南京大学
NANJING UNIVERSITY

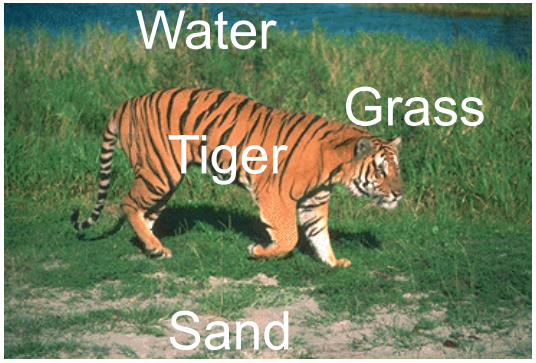
计算机视觉表征与识别
Chapter 6: Segmentation and Grouping

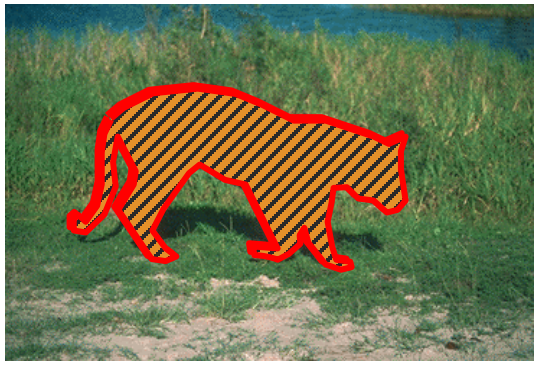
王利民

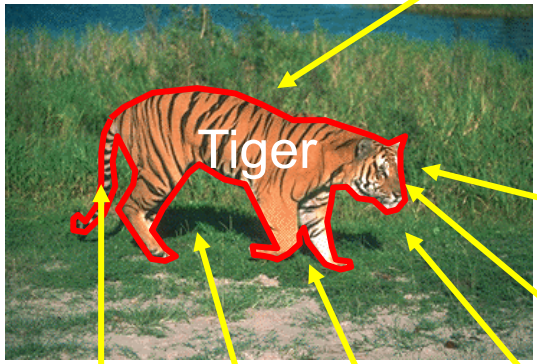
媒体计算课题组

http://mcg.nju.edu.cn/

From Pixels to Perception

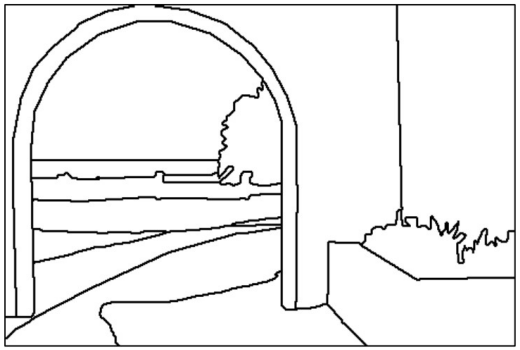




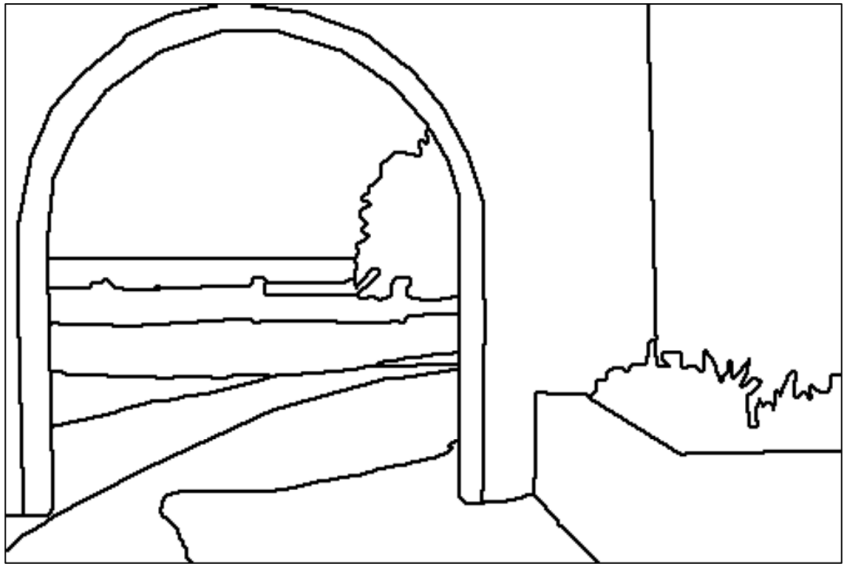


Mid-level operations of Segmentation and Grouping

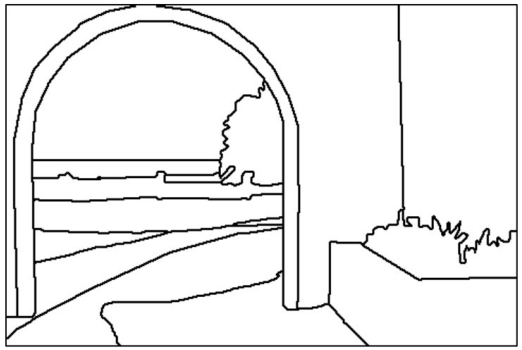
Edges



HxWx {0,1} classification problem



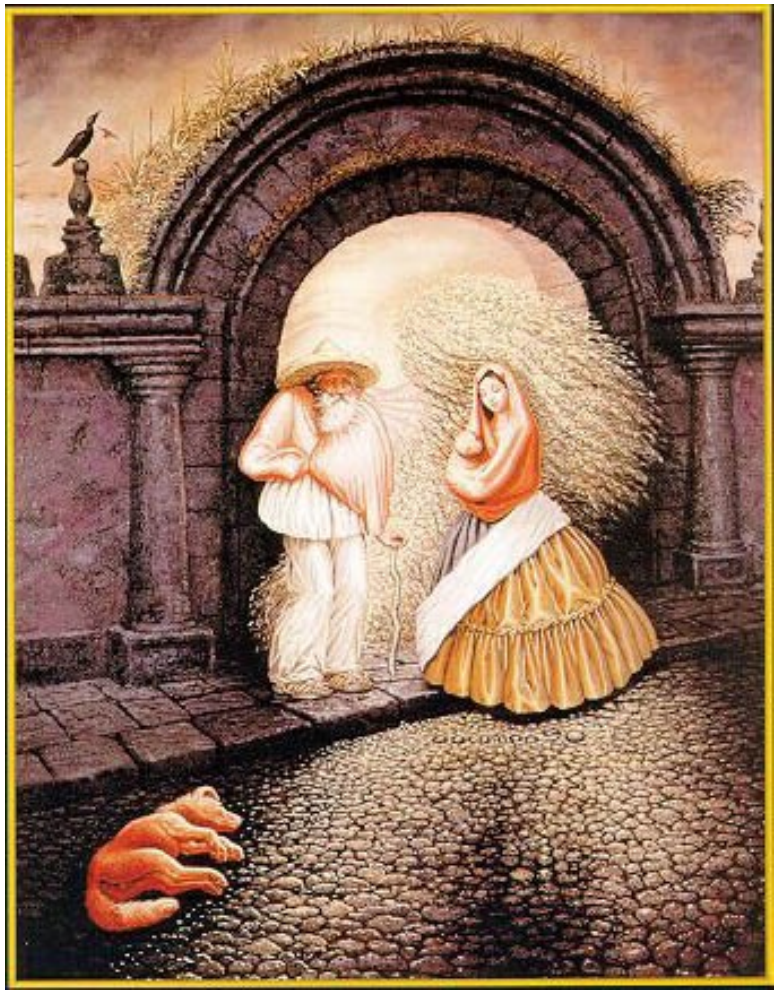
Regions



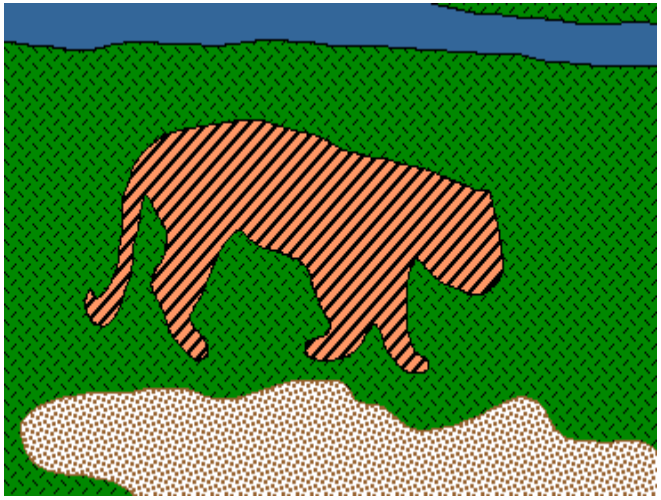
Decomposing image into K connected regions
(Clustering task)

Today's Class

- **What are grouping problems in vision**?
- Inspiration from human perception
  - Gestalt properties
- Bottom-up segmentation via clustering
  - Mode finding and mean shift: k-means, GMM, mean-shift
- Graph-based segmentation: Normalized Cut
- Oversegmentation
  - Watershed algorithm, Felzenszwalb and Huttenlocher graph-based
- Multiple segmentation



From Images to Objects

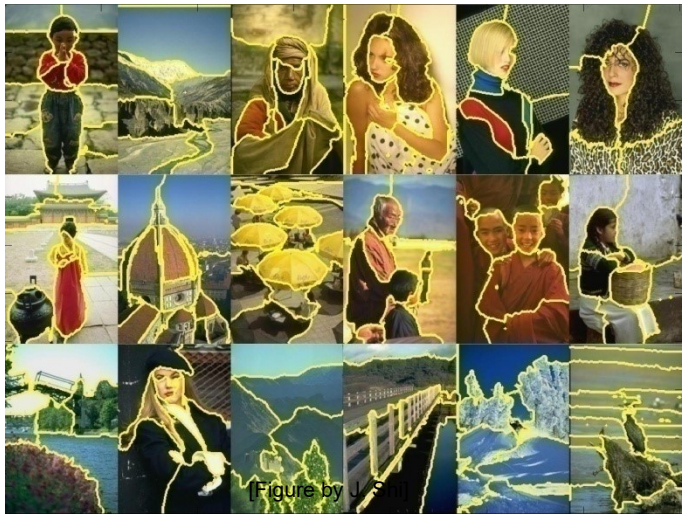


"I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightnesses and nuances of colour. Do I have "327"? No. I have sky, house, and trees." --Max Wertheimer

Grouping in vision

- Goals:
  - Gather features that belong together
  - **Obtain an intermediate representation that compactly describes key image (video) parts**
- Top down vs. bottom up segmentation
  - Top down: pixels belong together because they are from the same object
  - Bottom up: pixels belong together because they look similar
- Hard to measure success
  - What is interesting depends on the app.

Examples of grouping in vision



[Figure by J. Shi]

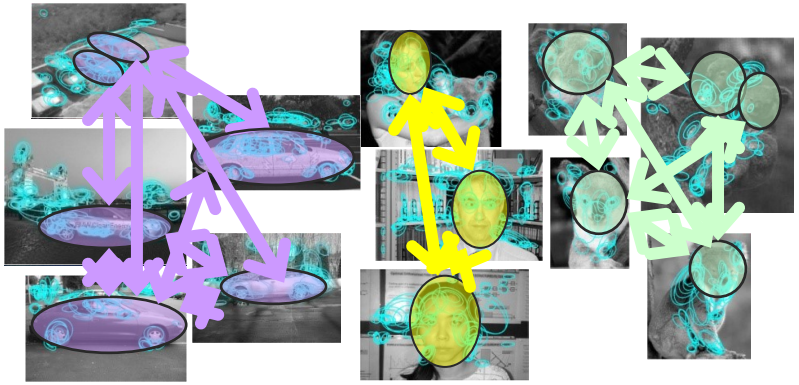
Determine image regions

[http://poseidon.csd.auth.gr/LAB_RESEARCH/Latest/imgs/SpeakDepVidIndex_img2.jpg]

Group video frames into shots

[Figure by Wang & Suter]

Figure-ground



[Figure by Grauman & Darrell]

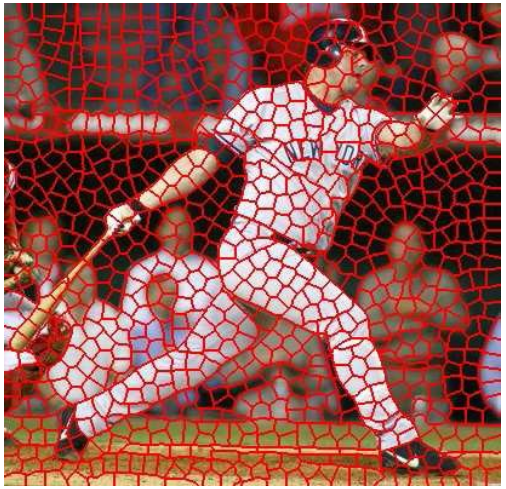
Object-level grouping

Image segmentation

Goal: Group pixels into meaningful or perceptually similar regions

Segmentation for efficiency: “superpixels”

[Felzenszwalb and Huttenlocher 2004]

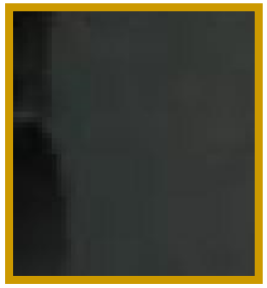


[Shi and Malik 2001]

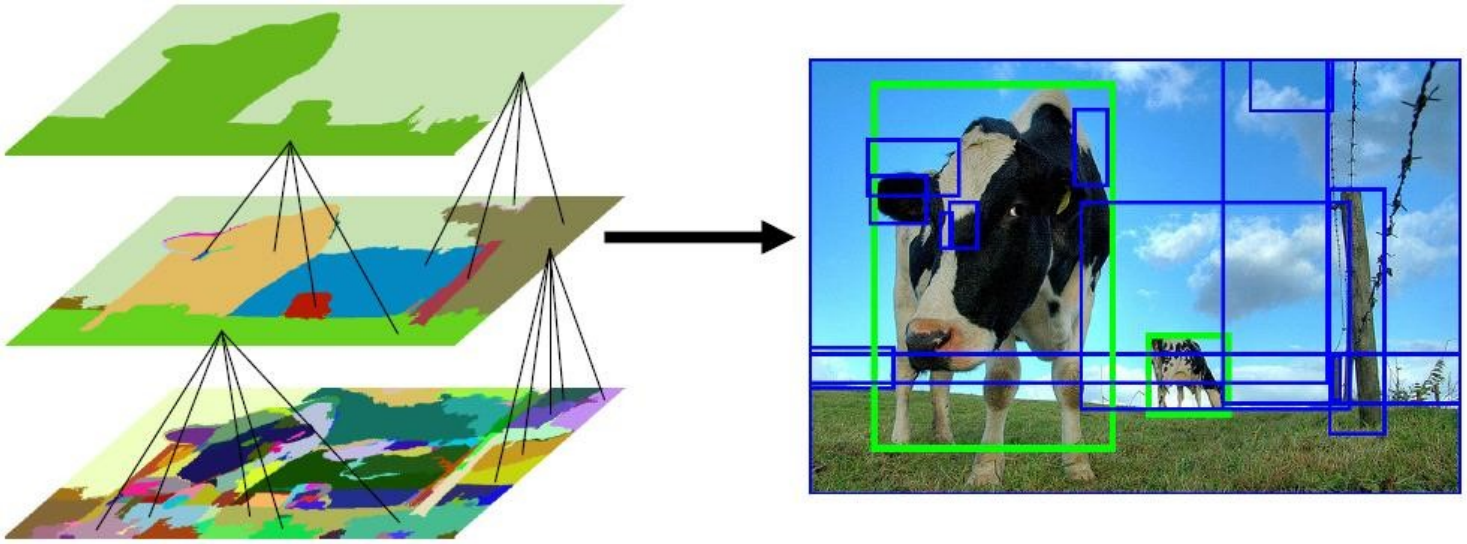
Segmentation for feature support

50x50 Patch

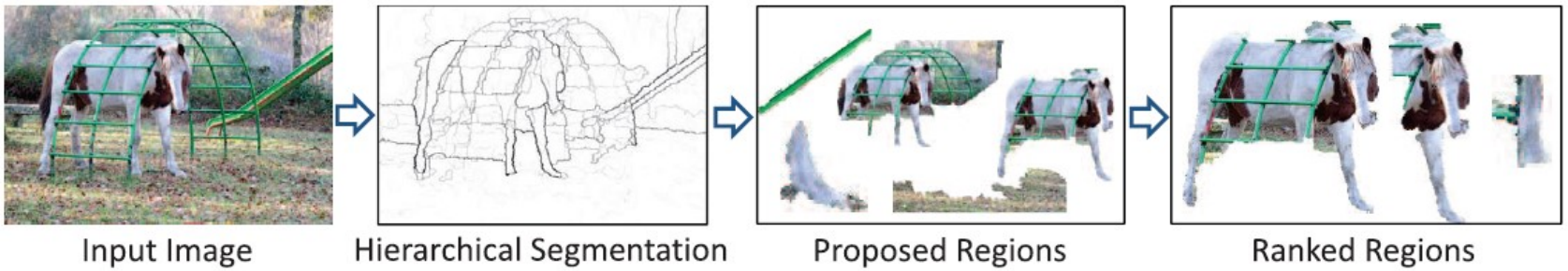
50x50 Patch



Segmentation for object proposals



"Selective Search" [Sande, Uijlings et al. ICCV 2011, IJCV 2013]

