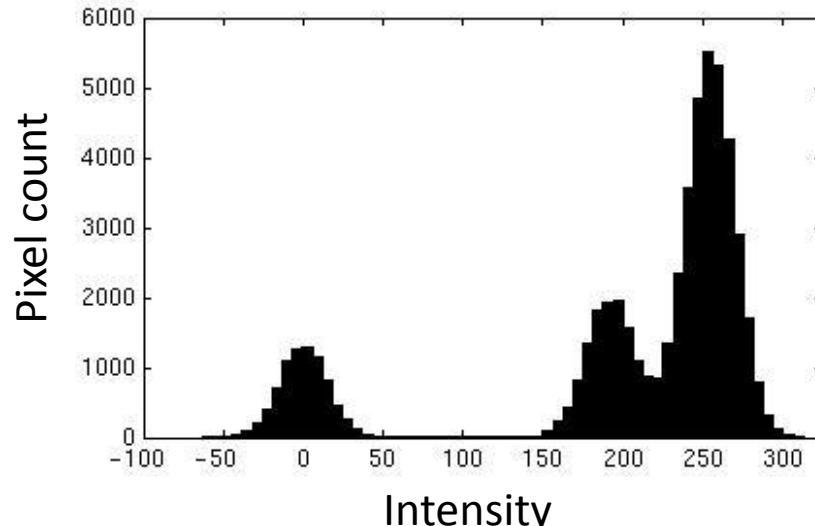


Input image

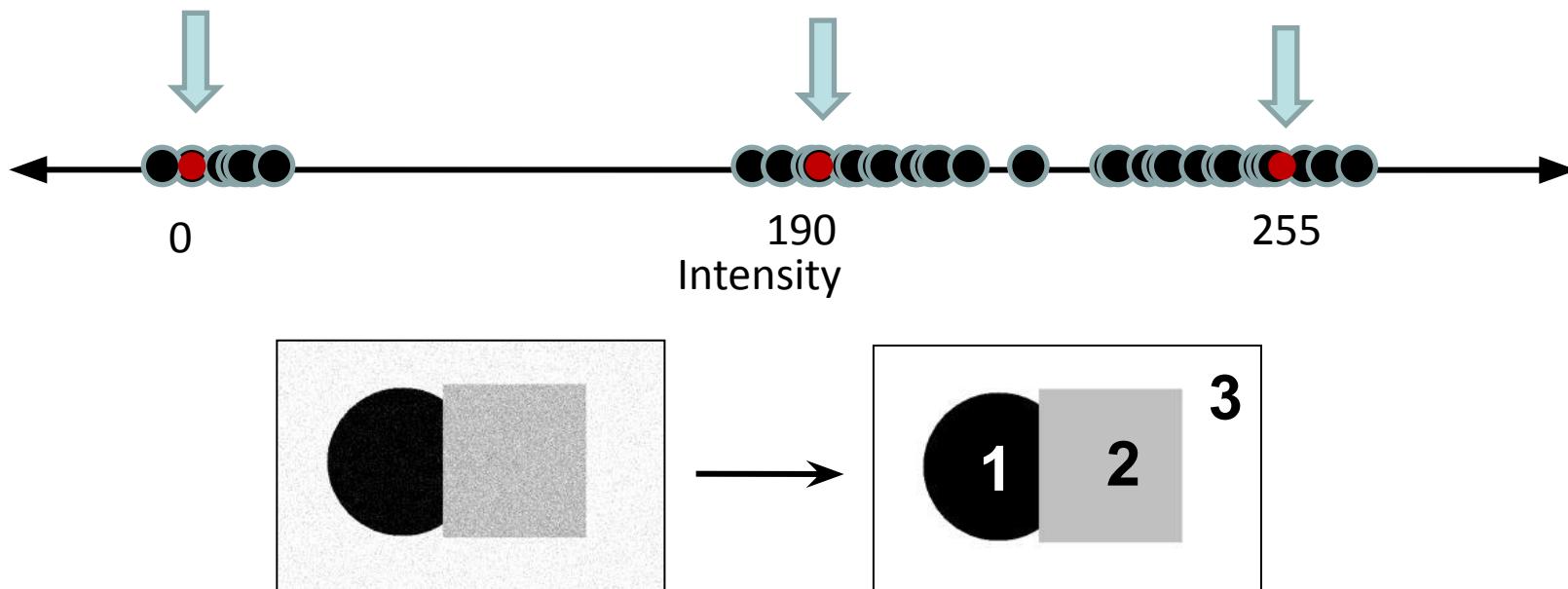




Input image



- How do we determine the three main intensities that define our groups?
- Each cluster has a cluster center
  - A mean cluster value.



- 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  $c_i$ :

$$SSD = \sum_C \sum_{v \in C} (v - c_i)^2$$

# Clustering

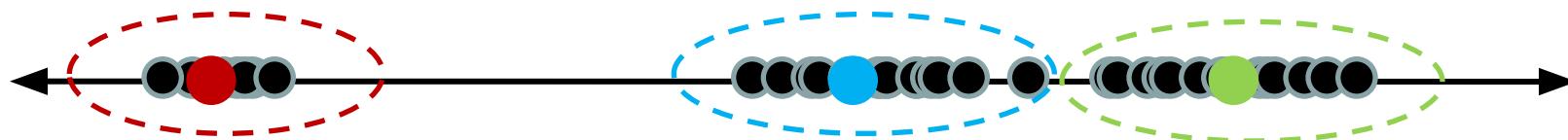
**Goal:** cluster to minimize distance of pixels to their cluster centers

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

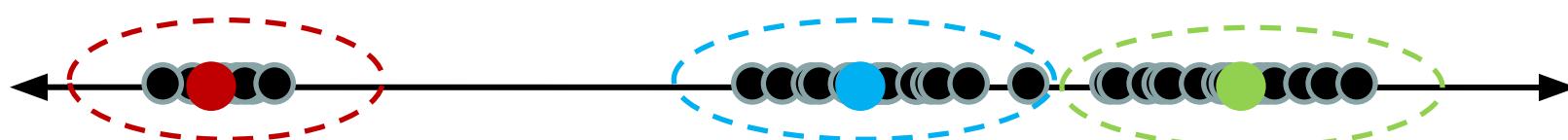
Cluster center      Data  
                                ↓  
                                ↓  
                                ↓  
Whether  $v_j$  is assigned to  $c_i$

# 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  $c_1, \dots, c_K$

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

$$\delta^t = \arg \min_{\delta} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij}^{t-1} (c_i^{t-1} - v_j)^2$$

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

$$\delta^t = \arg \min_{\delta} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij}^{t-1} (c_i^{t-1} - v_j)^2$$

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

$$c^t = \arg \min_c \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij}^t (c_i^{t-1} - v_j)^2$$

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

$$\delta^t = \arg \min_{\delta} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij}^{t-1} (c_i^{t-1} - v_j)^2$$

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

$$c^t = \arg \min_c \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij}^t (c_i^{t-1} - v_j)^2$$

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



# K-means clustering

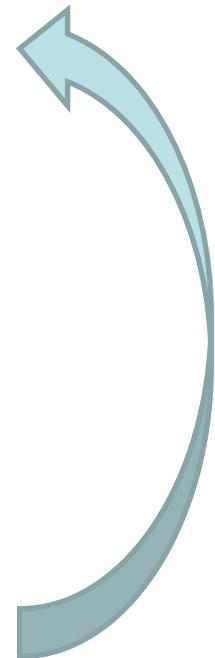
1. Initialize ( $t = 0$ ): cluster centers  $c_1, \dots, c_K$
2. Compute  $\delta^t$ : assign each point to the closest center
  - o  $\delta^t$  denotes the set of assignment for each  $v_j$  to cluster  $c_i$  at iteration  $t$

$$\delta^t = \arg \min_{\delta} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij}^{t-1} (c_i^{t-1} - v_j)^2$$

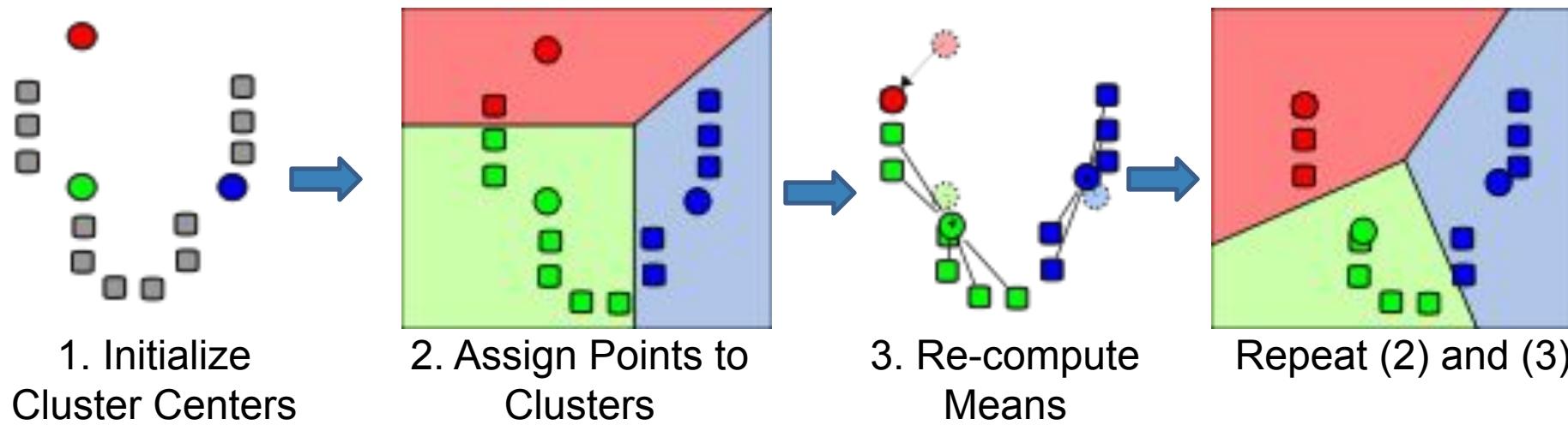
3. Computer  $c^t$ : update cluster centers as the mean of the points

$$c^t = \arg \min_c \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij}^t (c_i^{t-1} - v_j)^2$$