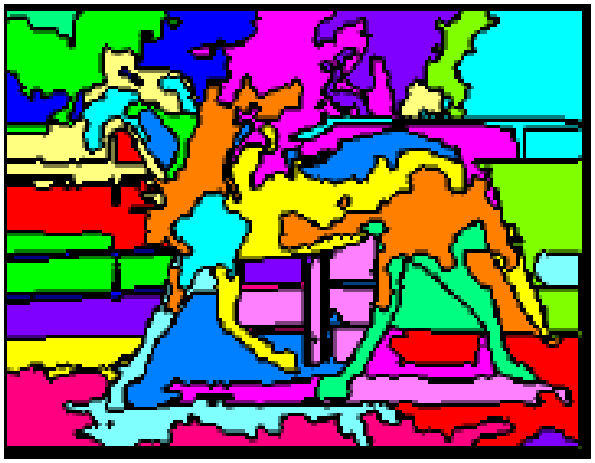


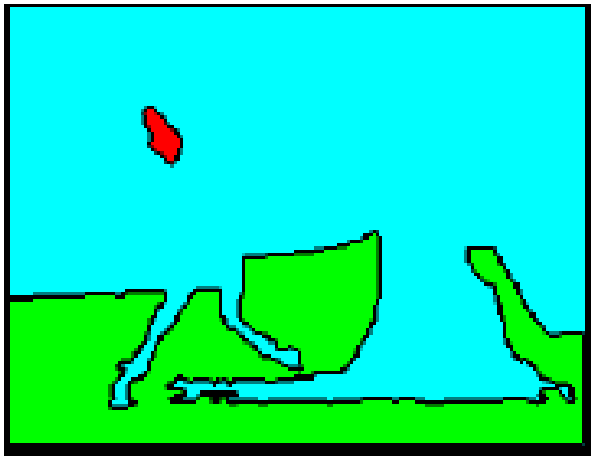
[Endres Hoiem ECCV 2010, IJCV 2014]

Segmentation as a result

Rother et al. 2004

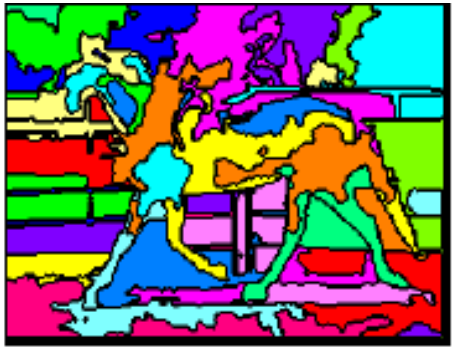
Types of segmentations

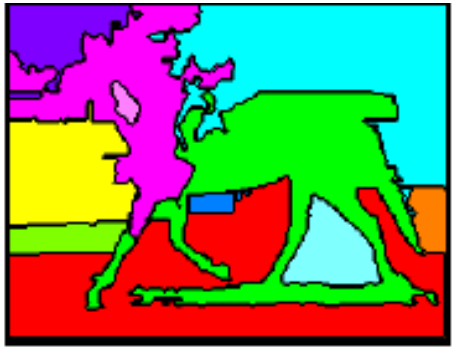


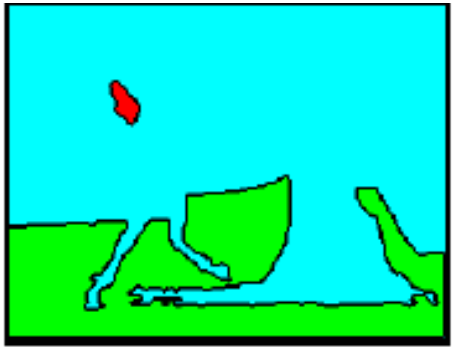


Oversegmentation

Undersegmentation







Multiple Segmentations

Today's Class

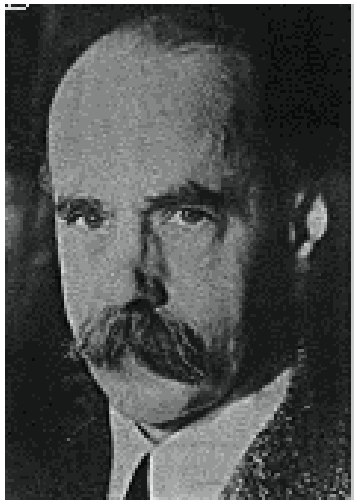
- What are grouping problems in vision?
- **Inspiration from human perception**
  - Gestalt properties
- Bottom-up segmentation via clustering
  - Mode finding and mean shift: k-means, GMM, mean-shift
- Graph-based segmentation: Normalized Cut
- Oversegmentation
  - Watershed algorithm, Felzenszwalb and Huttenlocher graph-based
- Multiple segmentation

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)

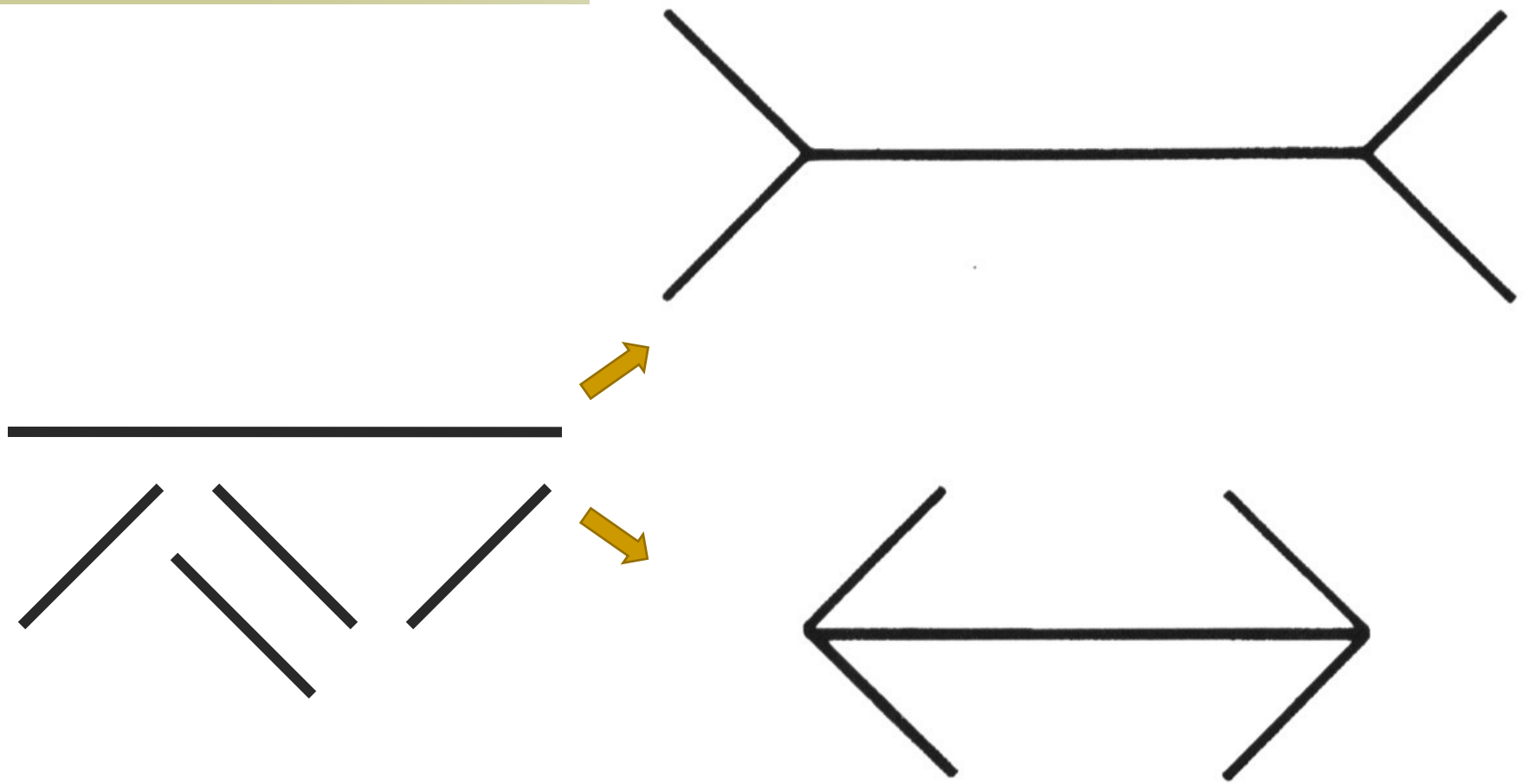
"I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightnesses and nuances of colour. Do I have "327"? No. I have sky, house, and trees."

Max Wertheimer
(1880-1943)



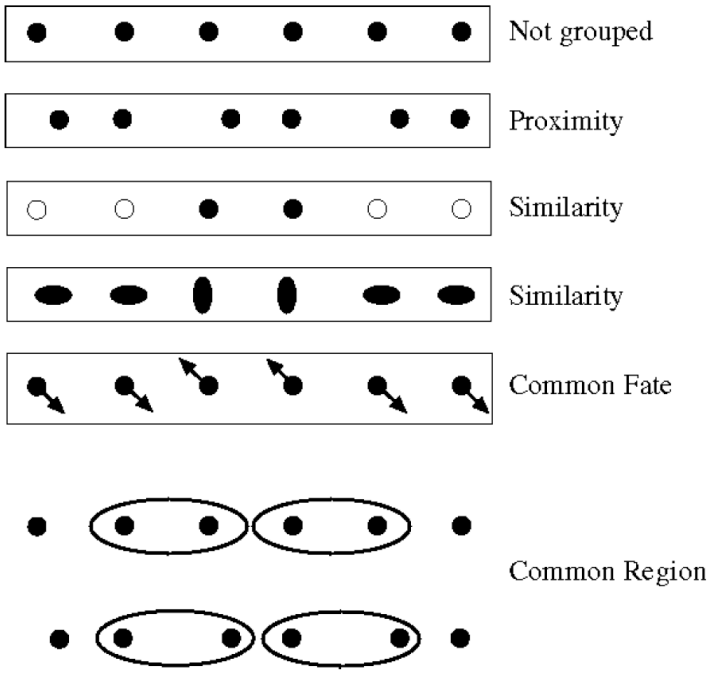
Untersuchungen zur Lehre von der Gestalt,
***Psychologische Forschung*, Vol. 4, pp. 301-350, 1923**
http://psy.ed.asu.edu/~classics/Wertheimer/Forms/forms.htm

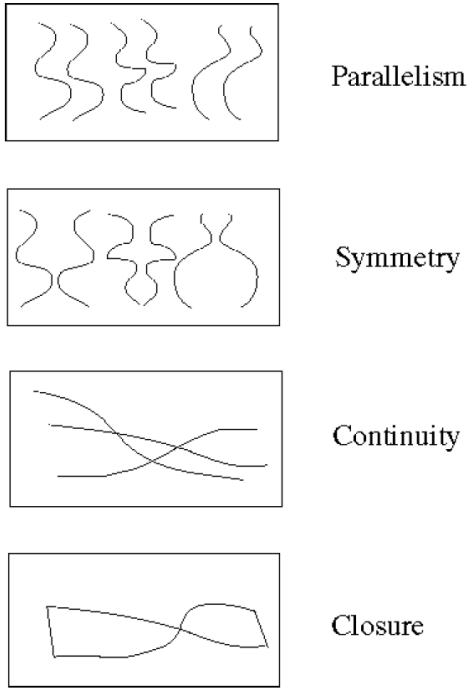
Gestaltism



The Muller-Lyer illusion

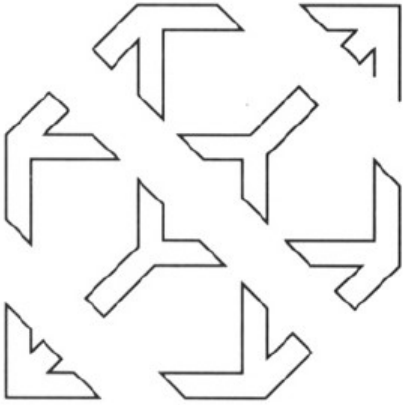
Gestalt Factors



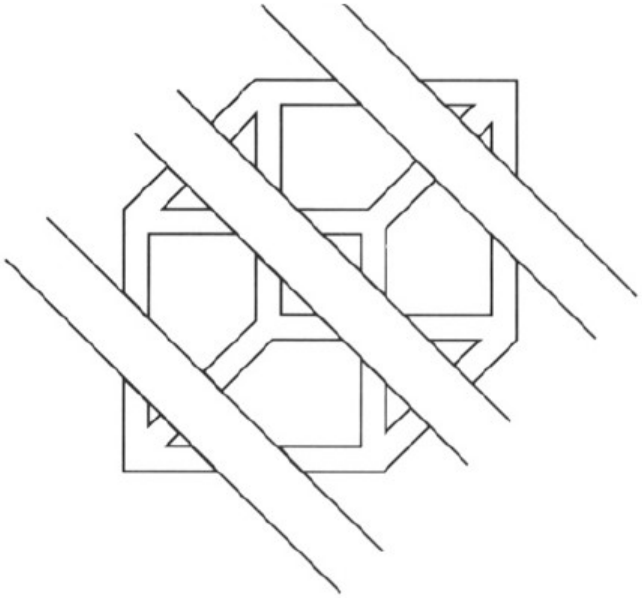


- These factors make intuitive sense, but are very difficult to translate into algorithms.

Continuity through Occlusion Cues



Continuity through Occlusion Cues



Continuity, explanation by occlusion

Continuity through Occlusion Cues

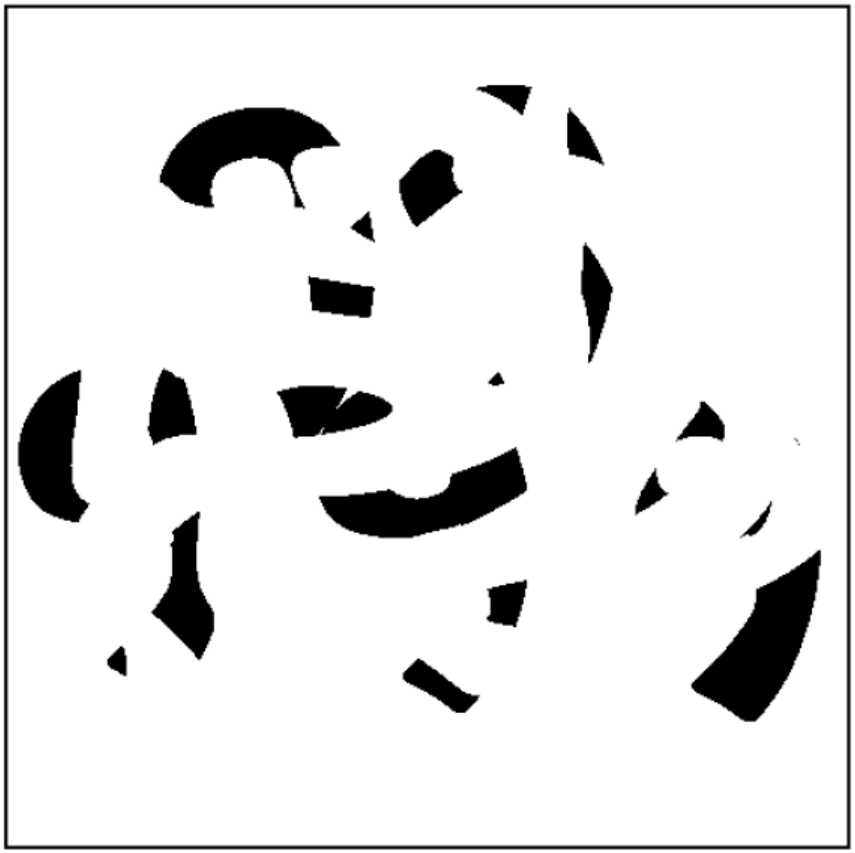
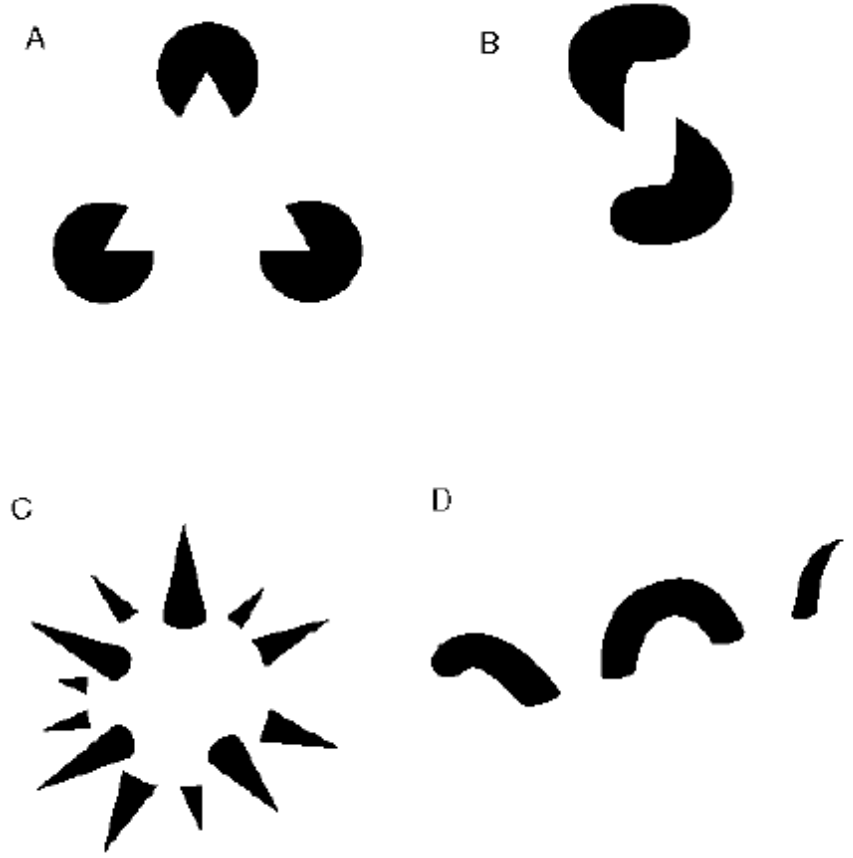


Image source: Forsyth & Ponce

Continuity through Occlusion Cues

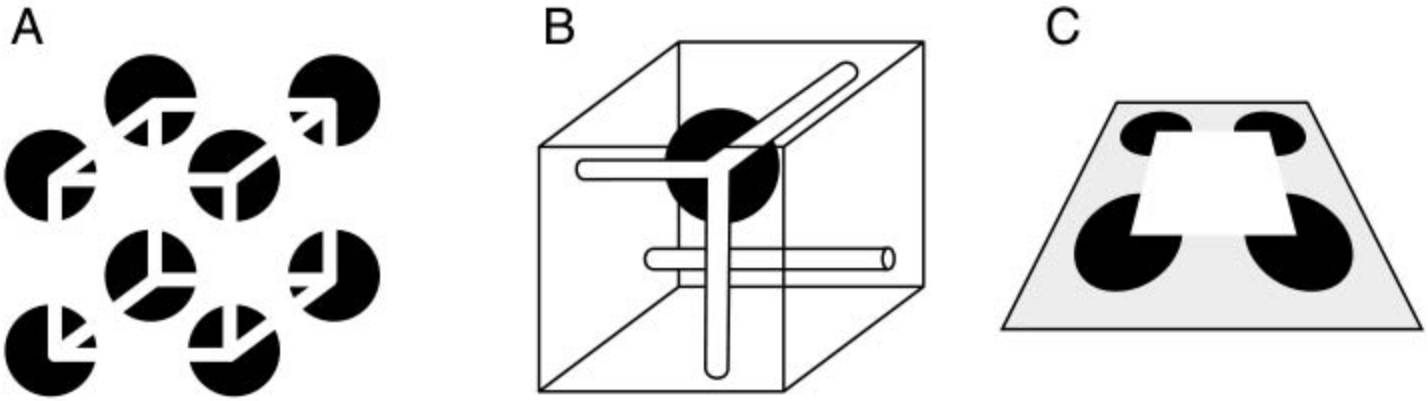
Image source: Forsyth & Ponce

Grouping by invisible completion



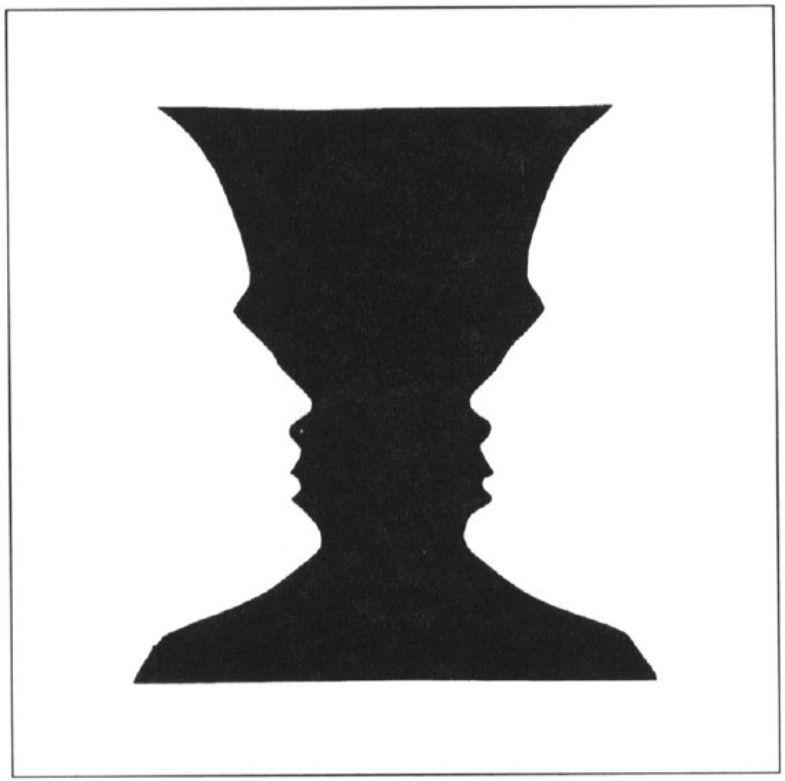
From Steve Lehar: The Constructive Aspect of Visual Perception

Grouping involves global interpretation



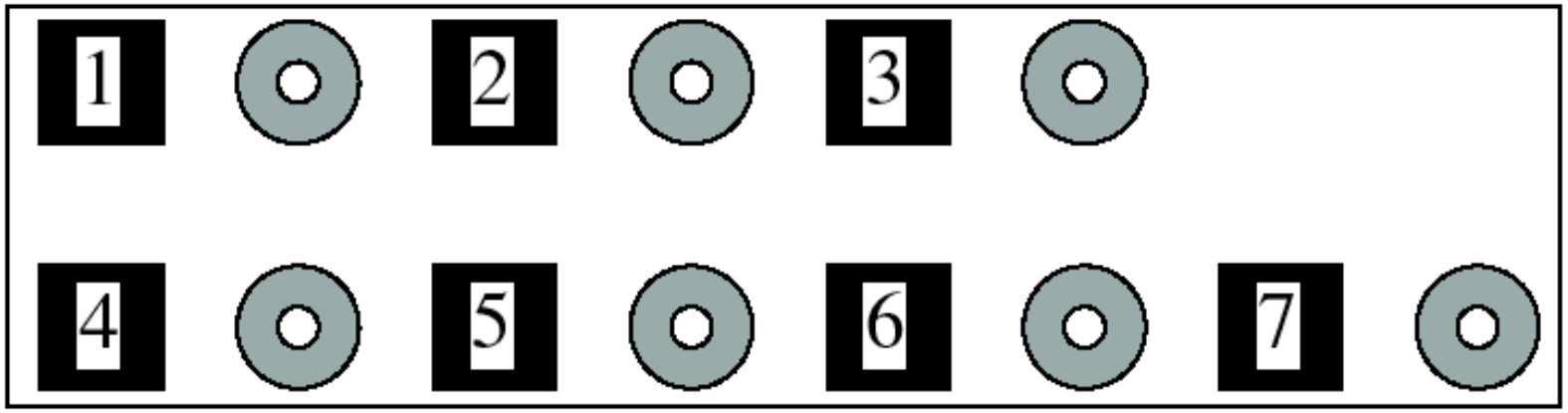
From Steve Lehar: The Constructive Aspect of Visual Perception

Figure-Ground Discrimination



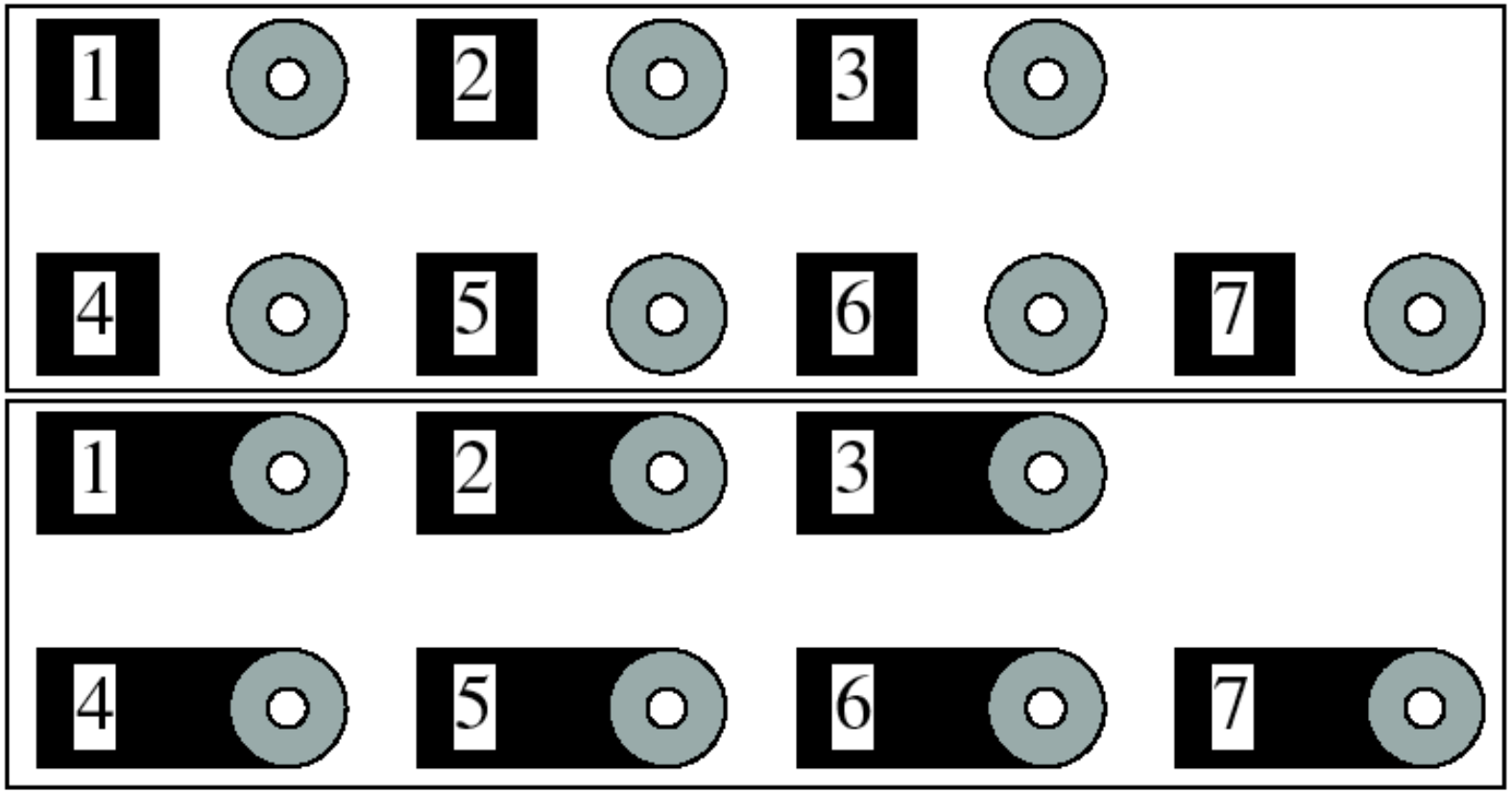
The Ultimate Gestalt?

Grouping phenomena in real life



Forsyth & Ponce, Figure 14.7

Grouping phenomena in real life



Forsyth & Ponce, Figure 14.7

Gestalt cues

- Good intuition and basic principles for grouping
- Basis for many ideas in segmentation and occlusion reasoning
- Some (e.g., symmetry) are difficult to implement in practice

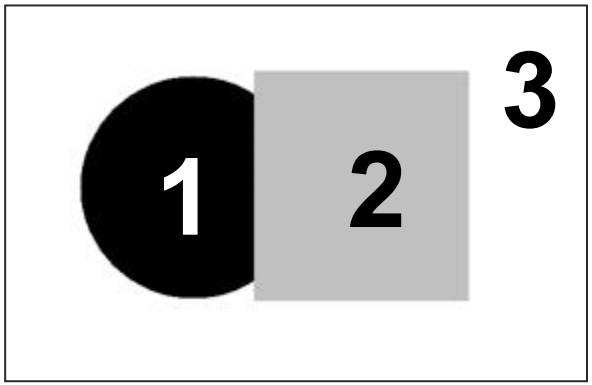
Today's Class

- What are grouping problems in vision?
- Inspiration from human perception
  - Gestalt properties
- **Bottom-up segmentation via clustering**
  - Mode finding and mean shift: k-means, GMM, mean-shift
- Graph-based segmentation: Normalized Cut
- Oversegmentation
  - Watershed algorithm, Felzenszwalb and Huttenlocher graph-based
- Multiple segmentation

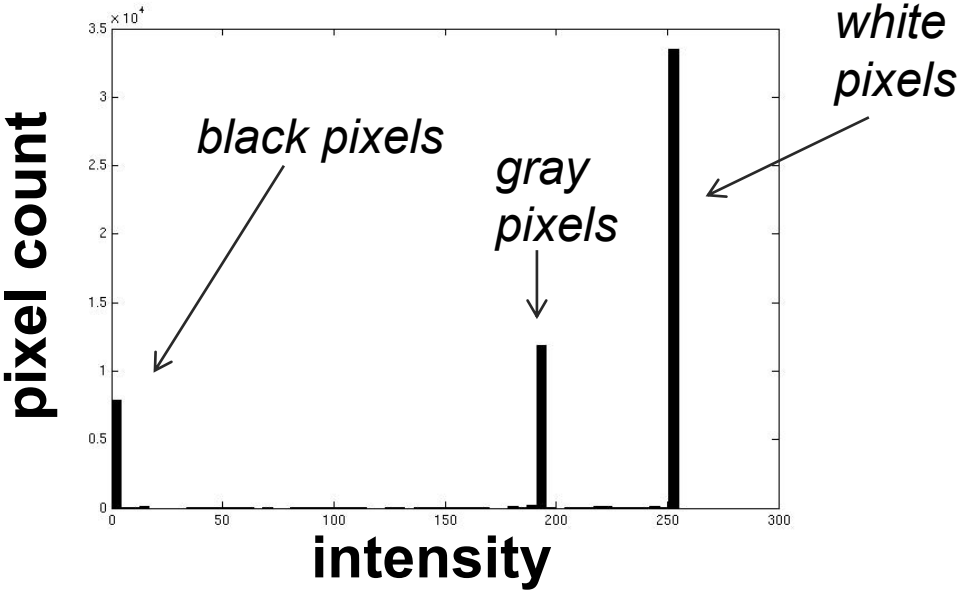
Clustering

- Clustering algorithms:
  - Unsupervised learning
  - Detect patterns in unlabeled data
    - E.g. group emails or search results
    - E.g. find categories of customers
    - E.g. group pixels into regions
  - Useful when don't know what you're looking for
  - Requires data, but no labels