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



# K-means clustering



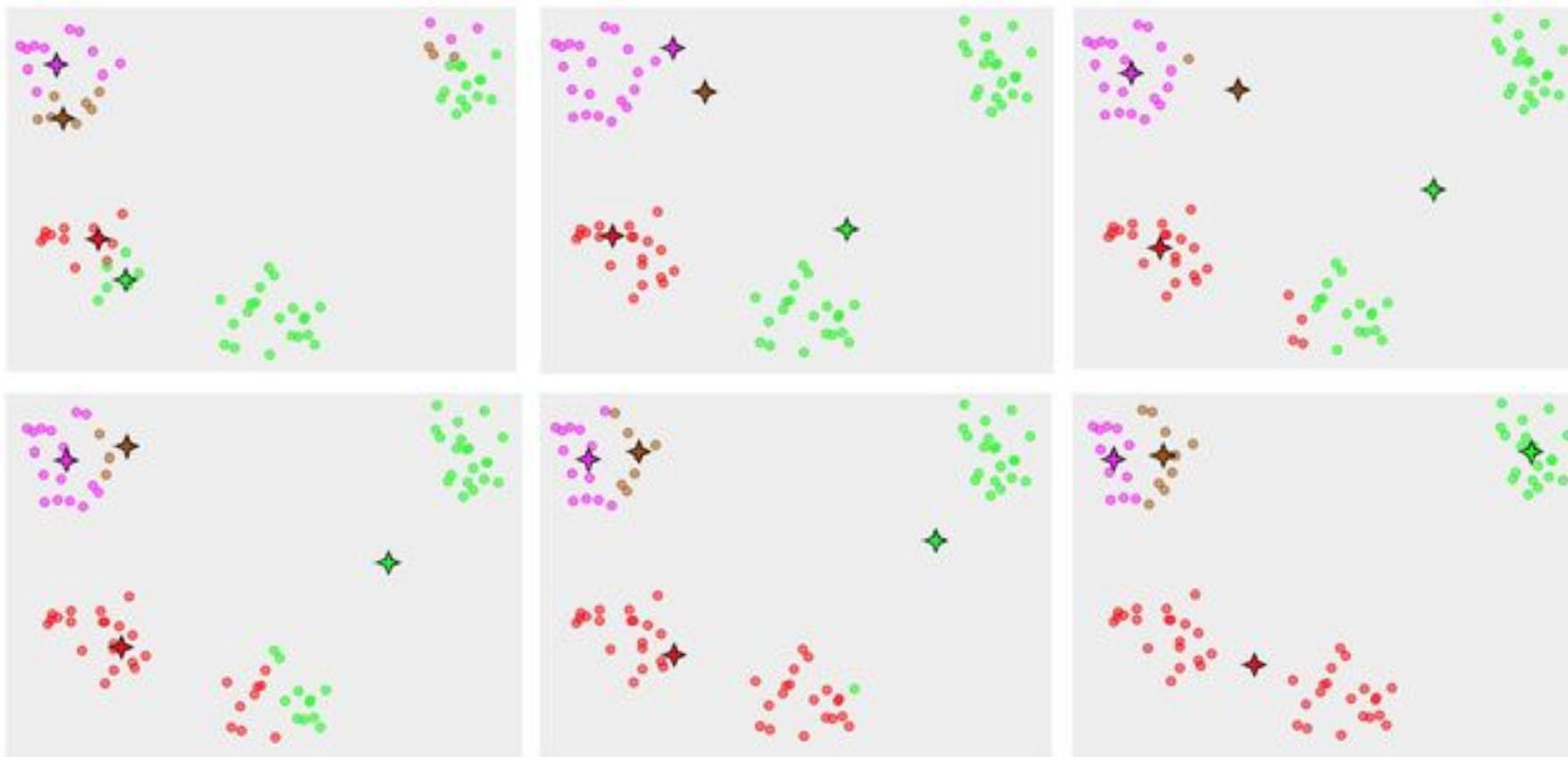
# K-means clustering

Initial cluster centers are randomly initialized

- Can lead to bad initializations
- Can cause bad clusters

# K-means Converges to a local minimum solution

Initialize multiple runs!



# K-Means++

Tries to prevent arbitrarily bad local minima?

1. Randomly choose first center.
2. Pick new center with prob. proportional to  $(c_i - v_j)^2$ 
  - a. Basically we want to find as good of an initialization as possible
3. Repeat until  $K$  centers.

# K-means clustering

Initial cluster centers are randomly initialized

- Can lead to bad initializations
- Can cause bad clusters

Different distance measures can change K-Means clusters

- Euclidean distance or cosine distance.

Different feature space can lead to different cluster

# Segmentation as Clustering



Original image



2 clusters



3 clusters

# Feature Space: pixel value

- Feature space: what measurements do we include in  $x_i$ ?
- Depending on what we choose as the *feature space*, we can group pixels in different ways.

- Grouping pixels based on **intensity** similarity

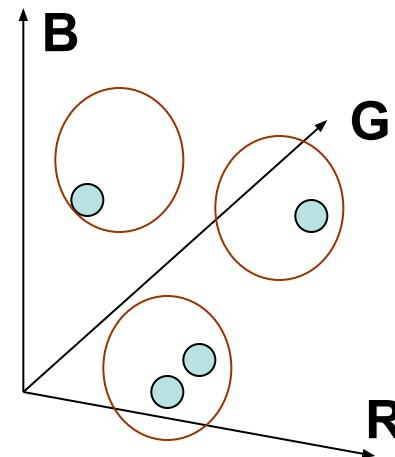


- Feature space: intensity value (1D)

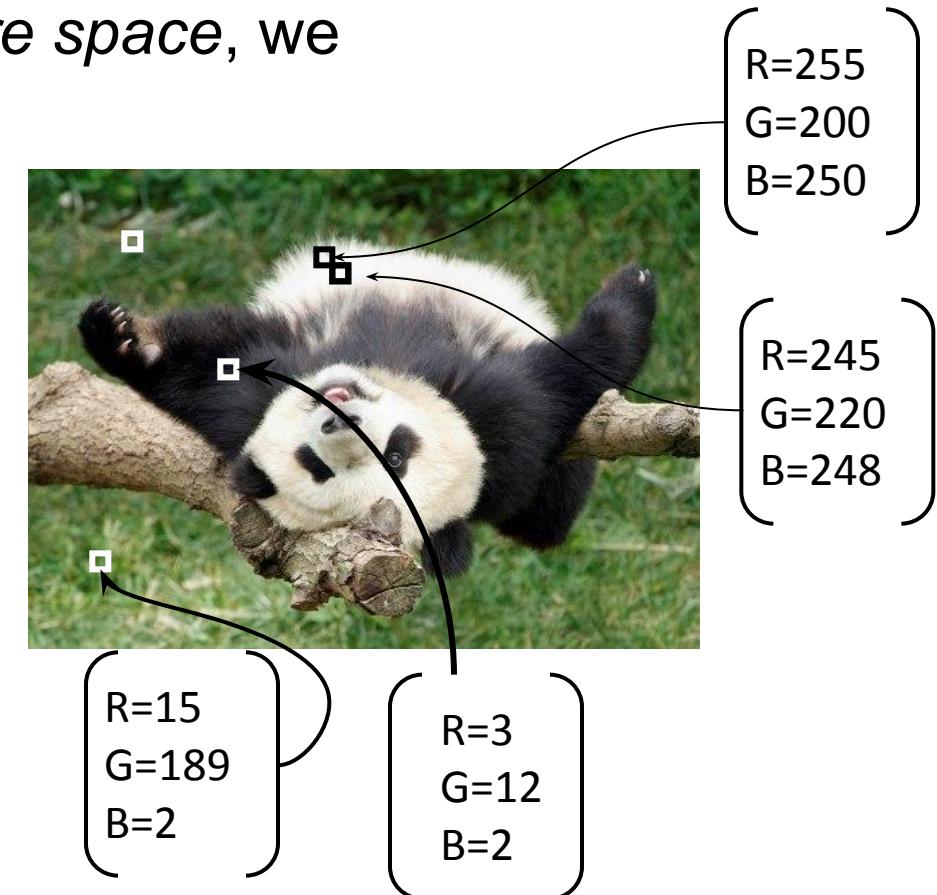
# Feature Space: RGB

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

- Grouping pixels based on **color** similarity

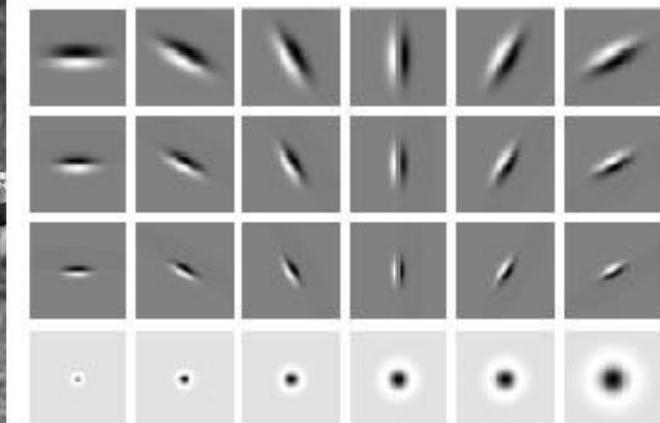
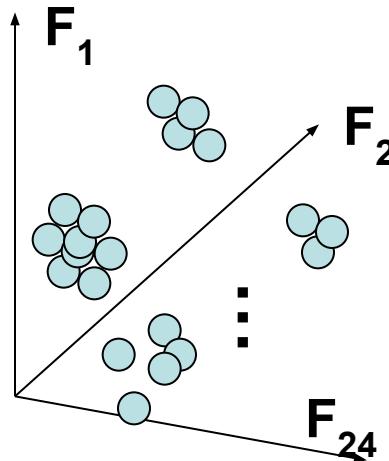


- Feature space: color value (3-dim)



# Feature Space: edges and blobs

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

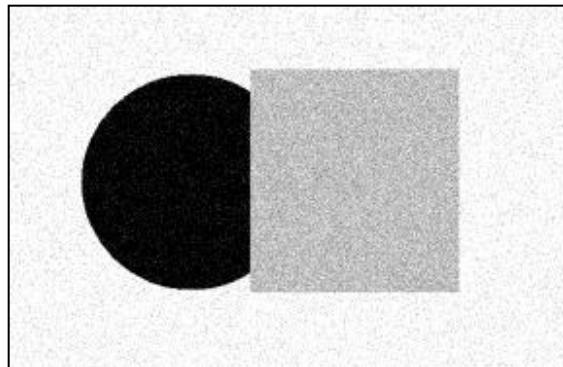


24 edge & blob filters

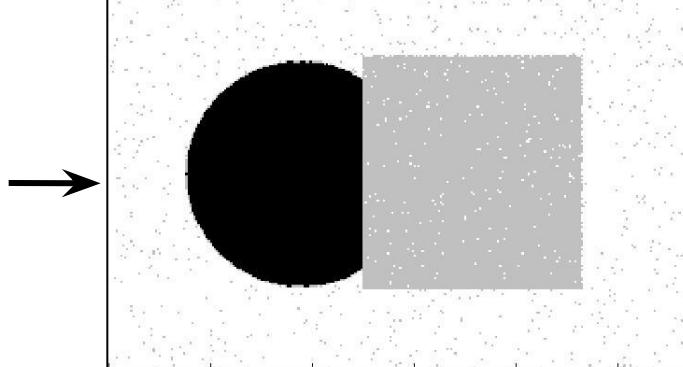
- Feature space: filter bank responses (e.g., 24D)

# Smoothing Out Cluster Assignments

- Assigning a cluster label per pixel may yield outliers:

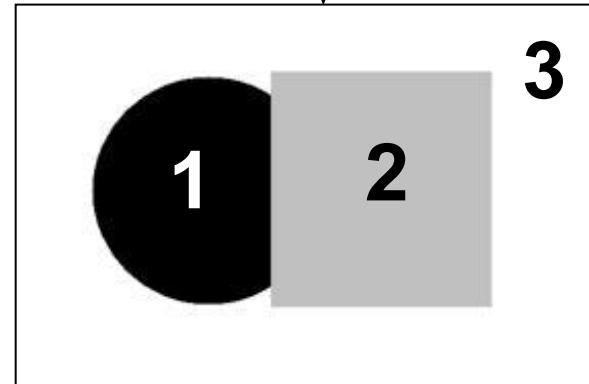


Original



Labeled by cluster center's intensity

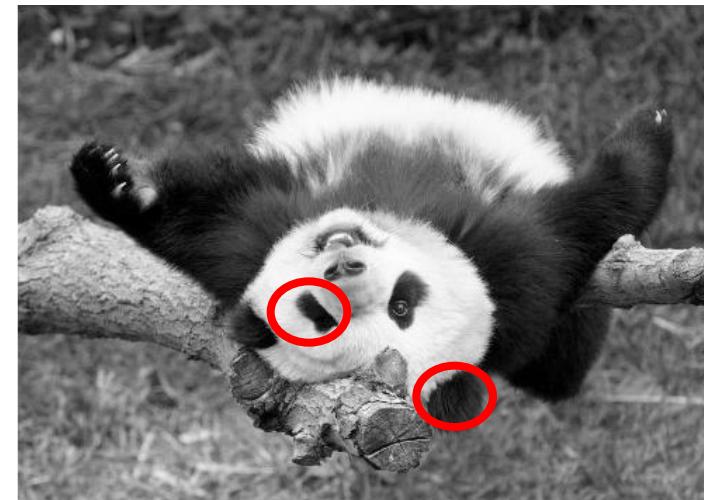
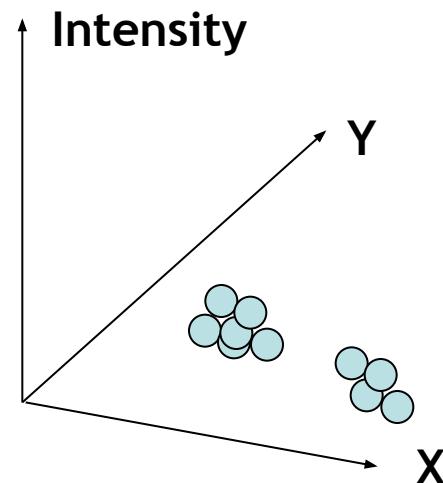
↓ ?



- How can we ensure they are spatially smooth?

# Feature Space: RGB + XY location

- 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

- Clusters don't have to be spatially coherent

Image



grayscale clusters



Color-based clusters



# K-Means Clustering Results

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



# How to evaluate clusters?

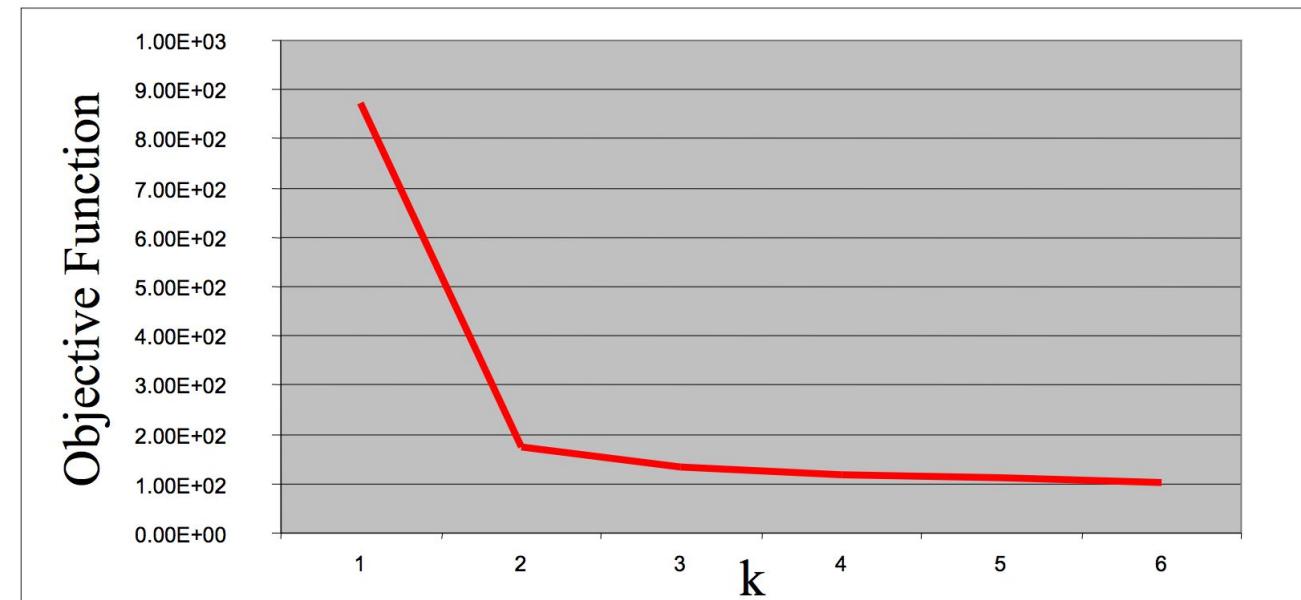
- **Generative**
  - How well are points reconstructed from the clusters?
- **Discriminative**
  - How well do the clusters correspond to labels?
    - Can we correctly classify which pixels belong to the panda?
  - Note: unsupervised clustering does not aim to be discriminative as we don't have the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

# How to choose the number of clusters?

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

Plot of SSD versus values of k

abrupt change at  $k=2$  is  
suggestive of two clusters in the  
data



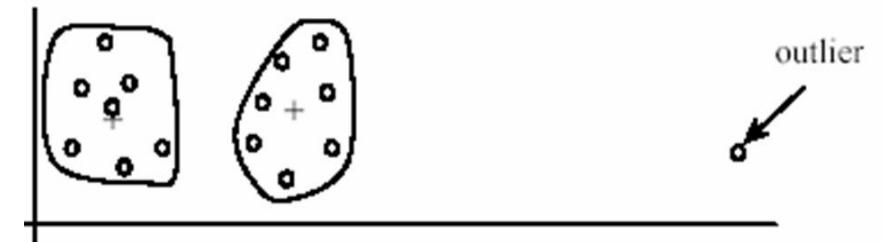
# K-Means pros and cons

- **Pros**

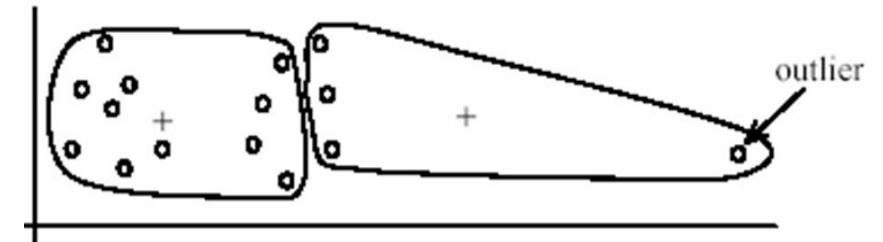
- Good representation of data
- Simple and fast, Easy to implement

- **Cons**

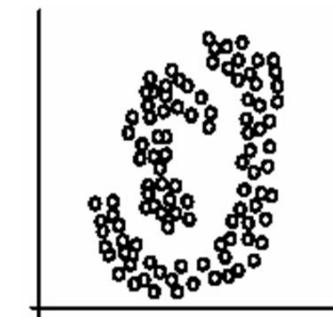
- Need to choose K
- Sensitive to outliers
- Prone to local minima
- All clusters have the same parameters (e.g., distance measure is non-adaptive)
- **Can still be slow: each iteration is  $O(KNd)$  for N d-dimensional pixels**



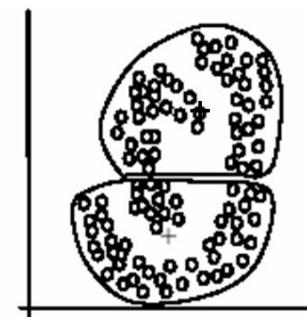
(B): Ideal clusters



outlier



(A): Two natural clusters



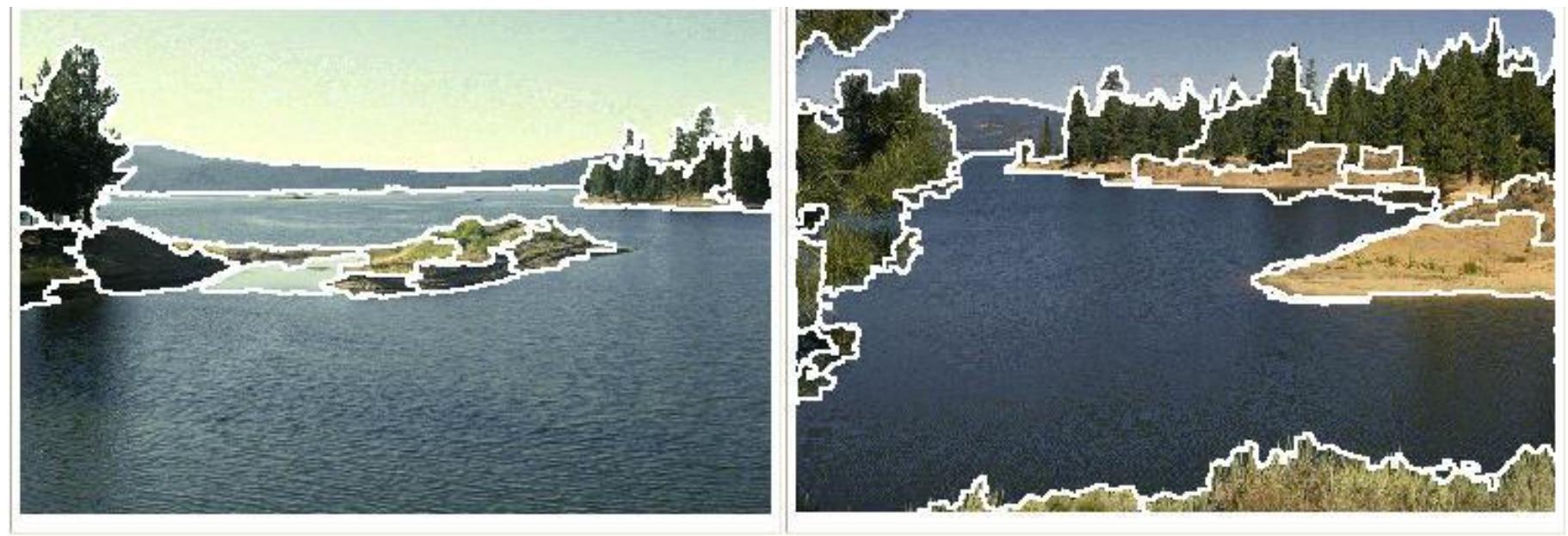
(B):  $k$ -means clusters

# What will we learn today?

- Introduction to segmentation and clustering
- Gestalt theory for perceptual grouping
- Graph-based oversegmentation
- Agglomerative clustering
- K-means clustering
- Mean-shift clustering