Image segmentation: toy example

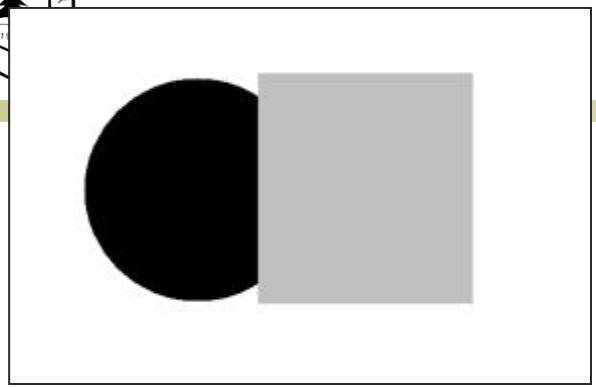


input image

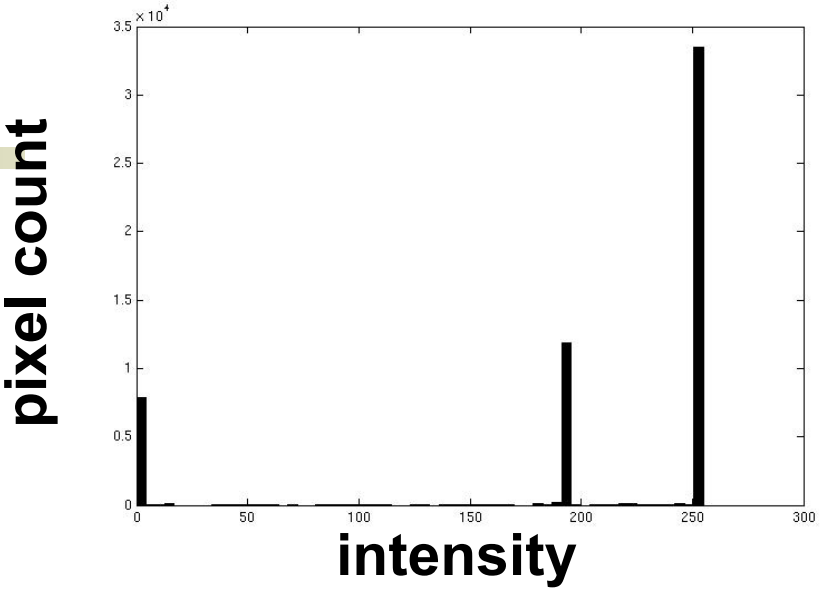


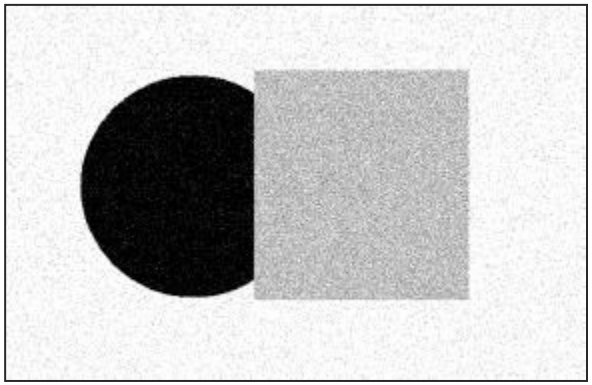
- These intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., *segment* the image based on the intensity feature.
- What if the image isn't quite so simple?

Slide credit: Kristen Grauman

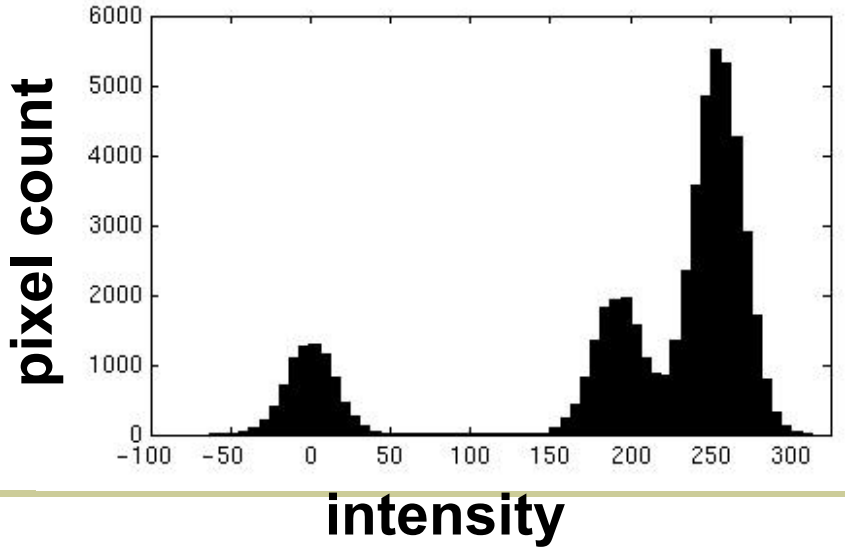


input image

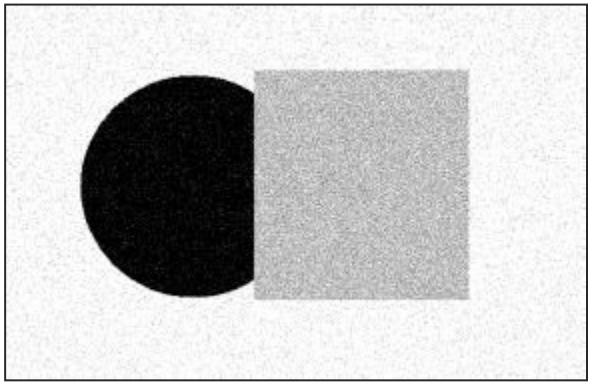




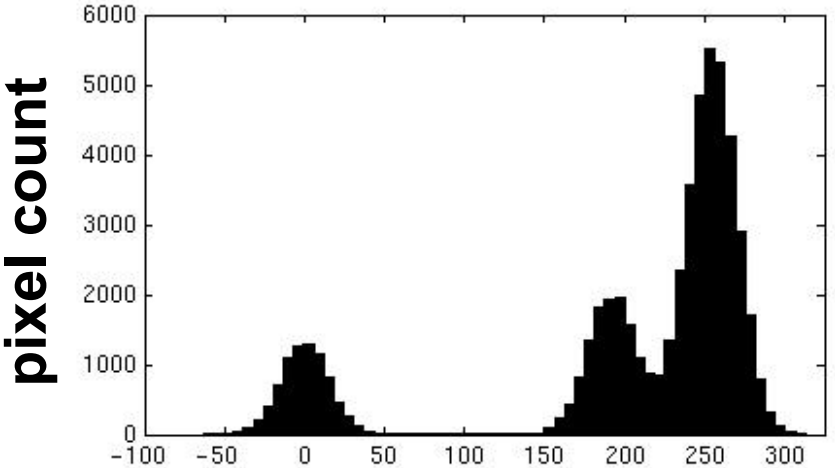
input image



Slide credit: Kristen Grauman

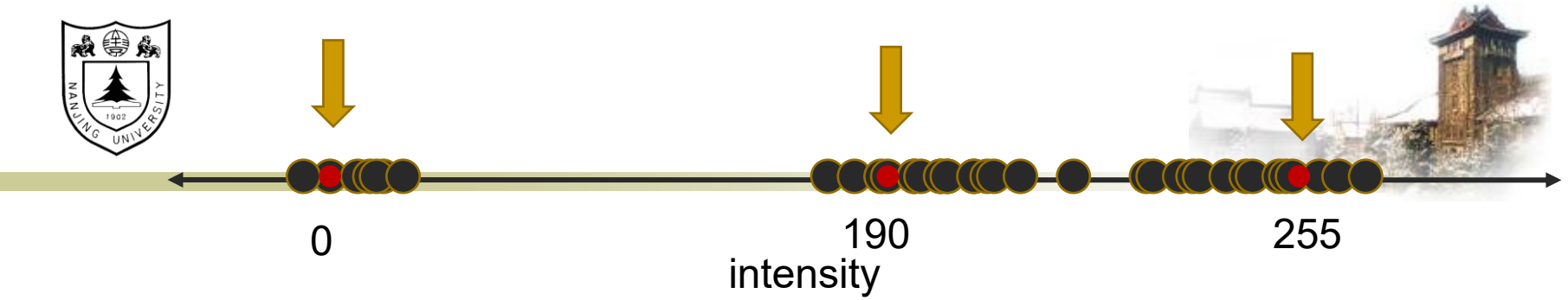


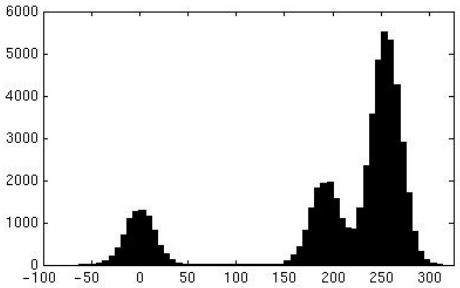
input image

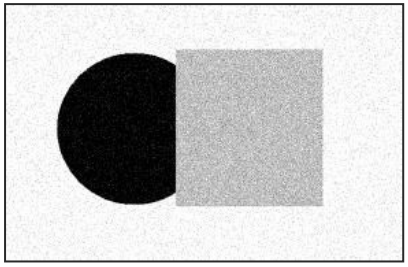


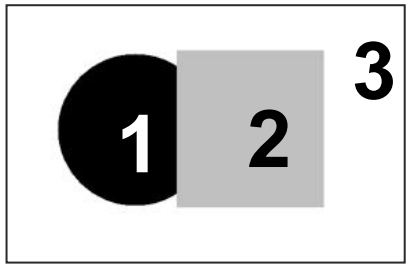
- Now how to determine the three main intensities that define our groups?
- We need to ***cluster.***

Slide credit: Kristen Grauman









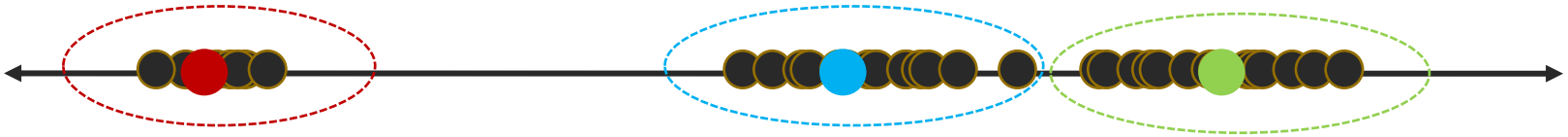
- Goal: choose three "centers" as the representative intensities, and label every pixel according to which of these centers it is nearest to.
- Best cluster centers are those that minimize SSD between all points and their nearest cluster center c_i:

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

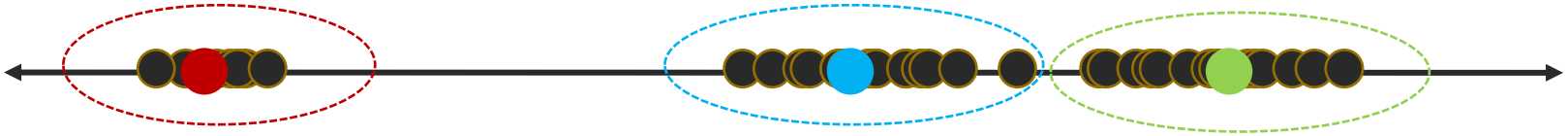
Slide credit: Kristen Grauman

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.



Slide credit: Kristen Grauman

K-means clustering

- Basic idea: randomly initialize the *k* cluster centers, and iterate between the two steps we just saw.

1. Randomly initialize the cluster centers, c_1, ..., c_K
2. Given cluster centers, determine points in each cluster
   - For each point p, find the closest c_i. Put p into cluster i
3. Given points in each cluster, solve for c_i
   - Set c_i to be the mean of points in cluster i
4. If c_i have changed, repeat Step 2

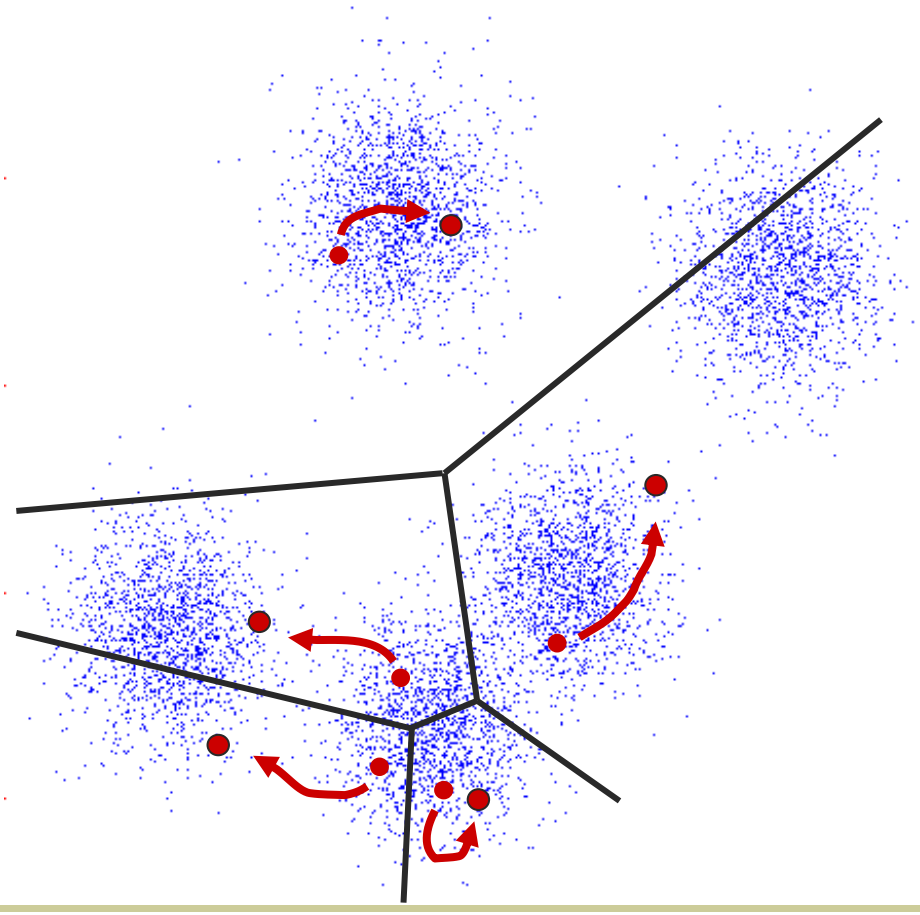
Properties

- Will always converge to *some* solution
- Can be a "local minimum"
  - does not always find the global minimum of objective function:

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

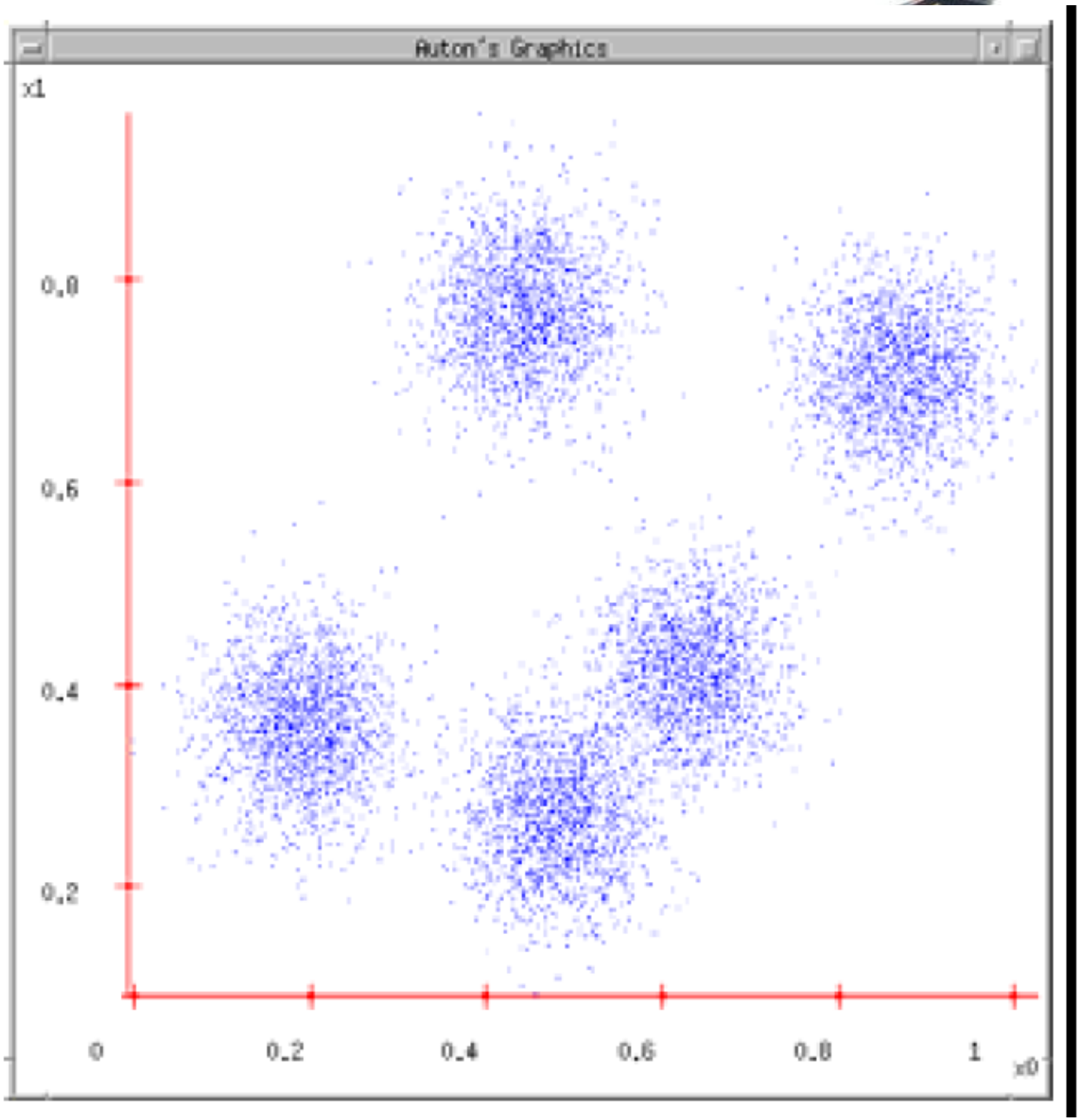
K-Means

- An iterative clustering algorithm
  - Pick K random points as cluster centers (means)
  - Alternate:
    - Assign data instances to closest mean
    - Assign each mean to the average of its assigned points
  - Stop when no points' assignments change