# Mean-Shift Segmentation

- An advanced and versatile technique for clustering-based segmentation



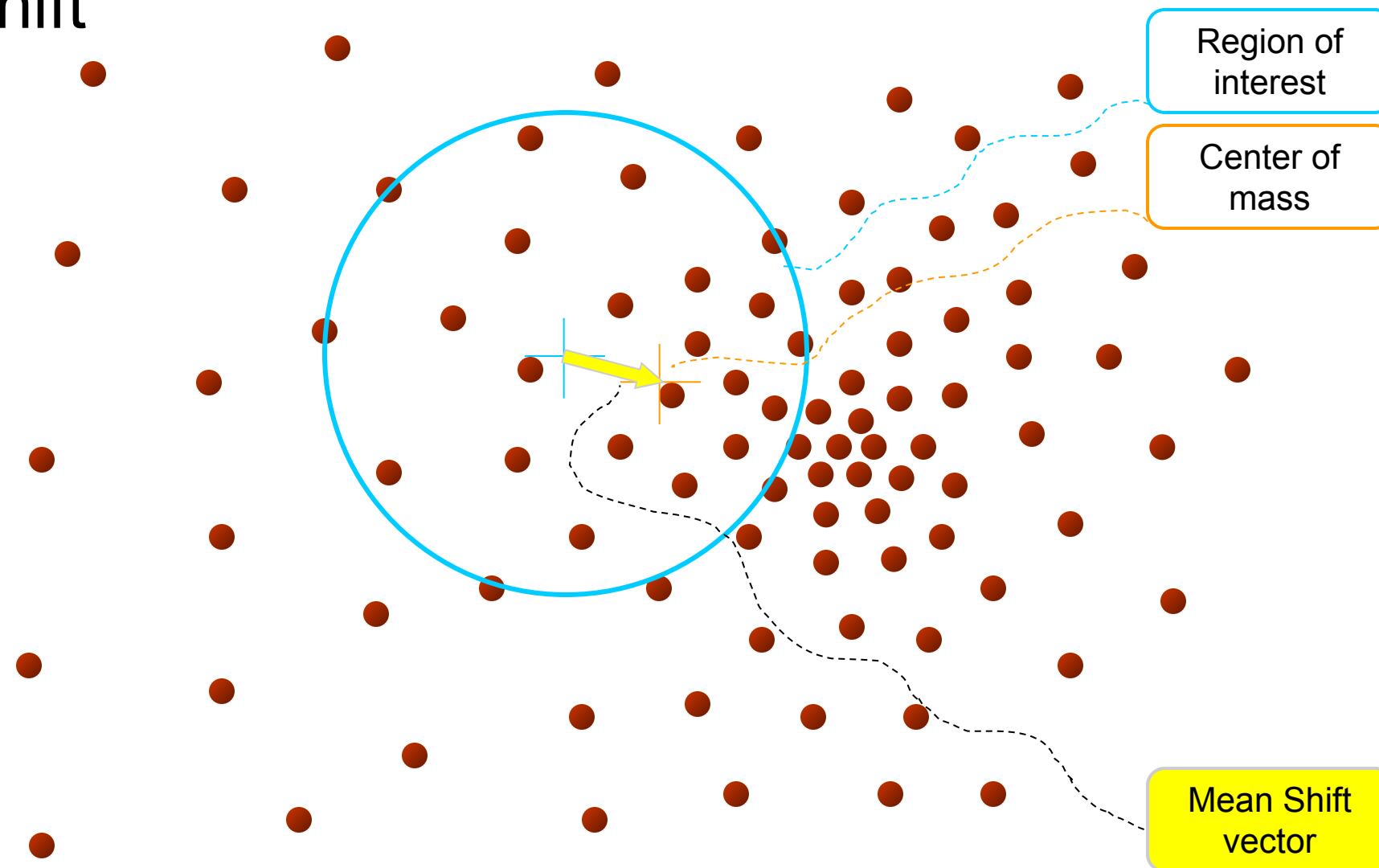
<http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html>

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

# Mean-Shift Algorithm

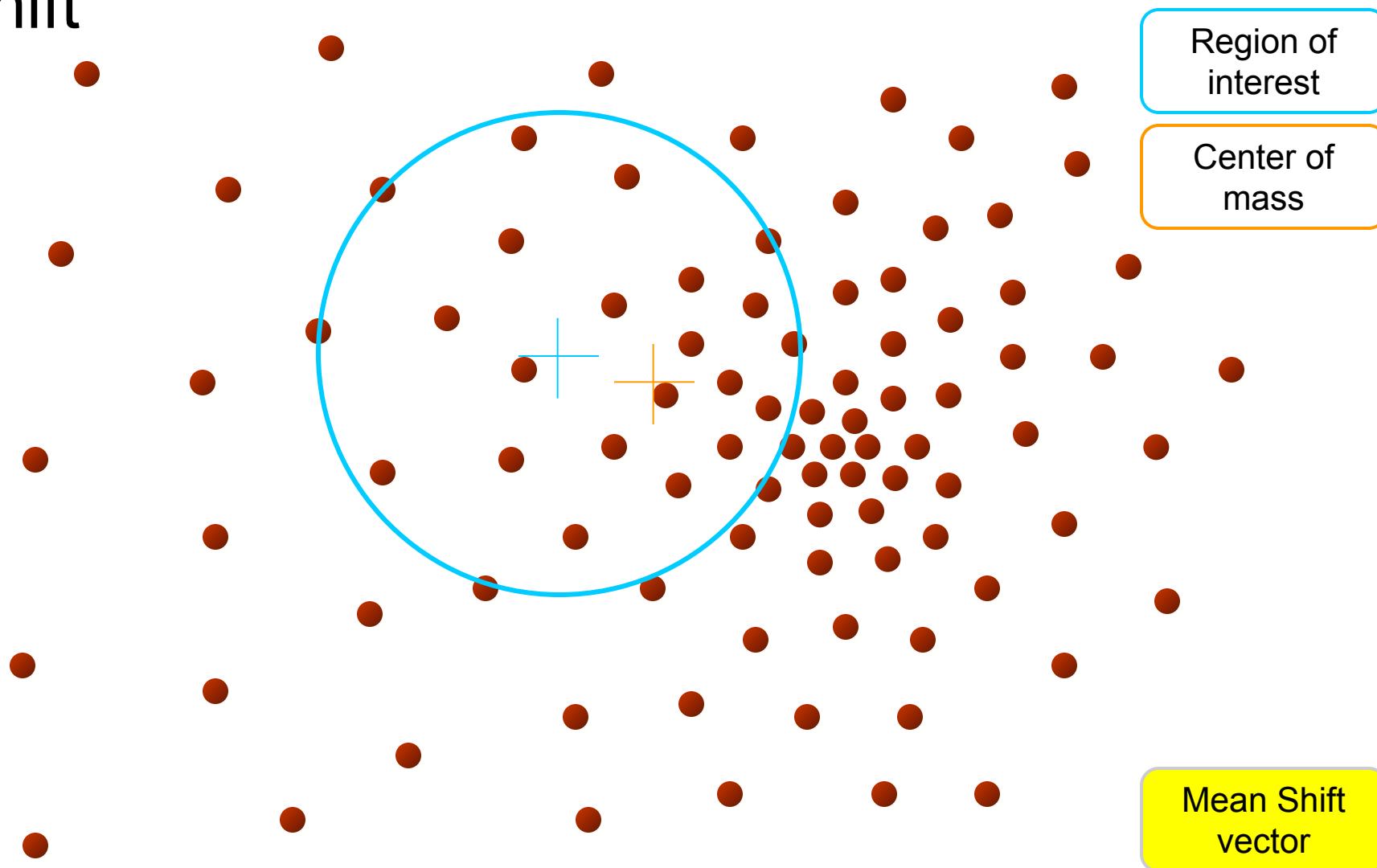
1. Represent each pixel  $i$  using some feature vector  $v_i$
2. Generate a window  $\mathbf{W}$  as a random pixel feature  $v_w$
3. Identify all the pixels within a radius  $r$  of  $v_w$
4. Calculate the mean (“center of gravity”) amongst the neighbors of  $\mathbf{W}$
5. Translate the window  $\mathbf{W}$  to the mean feature location
6. Repeat Step 2 until convergence

# Mean-Shift



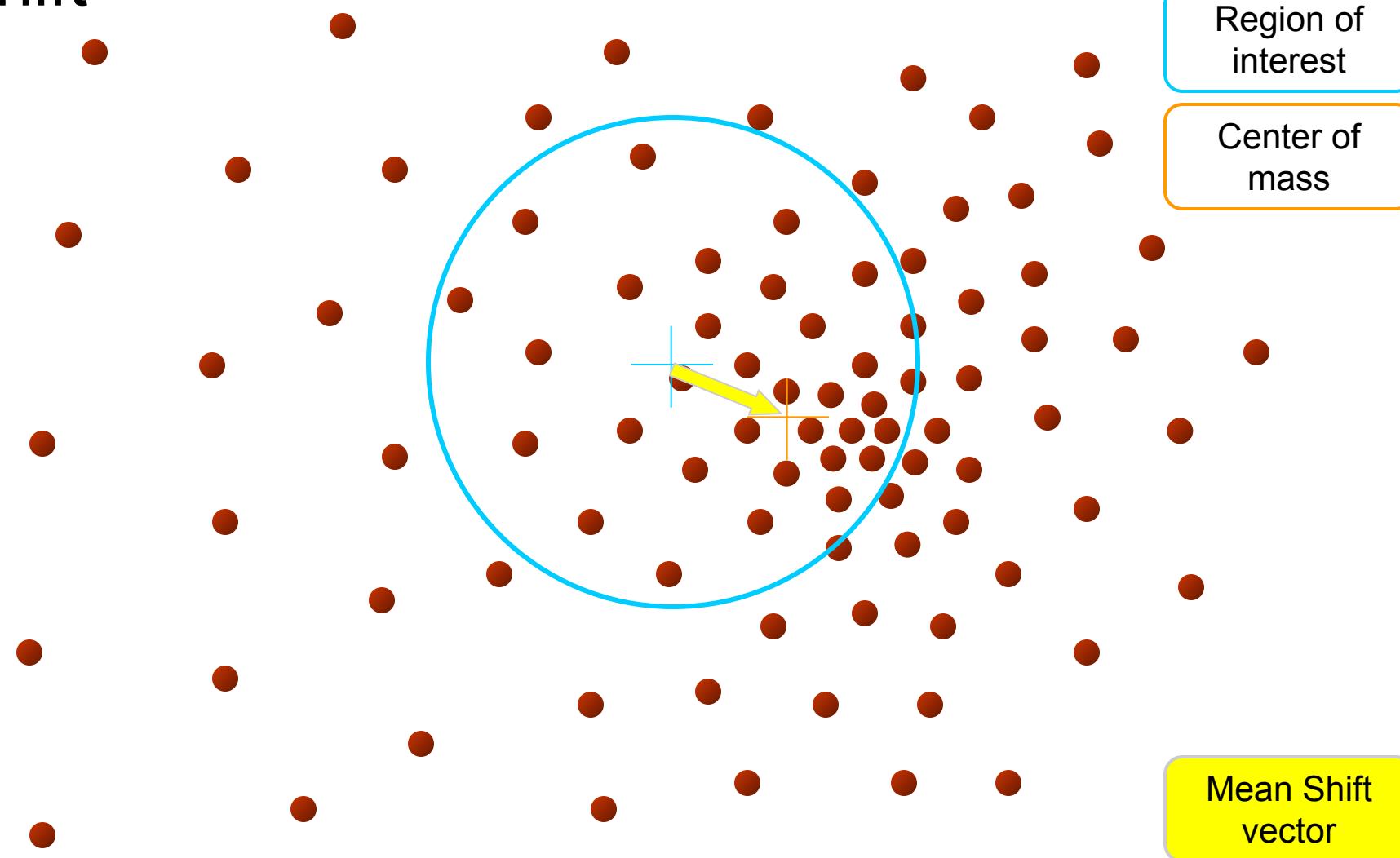
Slide by Y. Ukrainitz & B. Sarel

# Mean-Shift



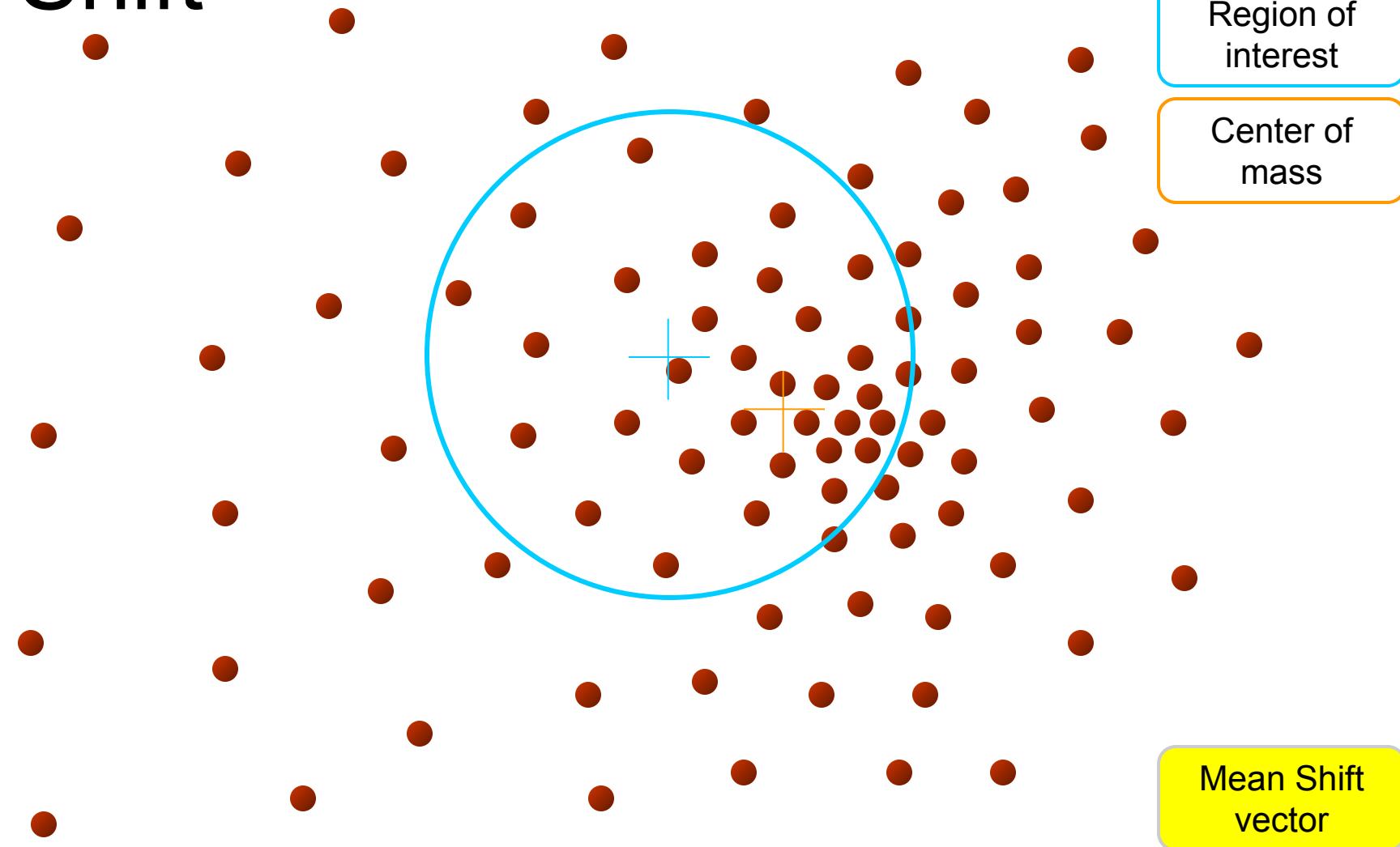
Slide by Y. Ukrainitz & B. Sarel

# Mean-Shift



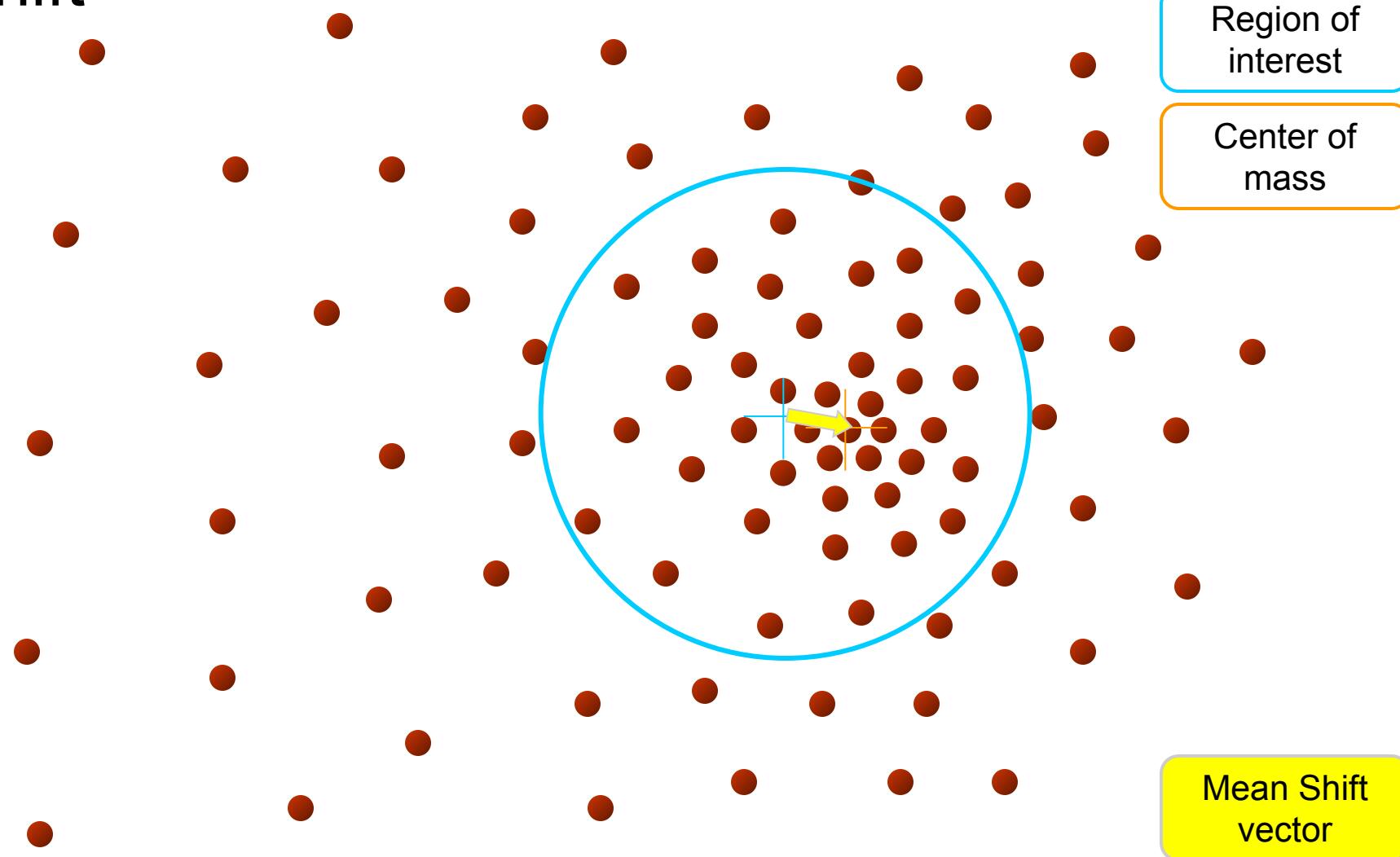
Slide by Y. Ukrainitz & B. Sarel

# Mean-Shift



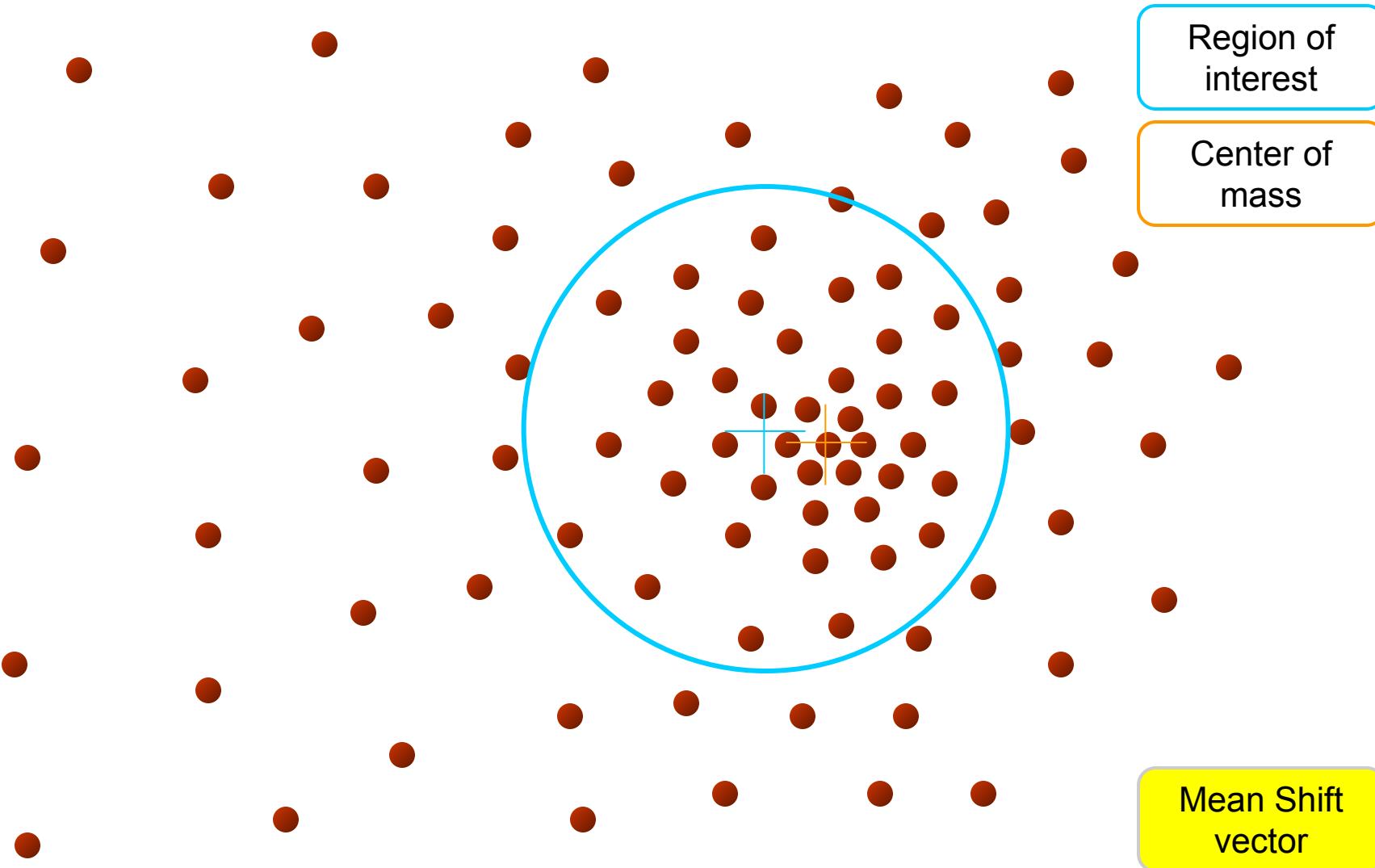
Slide by Y. Ukrainitz & B. Sarel

# Mean-Shift



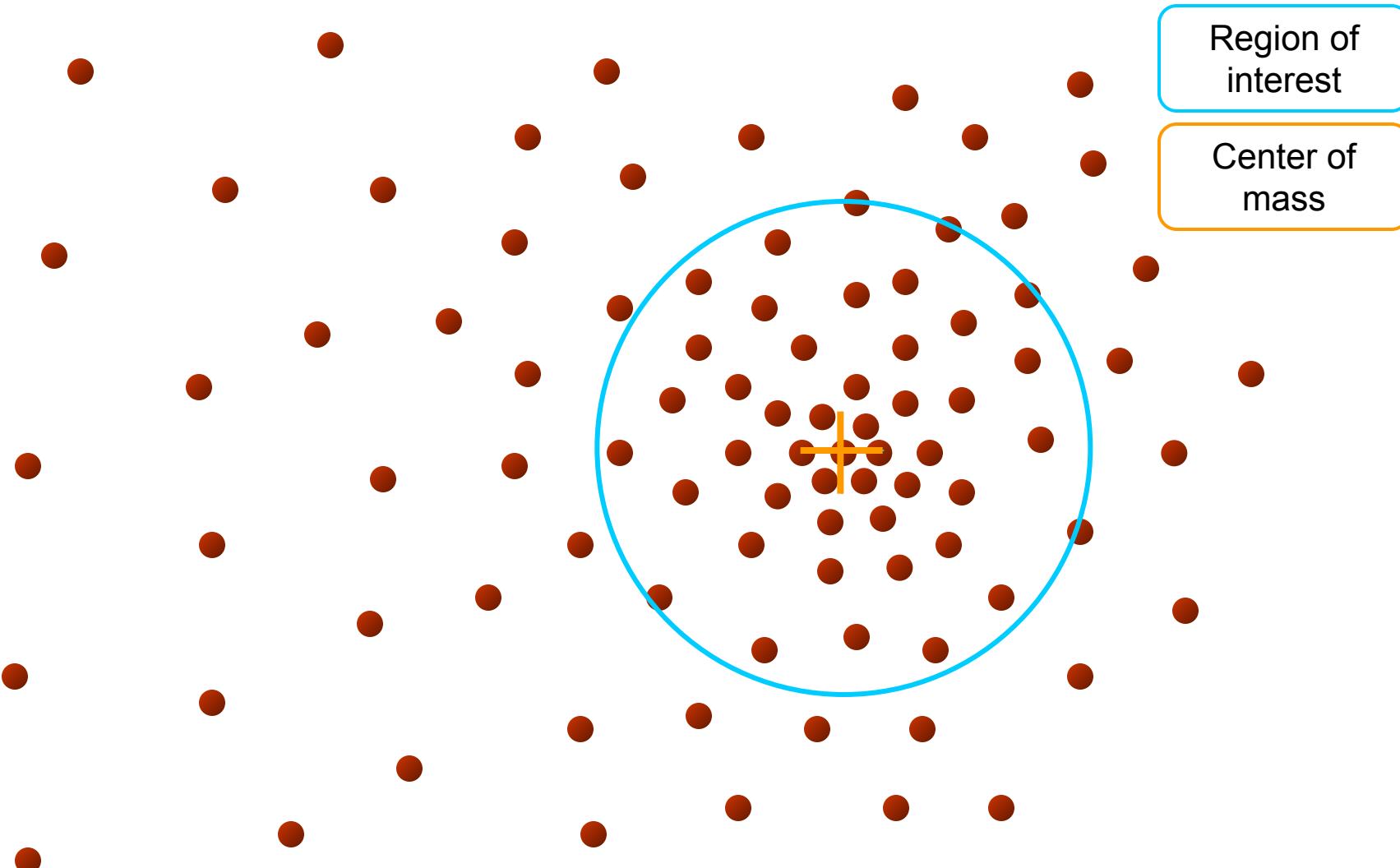
Slide by Y. Ukrainitz & B. Sarel