K-means

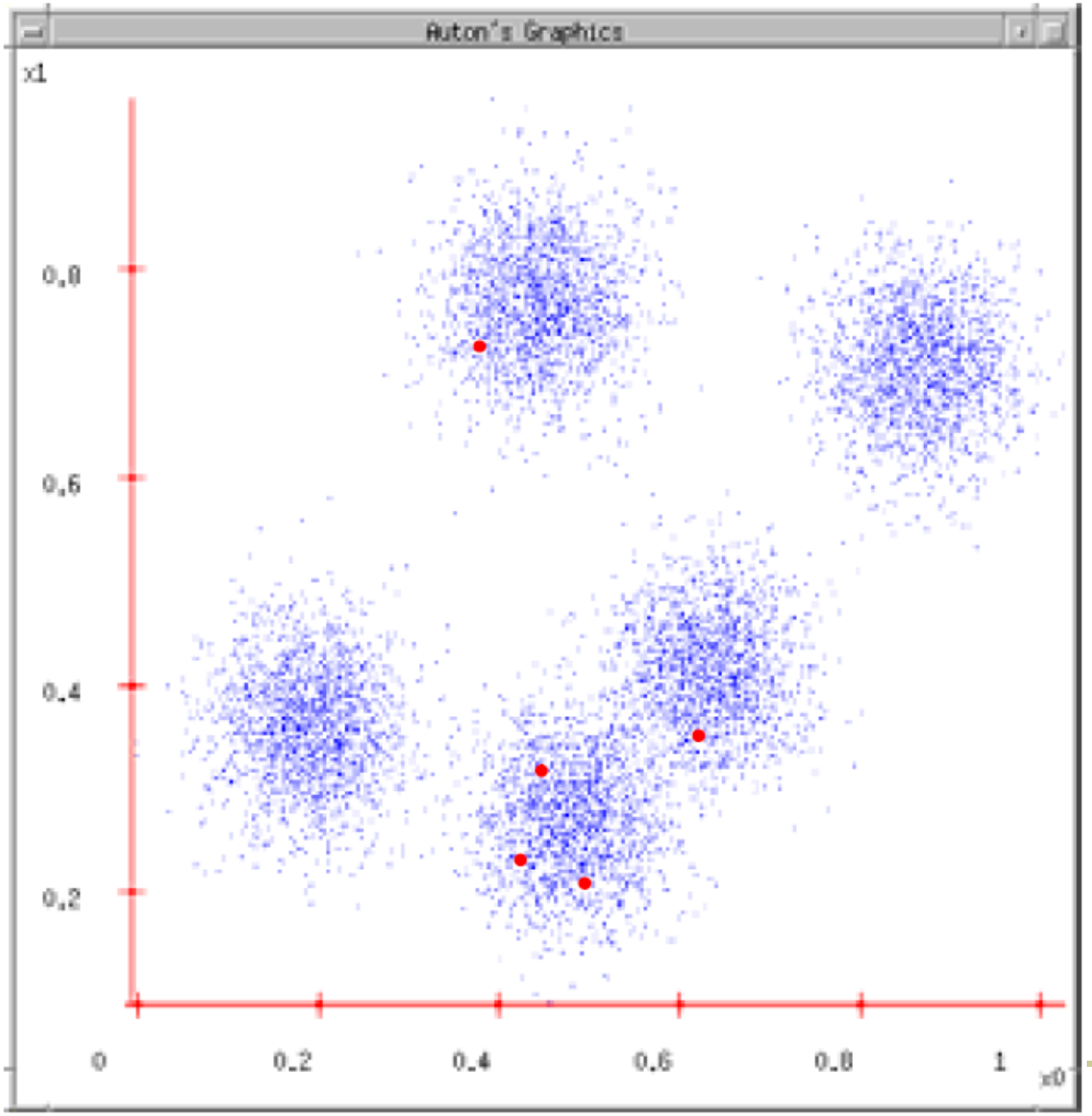
1. Ask user how many clusters they'd like. *(e.g. k=5)*



Slide credit Andrew Moore

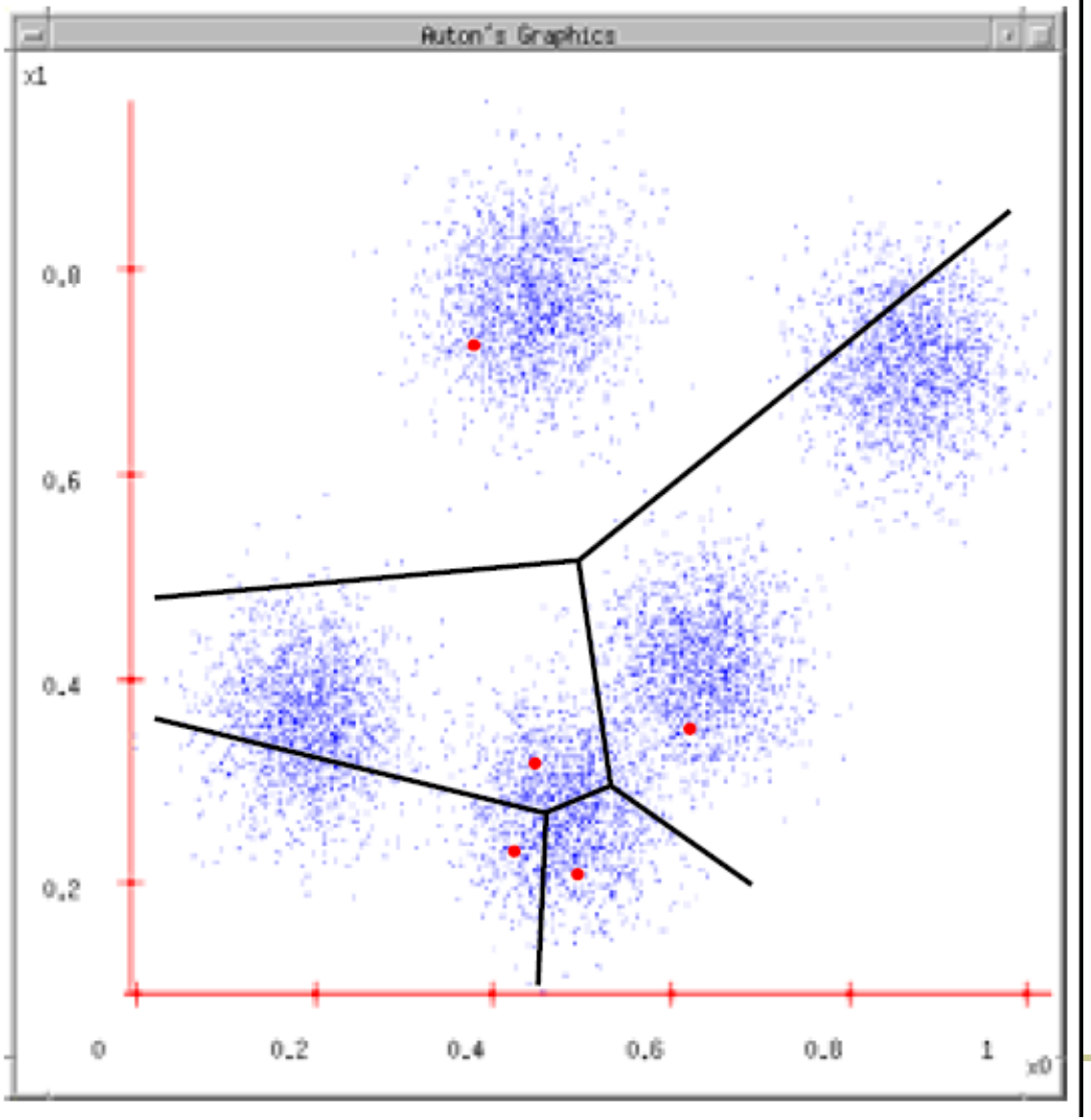
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



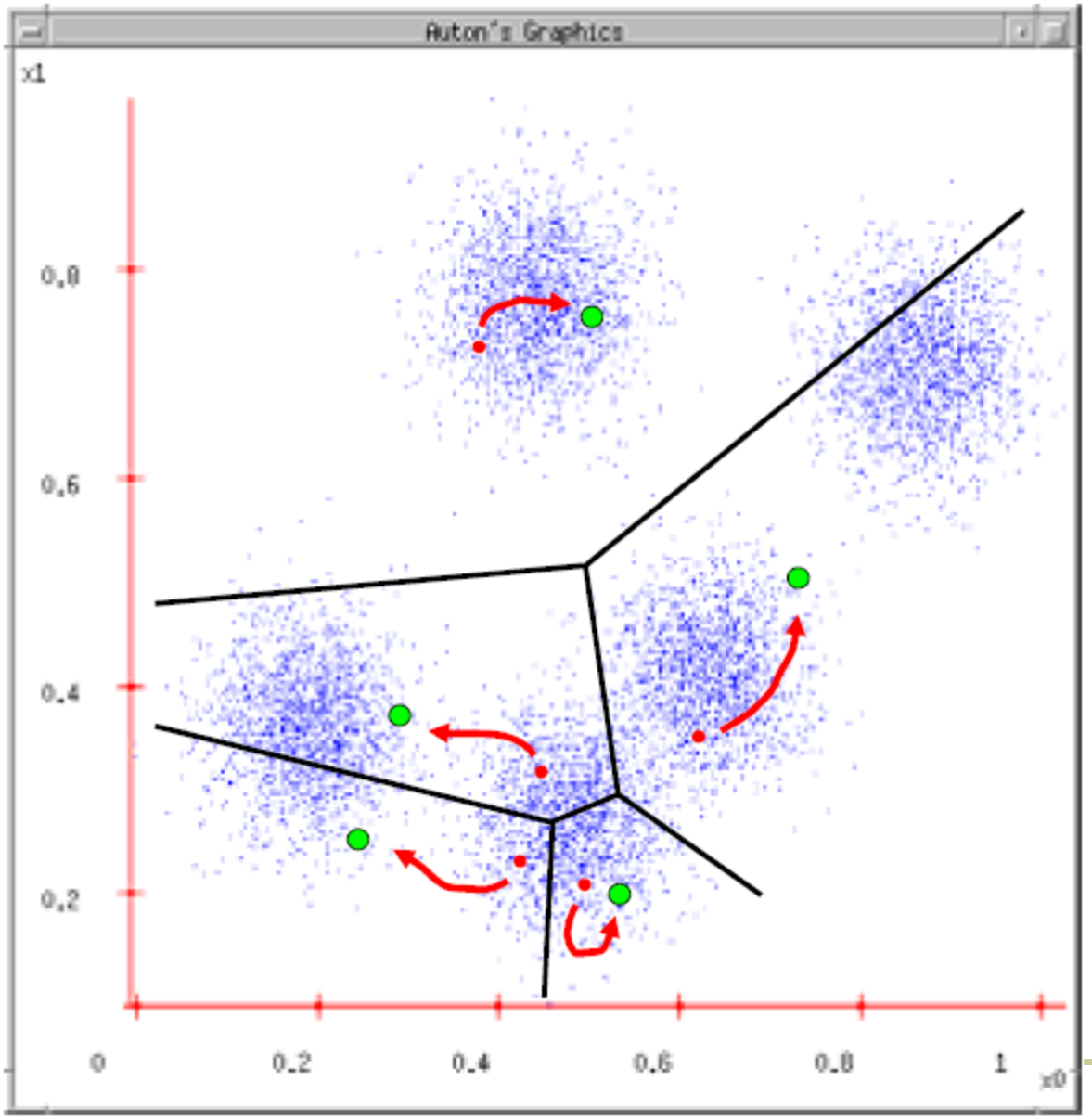
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



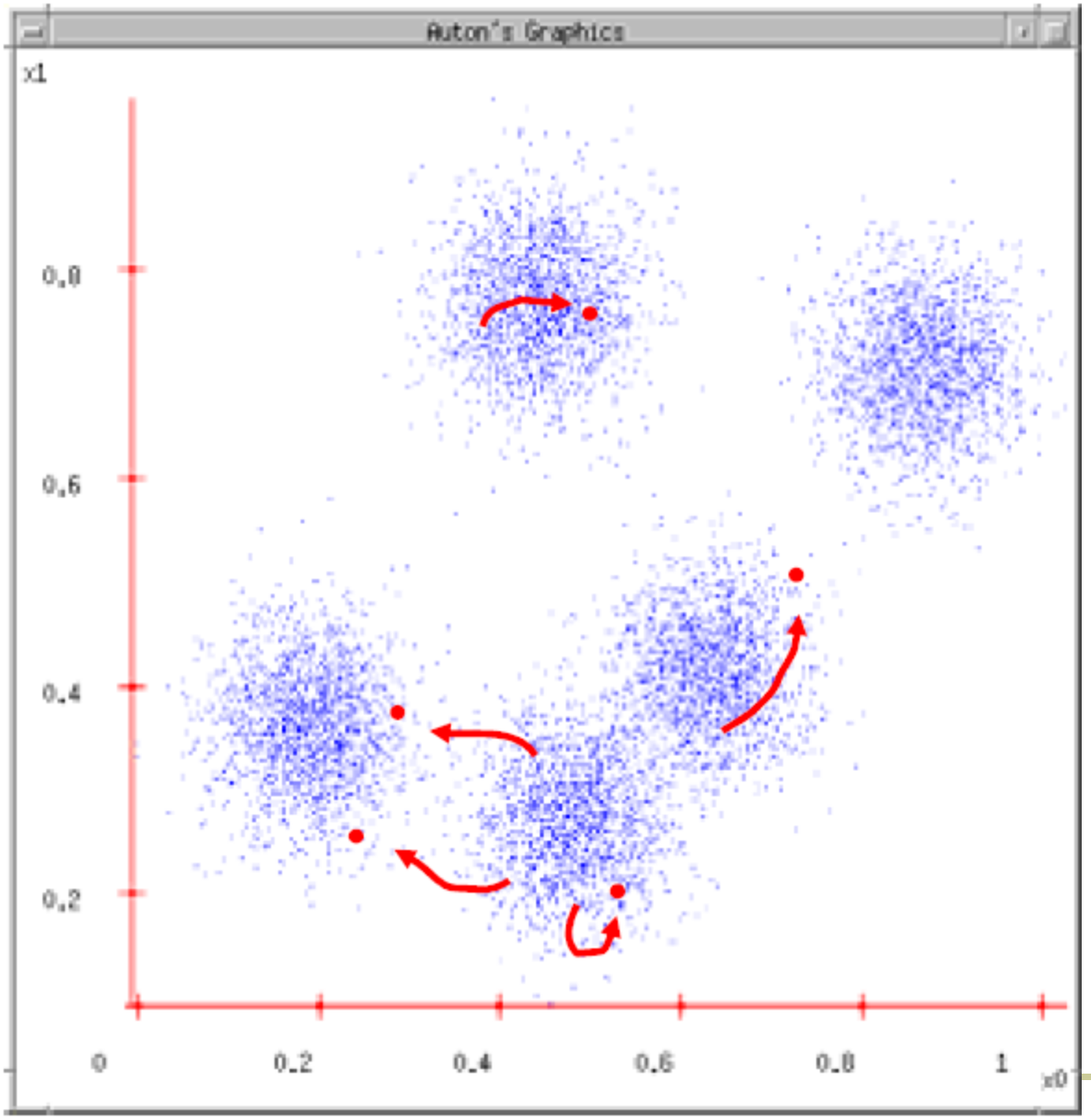
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



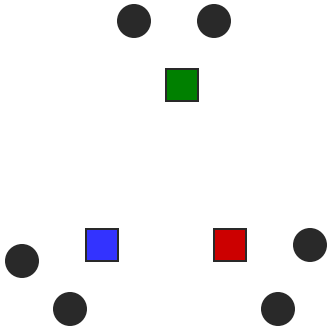
K-means

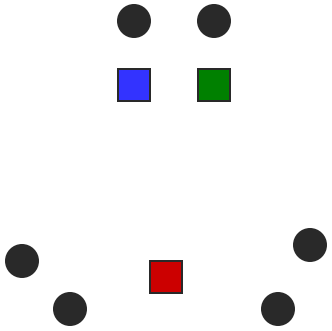
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!