# Mean-Shift



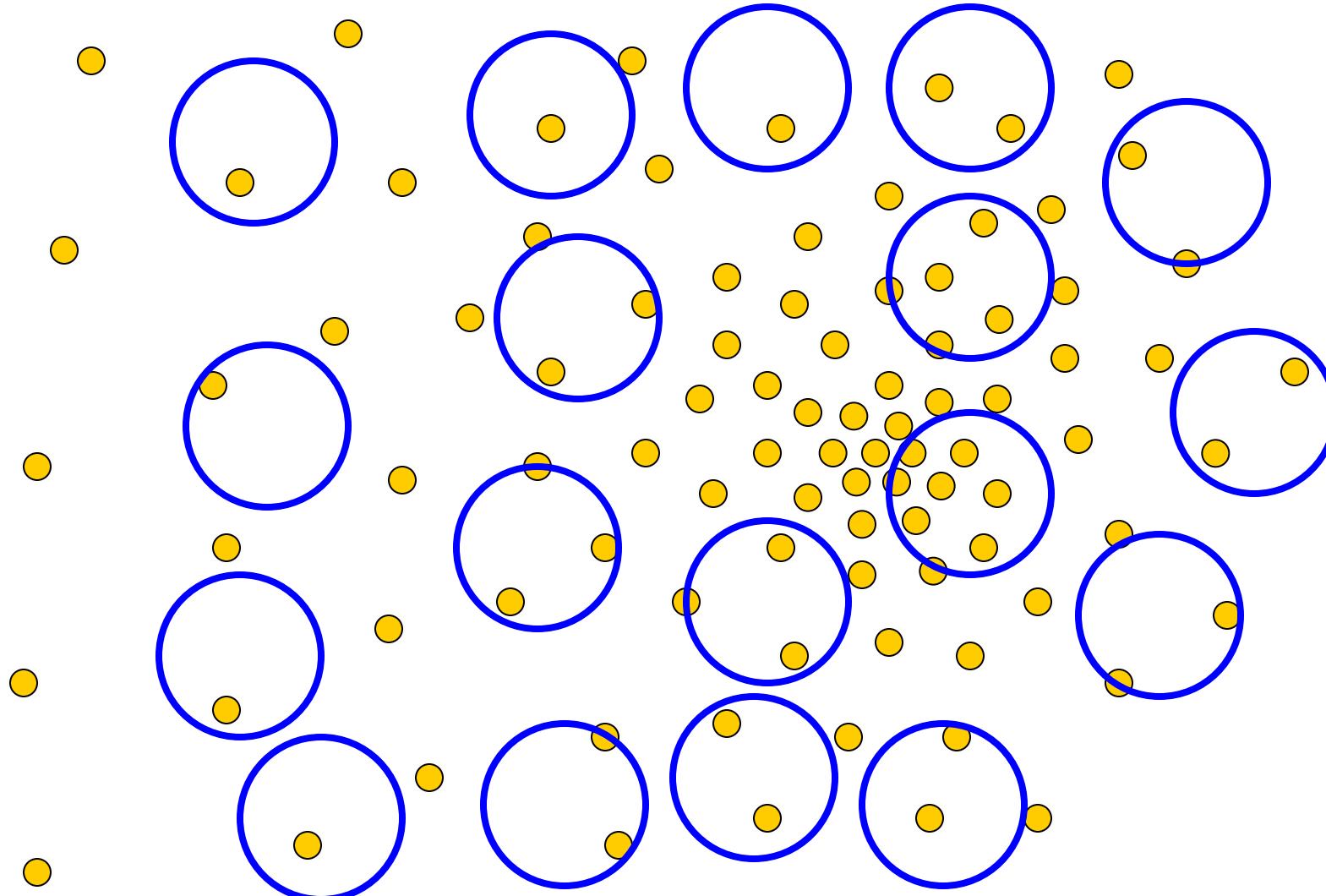
Slide by Y. Ukrainitz & B. Sarel

# Mean-Shift



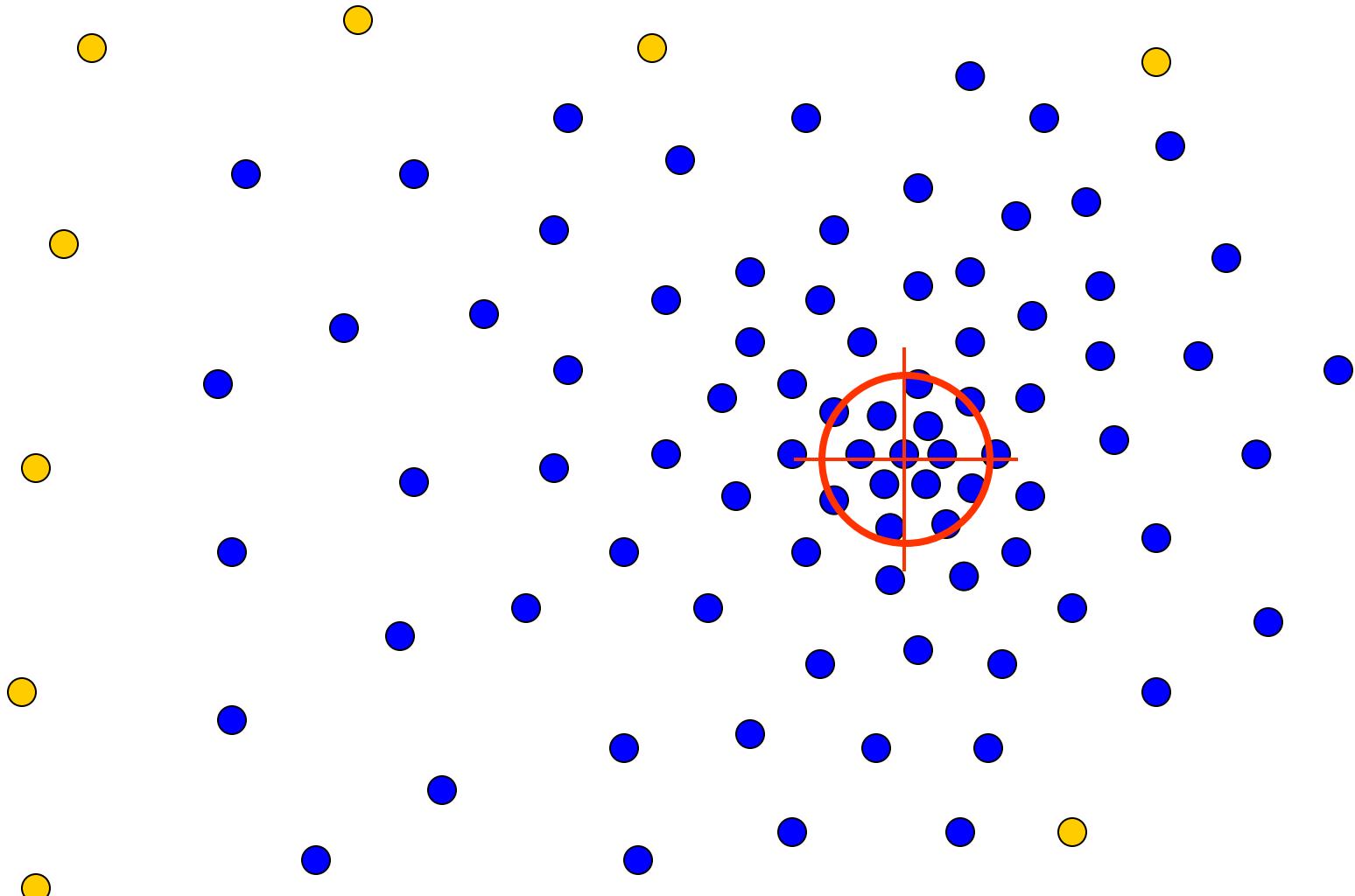
Slide by Y. Ukrainitz & B. Sarel

# Real Modality Analysis



Slide by Y. Ukrainitz & B. Sarel

# Real Modality Analysis

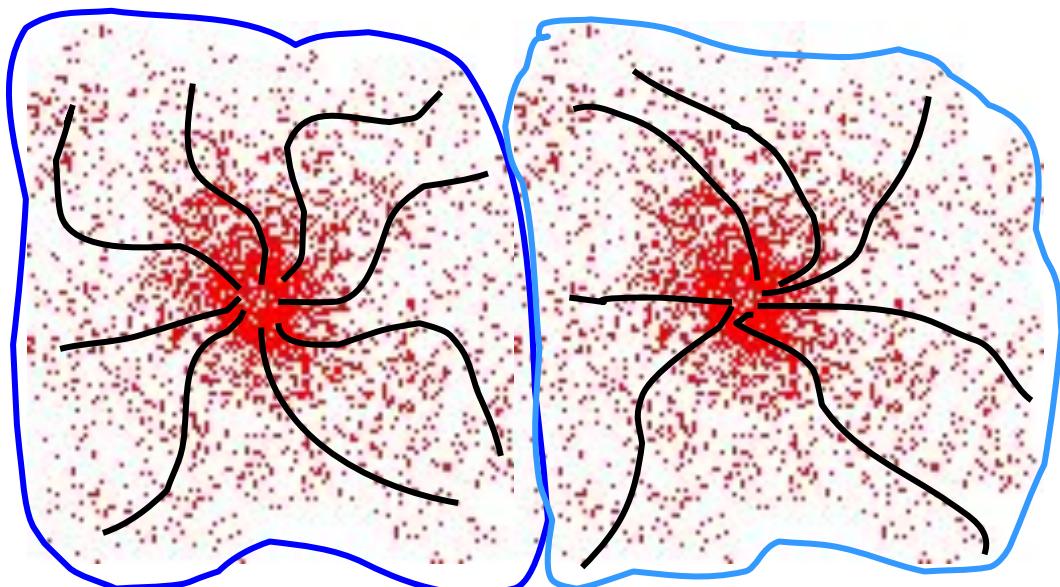


The **blue** data points were traversed by the windows towards the mode.

Slide by Y. Ukrainitz & B. Sarel

# Mean-Shift Clustering

- Initialize not just 1 window but a window for every pixel
- All pixels that end up in the same location belong to the same **cluster**
- **Attraction basin:** the region for which all windows end up in the same location



# Mean-Shift Segmentation Results

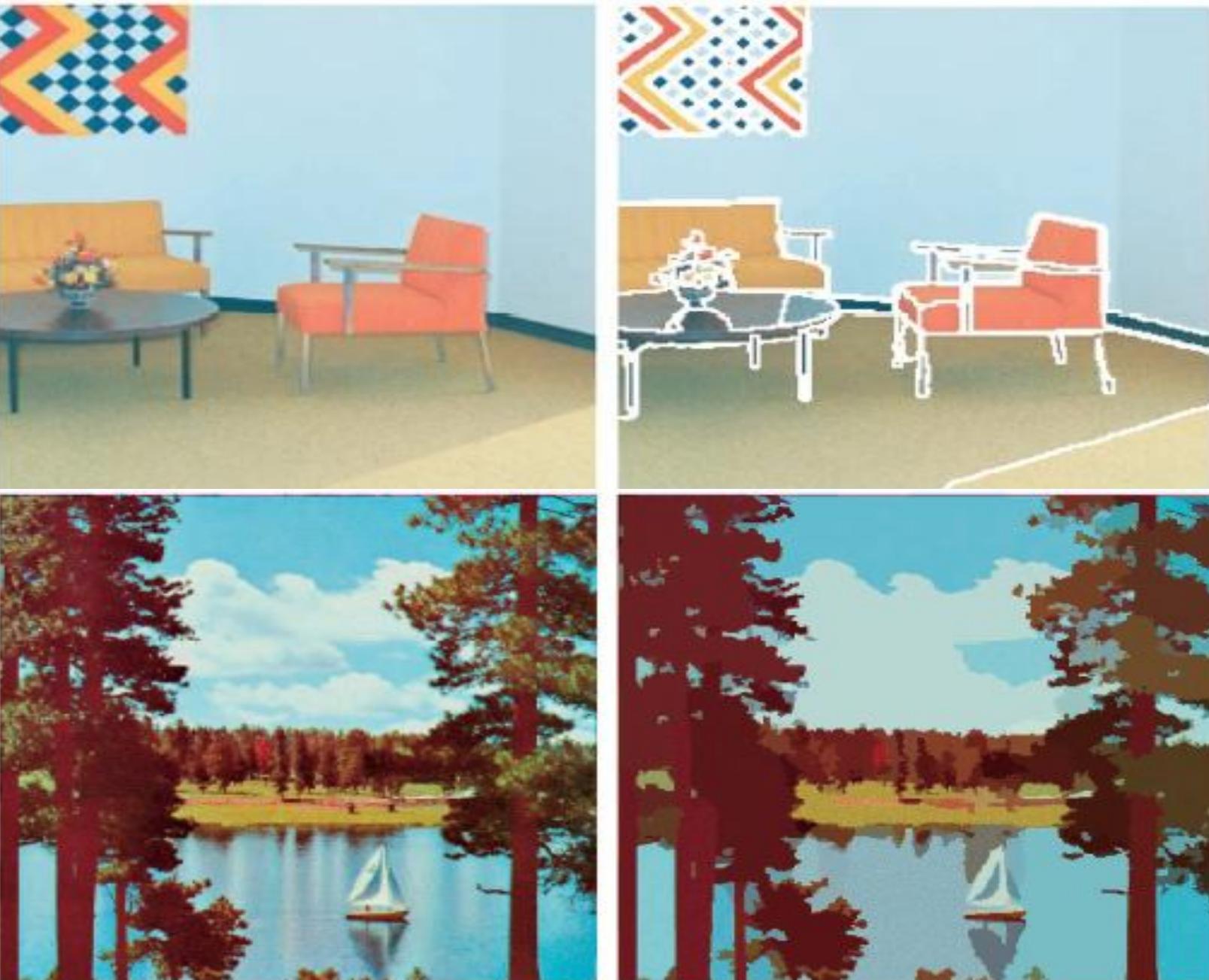


[http://www.caip.rutgers.edu/~comanici/MSPAM  
l/msPamiResults.html](http://www.caip.rutgers.edu/~comanici/MSPAM/l/msPamiResults.html)