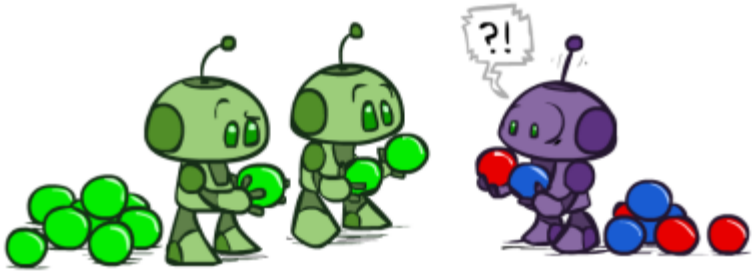
Initialization

- K-means is non-deterministic
  - Requires initial means
  - It does matter what you pick!
  - What can go wrong?
  - Various schemes for preventing this kind of thing

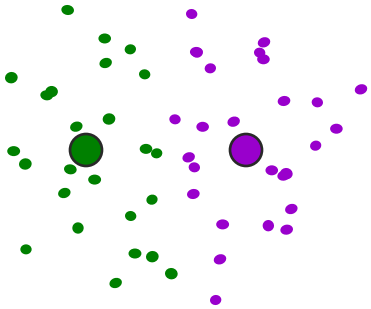


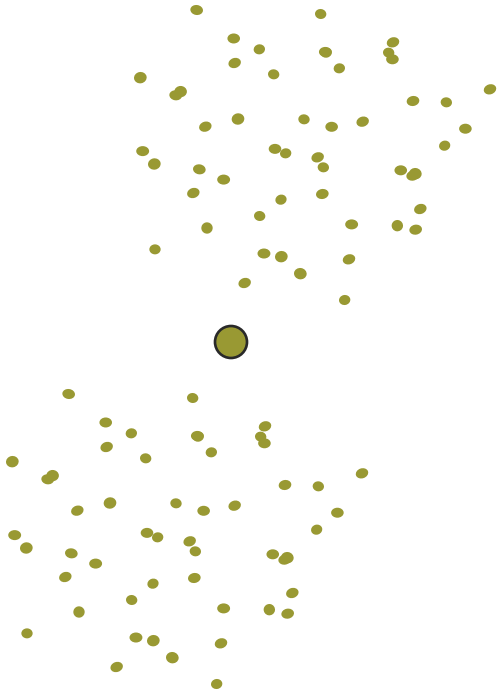


K-Means Getting Stuck



- A local optimum:

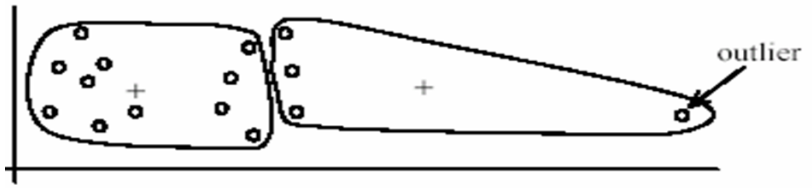




K-means: pros and cons

Pros

- Simple, fast to compute
- Converges to local minimum of within-cluster squared error

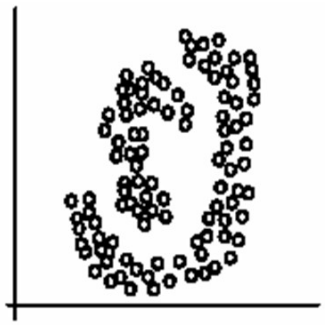


(A): Undesirable clusters

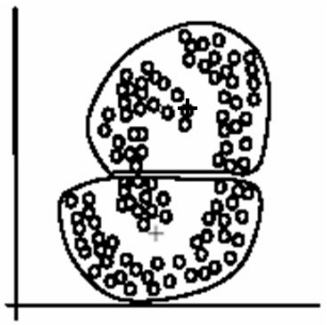
(B): Ideal clusters

Cons/issues

- Setting k?
- Sensitive to initial centers
- Sensitive to outliers
- Detects spherical clusters
- Assuming means can be computed



(A): Two natural clusters



(B): *k*-means clusters

Segmentation as clustering

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

Grouping pixels based on **intensity** similarity

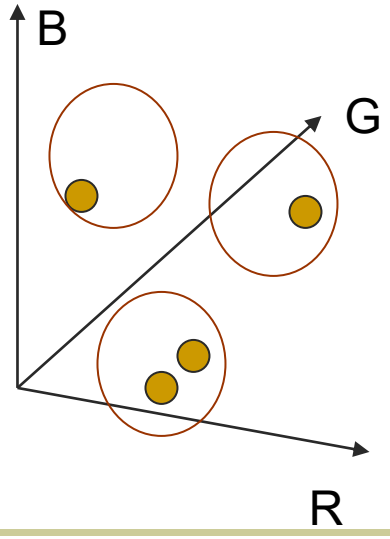
Feature space: intensity value (1-d)

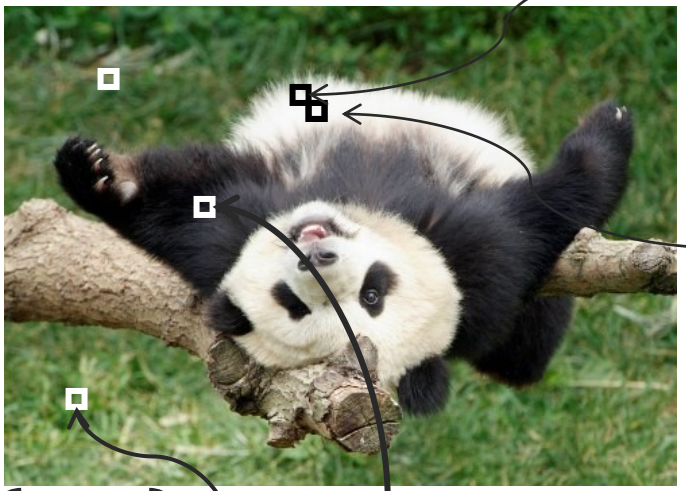
Slide credit: Kristen Grauman

Segmentation as clustering

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-d)

Slide credit: Kristen Grauman

Segmentation as clustering

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

Grouping pixels based on **intensity** similarity

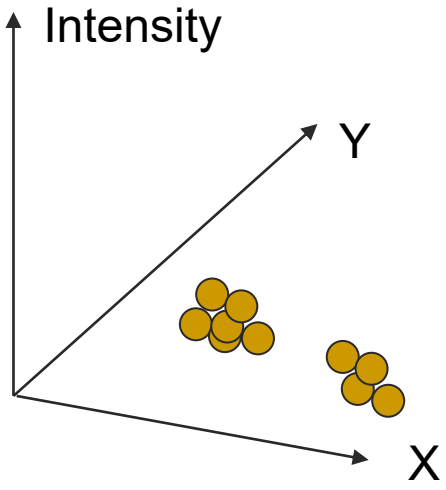
Clusters based on intensity similarity don't have to be spatially coherent.

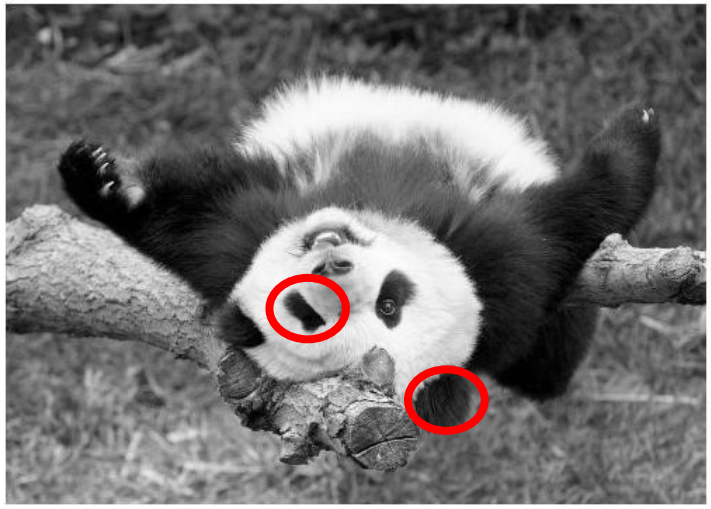
Slide credit: Kristen Grauman

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



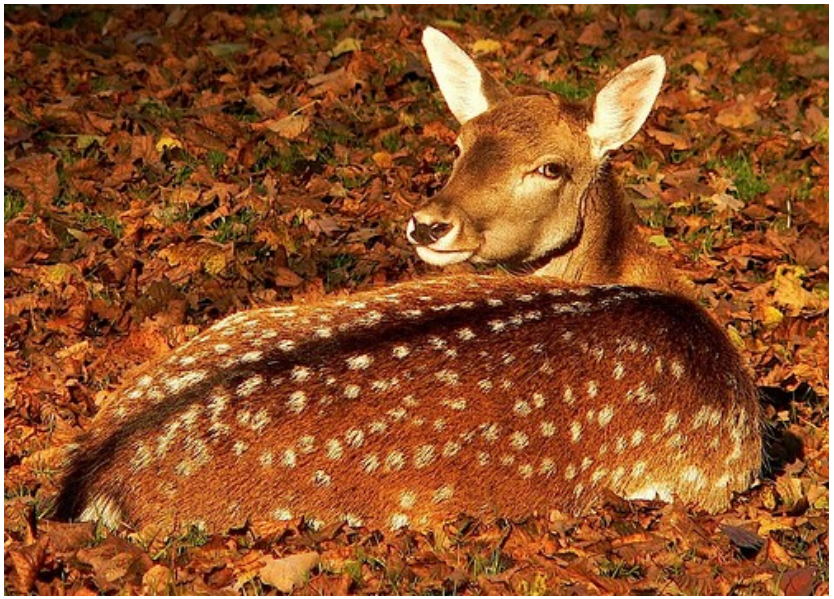


Both regions are black, but if we also include **position (x,y),** then we could group the two into distinct segments; way to encode both similarity & proximity.

Slide credit: Kristen Grauman

Segmentation as clustering

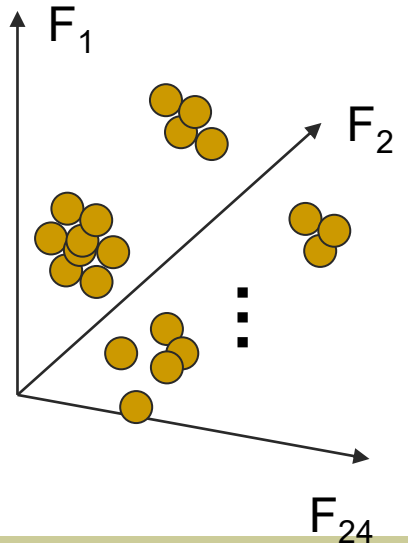
- Color, brightness, position alone are not enough to distinguish all regions…

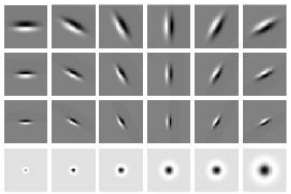


Segmentation as clustering

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

Grouping pixels based on **texture** similarity





Filter bank of 24 filters

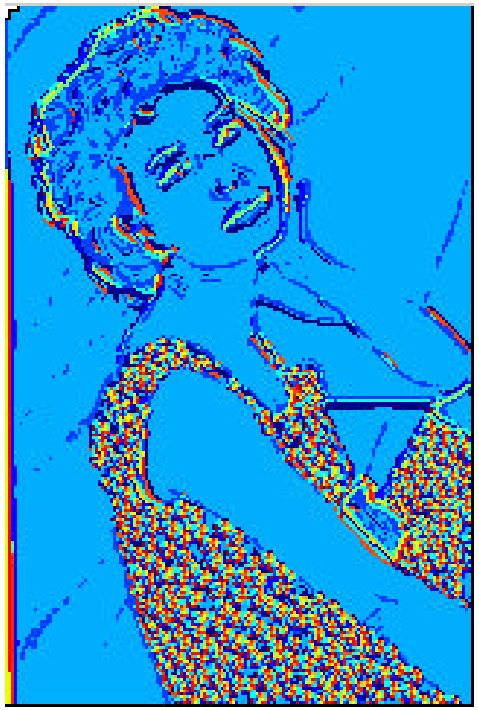
Feature space: filter bank responses (e.g., 24-d)

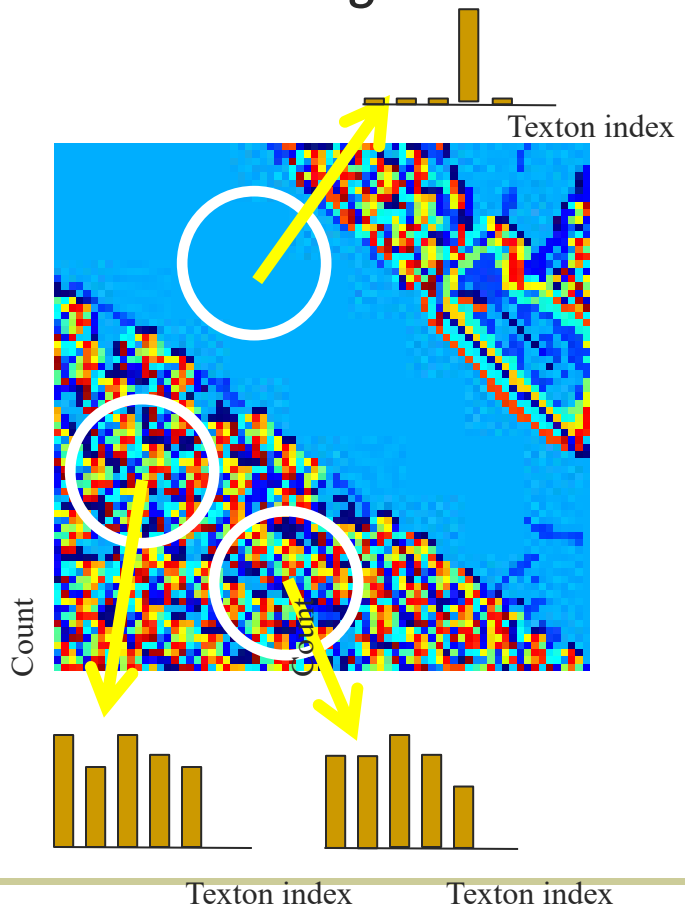
Segmentation with texture features

- Find “textons” by **clustering** vectors of filter bank outputs
- Describe texture in a window based on *texton histogram*

Image

Texton map

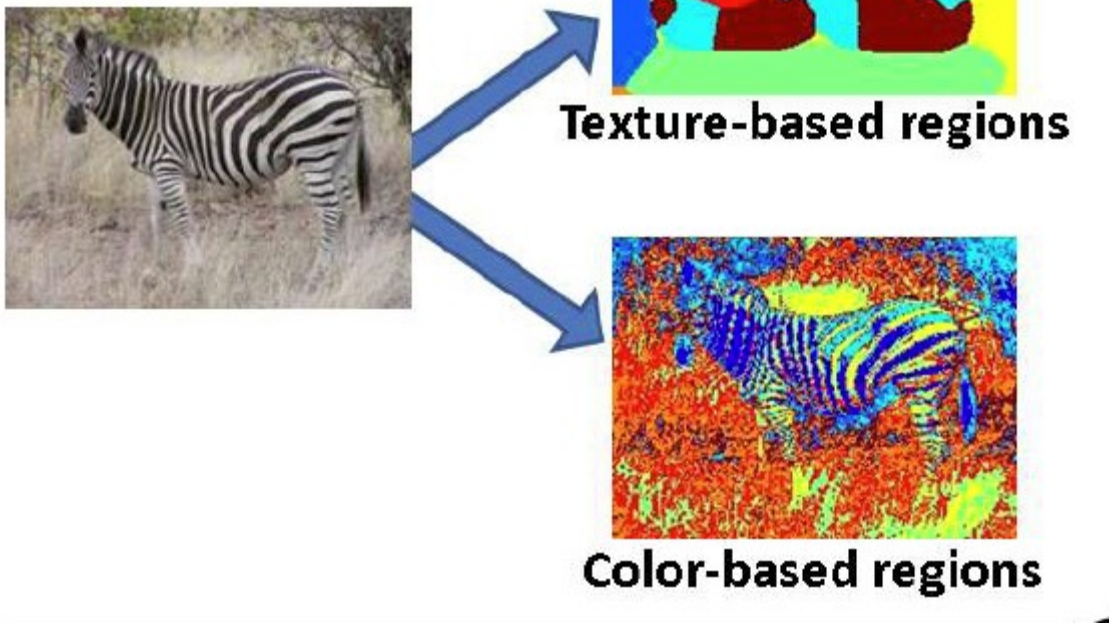




Malik, Belongie, Leung and Shi. IJCV 2001.

Adapted from Lana Lazebnik

Image segmentation example



Slide credit: Kristen Grauman

Probabilistic clustering

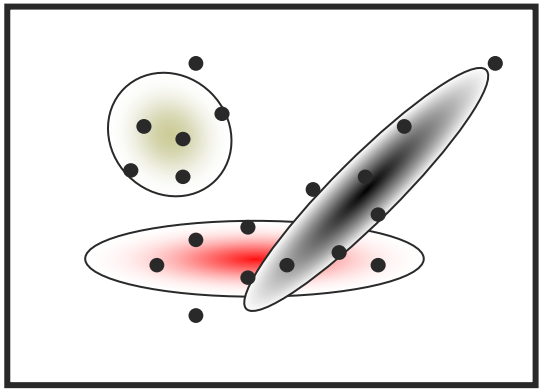
Basic questions

- what's the probability that a point **x** is in cluster m?
- what's the shape of each cluster?

K-means doesn't answer these questions

Probabilistic clustering (basic idea)

- Treat each cluster as a Gaussian density function



Slide credit: Steve Seitz

Mixture of Gaussians

component model parameters

component prior

mixture component

$$p\left(x_n \mid \boldsymbol{\mu}, \boldsymbol{\sigma}^2, \boldsymbol{\pi}\right) = \sum_m p\left(x_n, z_n = m \mid \mu_m, {\sigma_m}^2, \pi_m\right)$$

$$p\left(x_n, z_n = m \mid \boldsymbol{\mu}, \boldsymbol{\sigma}^2, \boldsymbol{\pi}\right) = p\left(x_n, z_n = m \mid \mu_m, {\sigma_m}^2, \pi_m\right)$$

$$= p\left(x_n \mid \mu_m, {\sigma_m}^2\right) p\left(z_n = m \mid \pi_m\right)$$

$$= \frac{1}{\sqrt{2\pi {\sigma_m}^2}} \exp\left(-\frac{\left(x_n - \mu_m\right)^2}{2{\sigma_m}^2}\right) \cdot \pi_m$$

Mixture of Gaussians

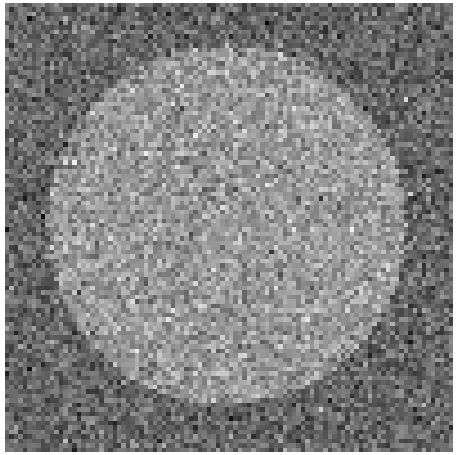
With enough components, can represent any probability density function

- Widely used as general purpose pdf estimator

Segmentation with Mixture of Gaussians

Pixels come from one of several Gaussian components

- We don’t know which pixels come from which components
- We don’t know the parameters for the components



Simple solution

1. Initialize parameters
2. Compute the probability of each hidden variable given the current parameters
3. Compute new parameters for each model, weighted by likelihood of hidden variables
4. Repeat 2-3 until convergence

Mixture of Gaussians: Simple Solution

1. Initialize parameters
2. Compute likelihood of hidden variables for current parameters

$$\alpha_{nm} = p(z_n = m \mid x_n, \boldsymbol{\mu}^{(t)}, \boldsymbol{\sigma}^{2(t)}, \boldsymbol{\pi}^{(t)})$$

3. Estimate new parameters for each model, weighted by likelihood

$$\hat{\mu}_m^{(t+1)} = \frac{1}{\sum_n \alpha_{nm}} \sum_n \alpha_{nm} x_n \qquad \hat{\sigma}_m^{2(t+1)} = \frac{1}{\sum_n \alpha_{nm}} \sum_n \alpha_{nm} \left(x_n - \hat{\mu}_m\right)^2 \qquad \hat{\pi}_m^{(t+1)} = \frac{\sum_n \alpha_{nm}}{N}$$

Expectation Maximization (EM)

$$\text{Goal: } \hat{\theta} = \underset{\theta}{\operatorname{argmax}} \log\left(\sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z} \mid \theta)\right)$$

Log of sums is intractable

Jensen's Inequality

$$f(\mathrm{E}[X]) \geq \mathrm{E}[f(X)]$$

for concave functions f(x)

(so we maximize the lower bound!)

Expectation Maximization (EM) Algorithm

Goal: $\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \log\left(\sum_{\mathbf{z}} p(\mathbf{x},\mathbf{z}\mid\theta)\right)$

1. E-step: compute

$$\mathrm{E}_{z|x,\theta^{(t)}}\left[\log\left(p(\mathbf{x},\mathbf{z}\mid\theta)\right)\right] = \sum_{\mathbf{z}} \log\left(p(\mathbf{x},\mathbf{z}\mid\theta)\right) p\left(\mathbf{z}\mid\mathbf{x},\theta^{(t)}\right)$$

2. M-step: solve

$$\theta^{(t+1)} = \underset{\theta}{\operatorname{argmax}} \sum_{\mathbf{z}} \log\left(p(\mathbf{x},\mathbf{z}\mid\theta)\right) p\left(\mathbf{z}\mid\mathbf{x},\theta^{(t)}\right)$$

Expectation Maximization (EM) Algorithm

Goal: $\hat{\theta} = \arg\max_{\theta} \log\left(\sum_{\mathbf{z}} p(\mathbf{x},\mathbf{z}|\theta)\right) = \log\left(\sum_{\mathbf{z}} q(\mathbf{z}) \frac{p(\mathbf{x},\mathbf{z}|\theta)}{q(\mathbf{z})}\right)$

log of expectation

$$f(\mathrm{E}[X]) \geq \mathrm{E}[f(X)]$$

1. E-step: compute

$$\mathrm{E}_{z|x,\theta^{(t)}}\left[\log\left(p(\mathbf{x},\mathbf{z}\mid\theta)\right)\right] = \sum_{\mathbf{z}} \log\left(p(\mathbf{x},\mathbf{z}\mid\theta)\right) p\left(\mathbf{z}\mid\mathbf{x},\theta^{(t)}\right)$$

expectation of log

2. M-step: solve

$$\theta^{(t+1)} = \arg\max_{\theta} \sum_{\mathbf{z}} \log\left(p(\mathbf{x},\mathbf{z}\mid\theta)\right) p\left(\mathbf{z}\mid\mathbf{x},\theta^{(t)}\right)$$

EM for Mixture of Gaussians

1. E-step: $\mathrm{E}_{z|x,\theta^{(t)}}\left[\log\left(p(\mathbf{x},\mathbf{z}\mid\theta)\right)\right]=\sum_{\mathbf{z}}\log\left(p(\mathbf{x},\mathbf{z}\mid\theta)\right)p\left(\mathbf{z}\mid\mathbf{x},\theta^{(t)}\right)$

2. M-step: $\theta^{(t+1)}=\underset{\theta}{\operatorname{argmax}}\sum_{\mathbf{z}}\log\left(p(\mathbf{x},\mathbf{z}\mid\theta)\right)p\left(\mathbf{z}\mid\mathbf{x},\theta^{(t)}\right)$

$$p\left(x_n\mid\boldsymbol{\mu},\boldsymbol{\sigma}^2,\boldsymbol{\pi}\right)=\sum_m p\left(x_n,z_n=m\mid\mu_m,{\sigma_m}^2,\pi_m\right)=\sum_m\frac{1}{\sqrt{2\pi{\sigma_m}^2}}\exp\left(-\frac{\left(x_n-\mu_m\right)^2}{{\sigma_m}^2}\right)\cdot\pi_m$$

$$\alpha_{nm}=p(z_n=m\mid x_n,\boldsymbol{\mu}^{(t)},\boldsymbol{\sigma}^{2(t)},\boldsymbol{\pi}^{(t)})$$

$$\hat{\mu}_m^{(t+1)}=\frac{1}{\sum_n\alpha_{nm}}\sum_n\alpha_{nm}x_n \qquad \hat{\sigma}_m^{2(t+1)}=\frac{1}{\sum_n\alpha_{nm}}\sum_n\alpha_{nm}\left(x_n-\hat{\mu}_m\right)^2 \qquad \hat{\pi}_m^{(t+1)}=\frac{\sum_n\alpha_{nm}}{N}$$

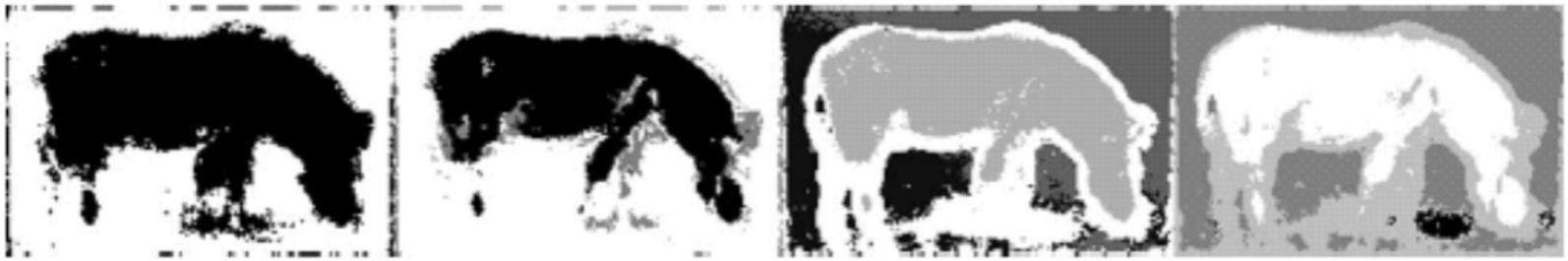
Application of EM

- **Turns out this is useful for all sorts of problems**
  - Any clustering problem
  - Any model estimation problem
  - Missing data problems
  - Finding outliers
  - Segmentation problems
    - Segmentation based on color
    - Segmentation based on motion
    - Foreground/background separation
  - …

Segmentation with EM

Original image

EM segmentation results



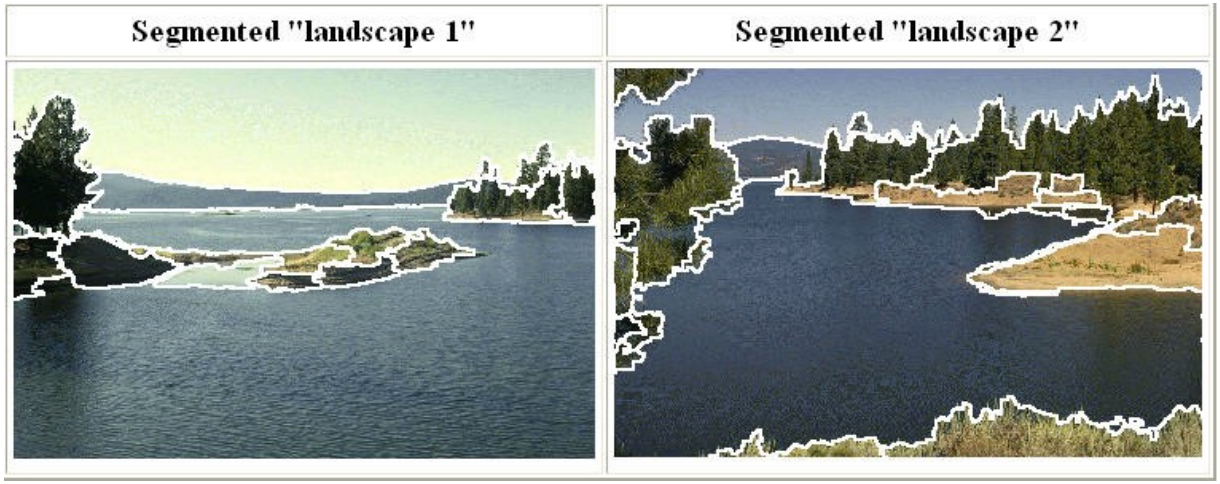
k=2 k=3 k=4 k=5

Summary of GMM

- **Pros**
  - Probabilistic interpretation
  - Soft assignments between data points and clusters
  - Generative model, can predict novel data points
  - Relatively compact storage
- **Cons**
  - Local minima
    - k-means is NP-hard even with k=2
  - Initialization
    - Often a good idea to start with some k-means iterations.
  - Need to know number of components
    - Solutions: model selection (AIC, BIC), Dirichlet process mixture
  - Need to choose generative model
  - Numerical problems are often a nuisance

Mean-Shift Segmentation

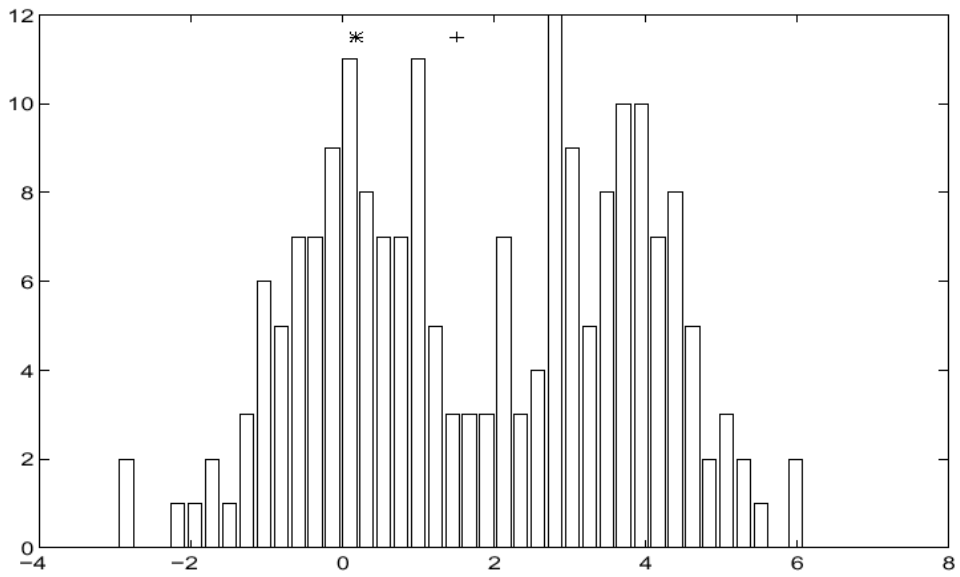
- An advanced and versatile technique for clustering-based segmentation



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.