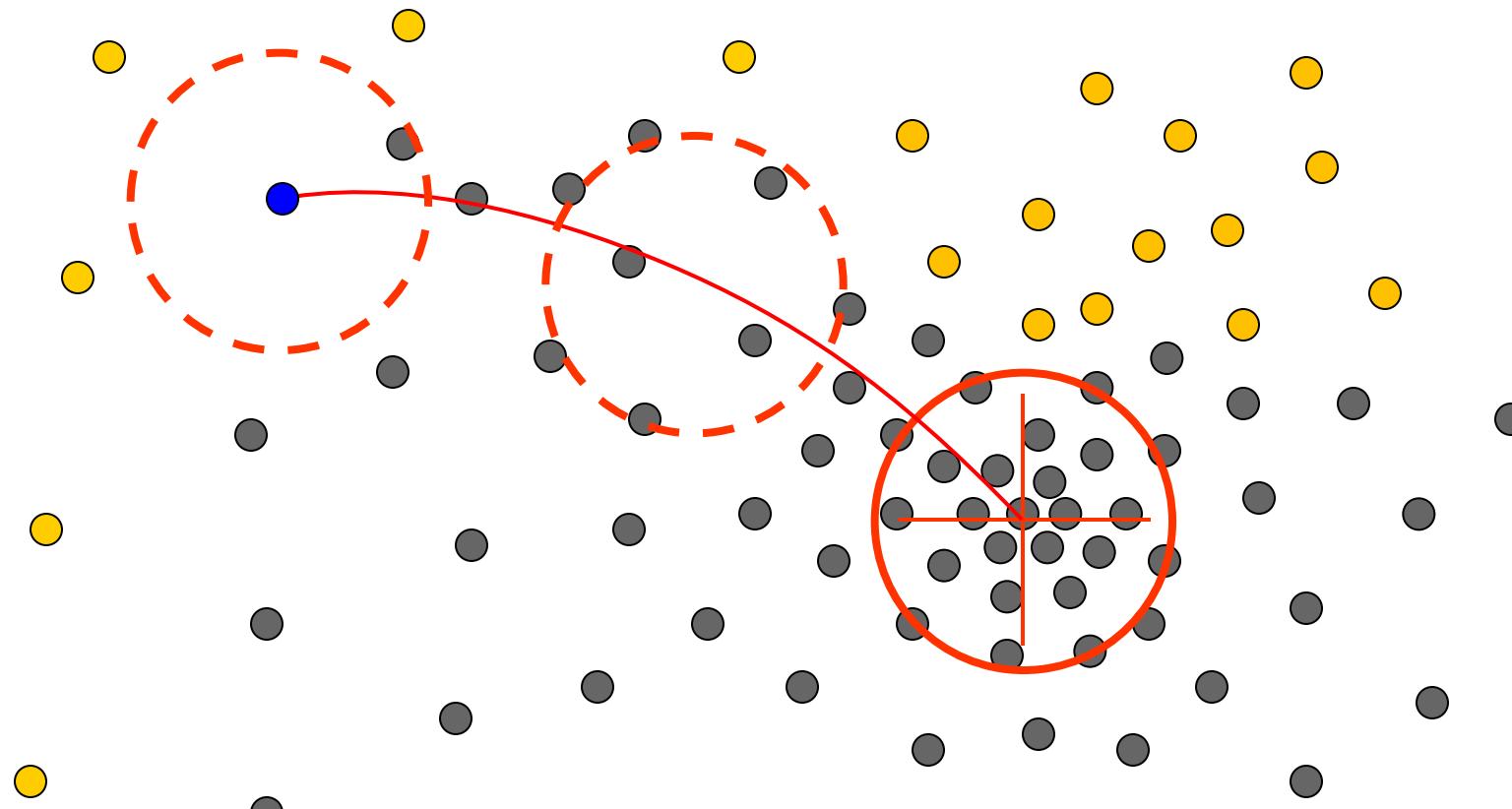
# More Results



# More Results



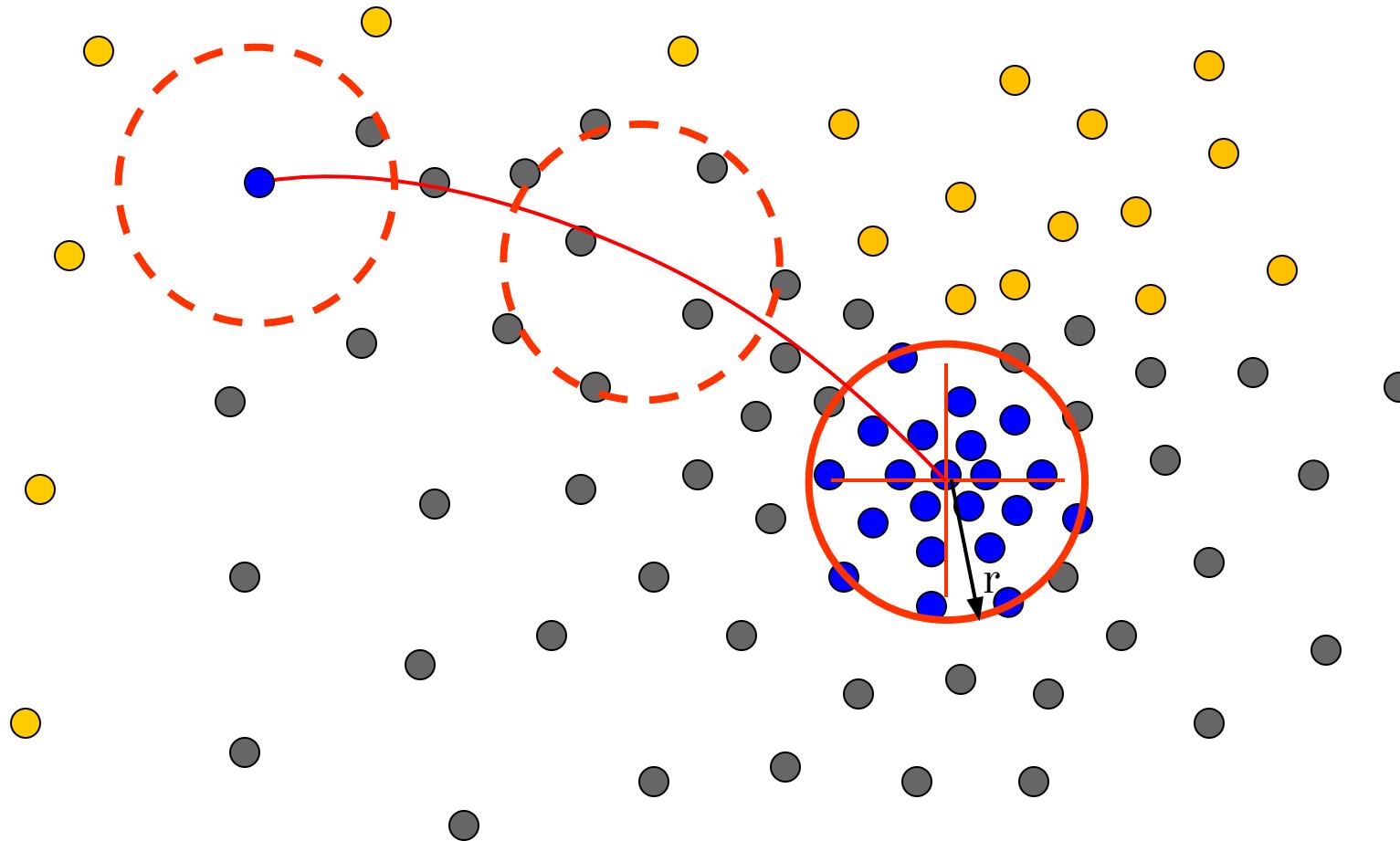
# Problem: Computational Complexity



- Need to shift one window for every pixel
- Many computations will be redundant.

Slide credit: Bastian Leibe

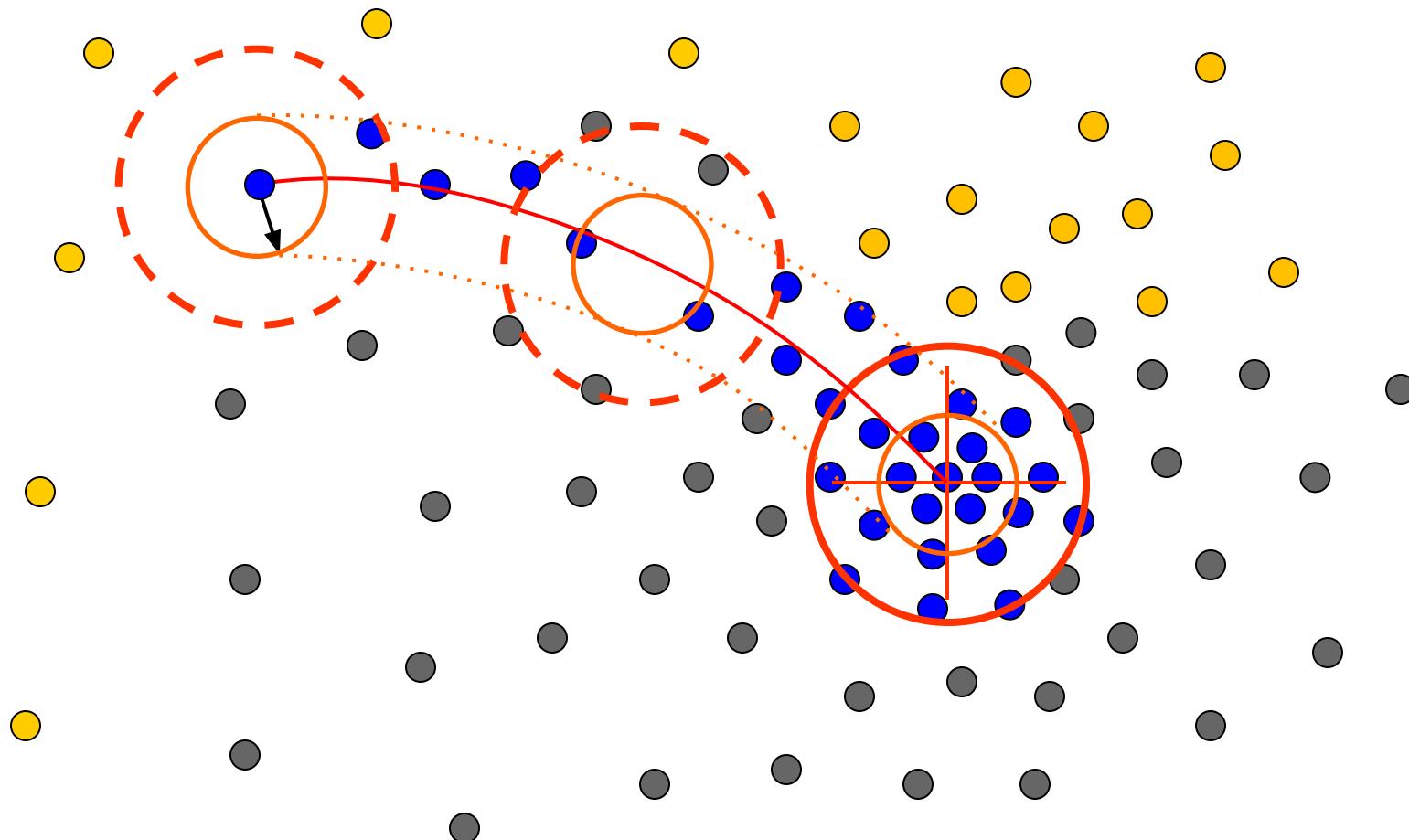
# Speedups: Basin of Attraction



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

Slide credit: Bastian Leibe

# Speedups

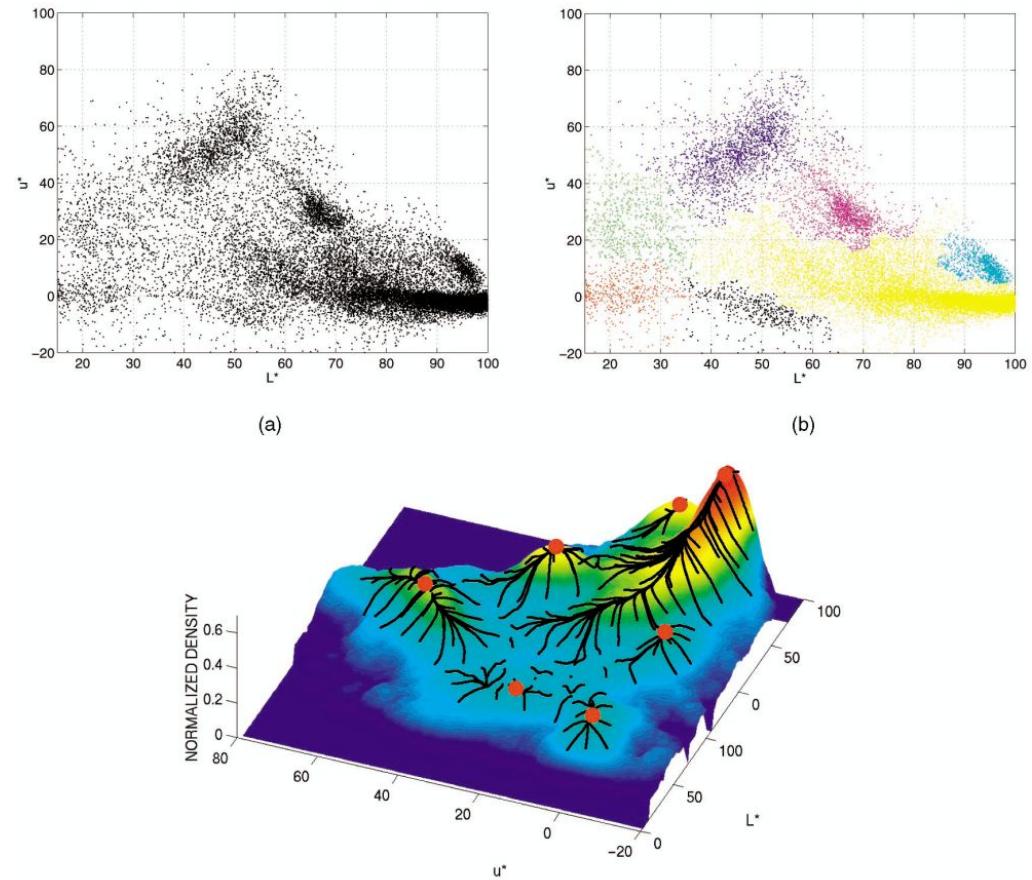


2. Assign all points within radius  $r/c$  of the search path to the mode -> reduce the number of data points to search.

Slide credit: Bastian Leibe

# Mean-Shift Clustering

- Find features (color, gradients, texture, etc)
- Initialize windows at individual pixel locations
- Perform mean shift for each window until convergence
- At every step, merge windows that have high overlap to reduce computation



# 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 :

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

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

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

[source](#)

# 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.

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

Comaniciu & Meer, 2002

# Mean-Shift pros and cons

- **Pros**

- General, application-independent algorithm
- Model-free, does not assume any prior shape (spherical, elliptical, etc.) of data clusters
- Just a single parameter (window size  $r$ )
  - $r$  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 easy
- Computationally (relatively) expensive (~2s/image)
- Does not scale well with dimension of feature space

# Summary

- Introduction to segmentation and clustering
- Gestalt theory for perceptual grouping
- Graph-based oversegmentation
- Agglomerative clustering
- K-means clustering
- Mean-shift clustering

# Next time

Cameras and Calibration

# 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.