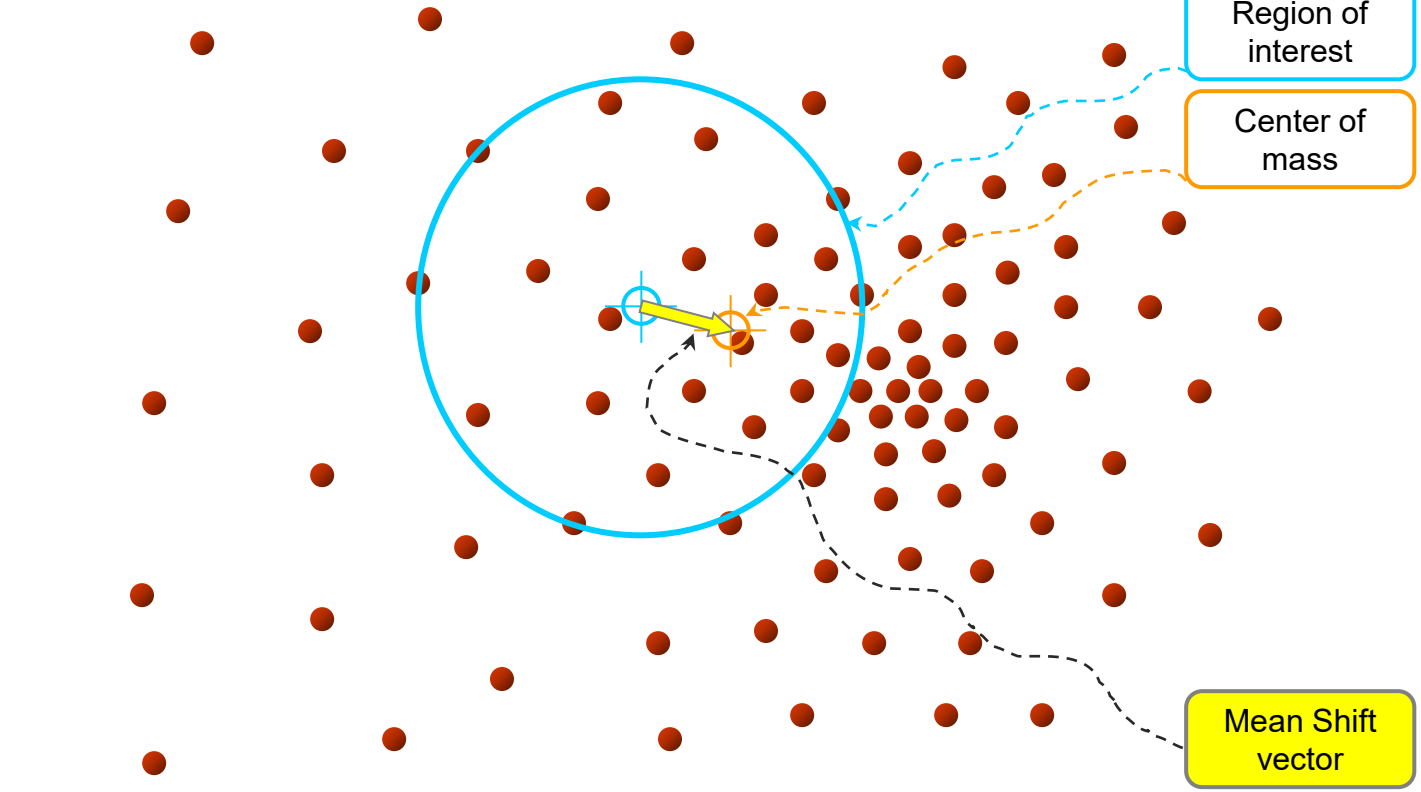
Mean-Shift Algorithm



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

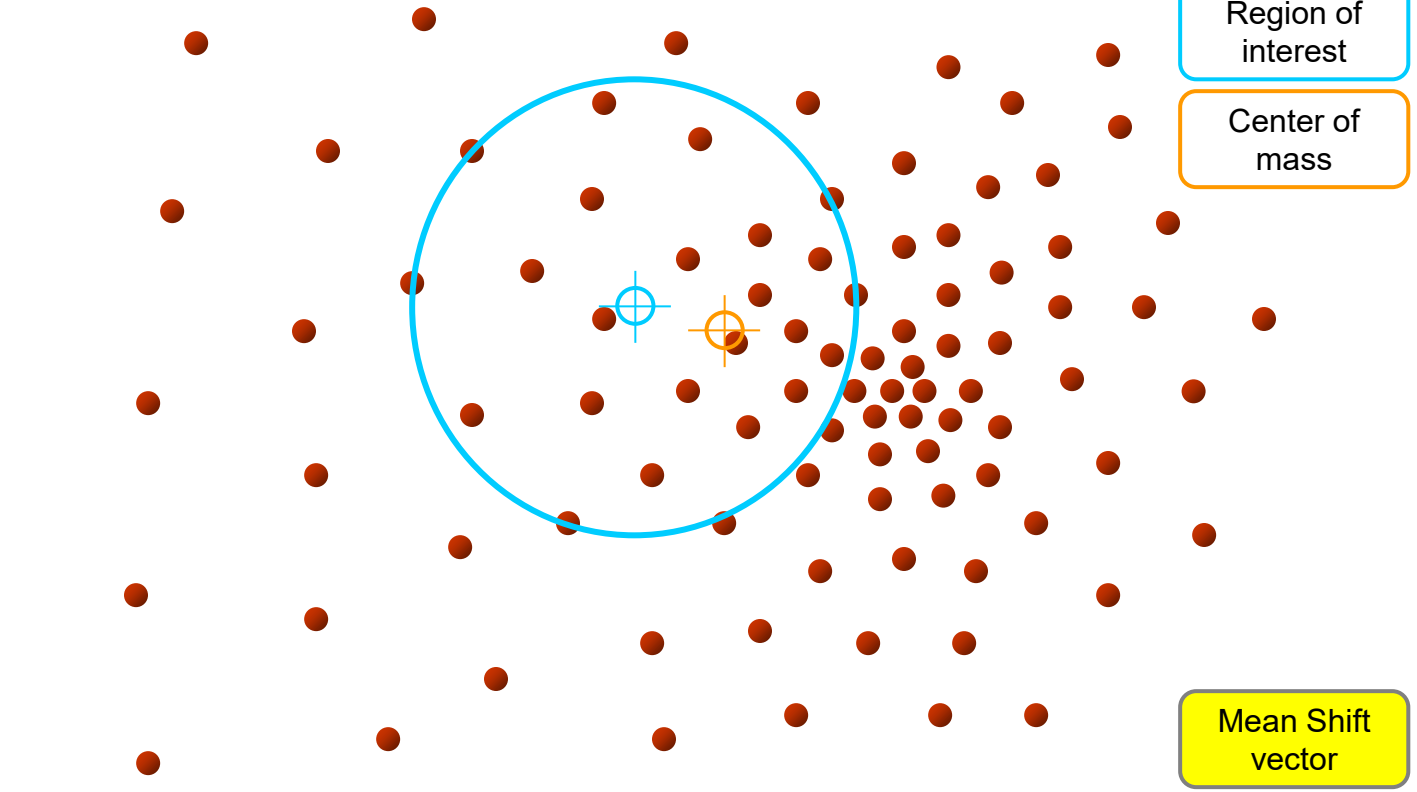
$$\sum_{x \in W} xH(x)$$

Mean-Shift



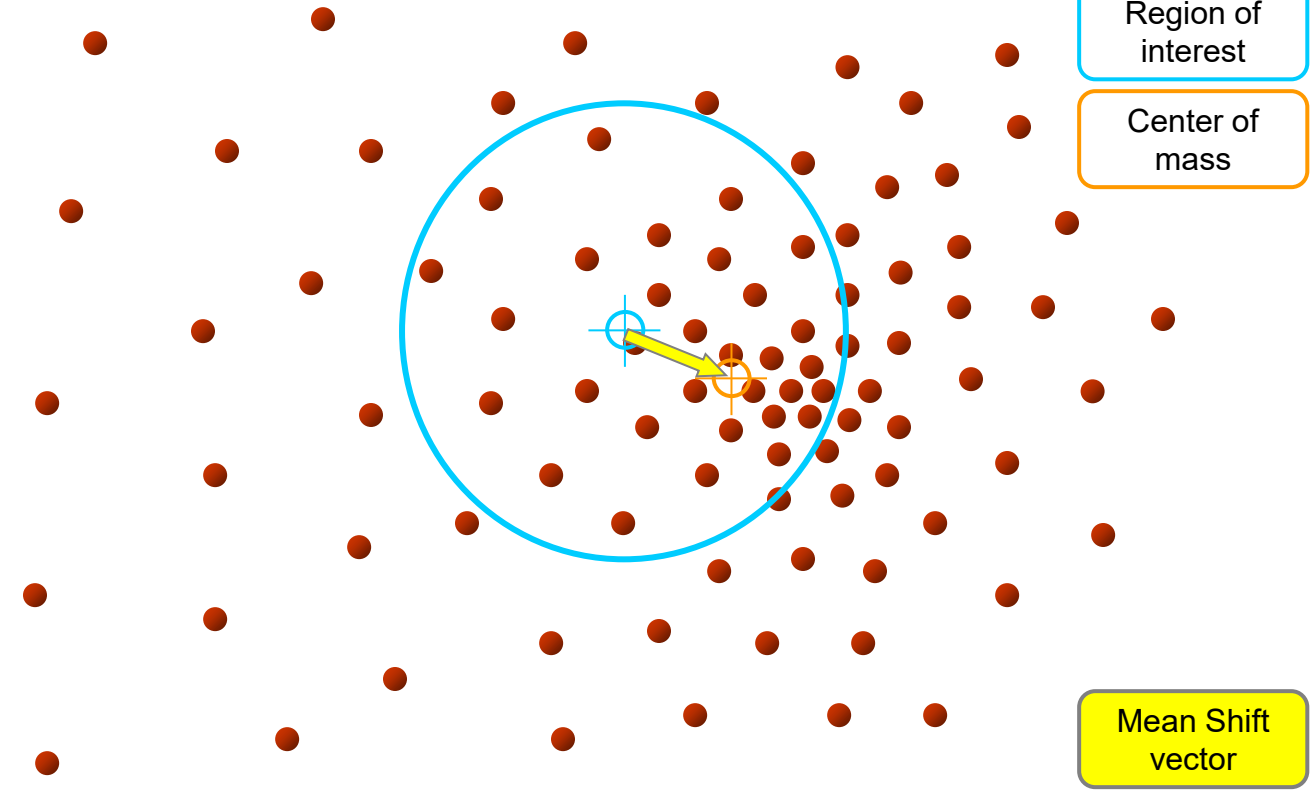
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



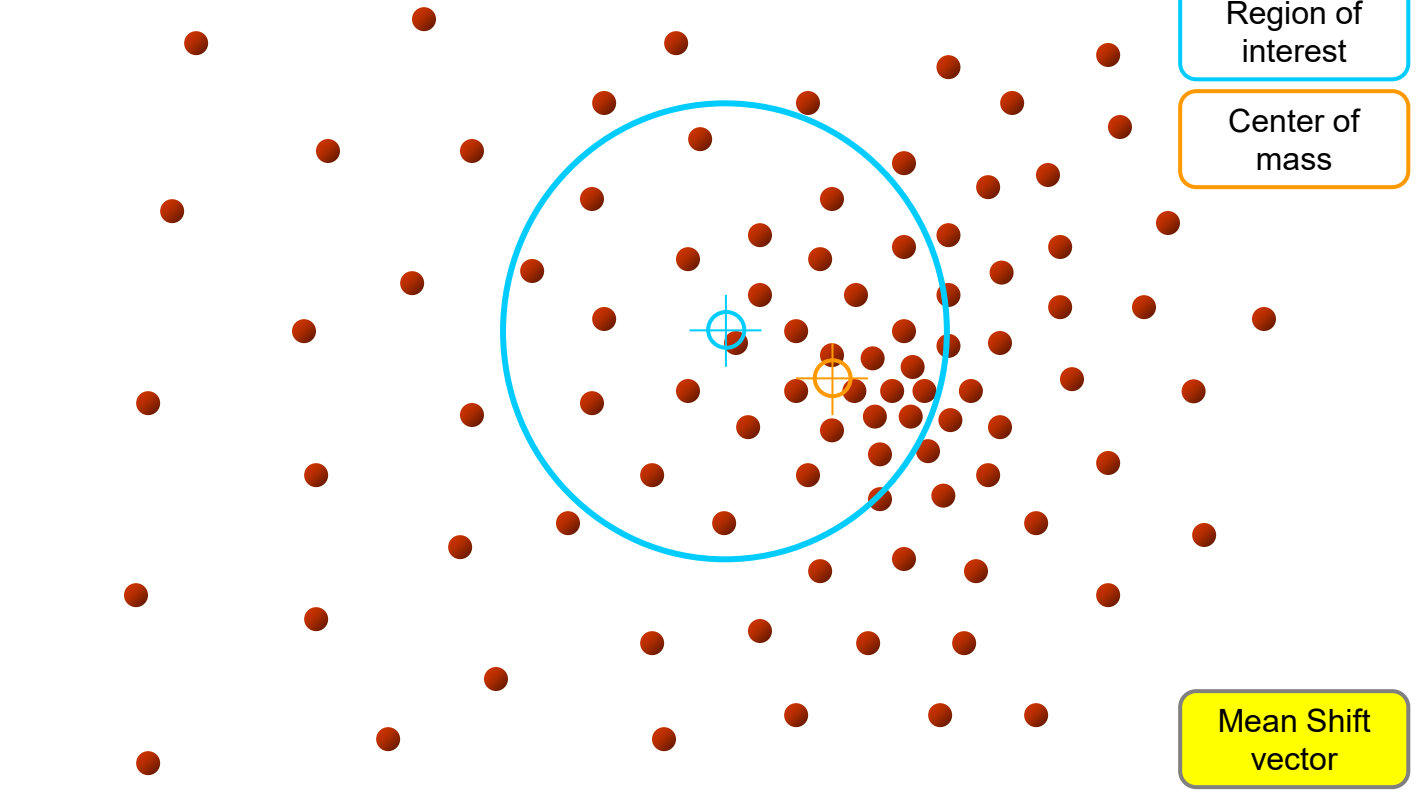
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



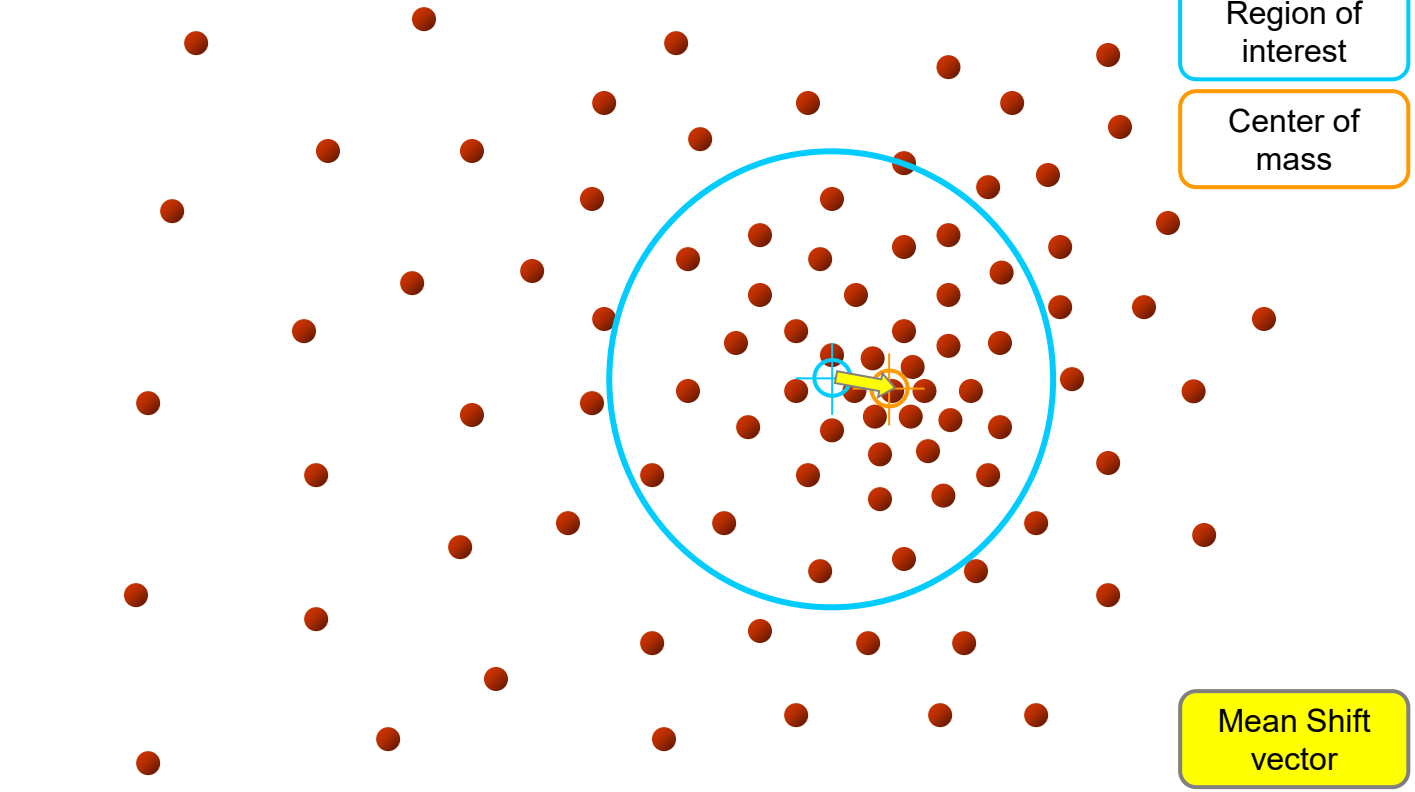
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



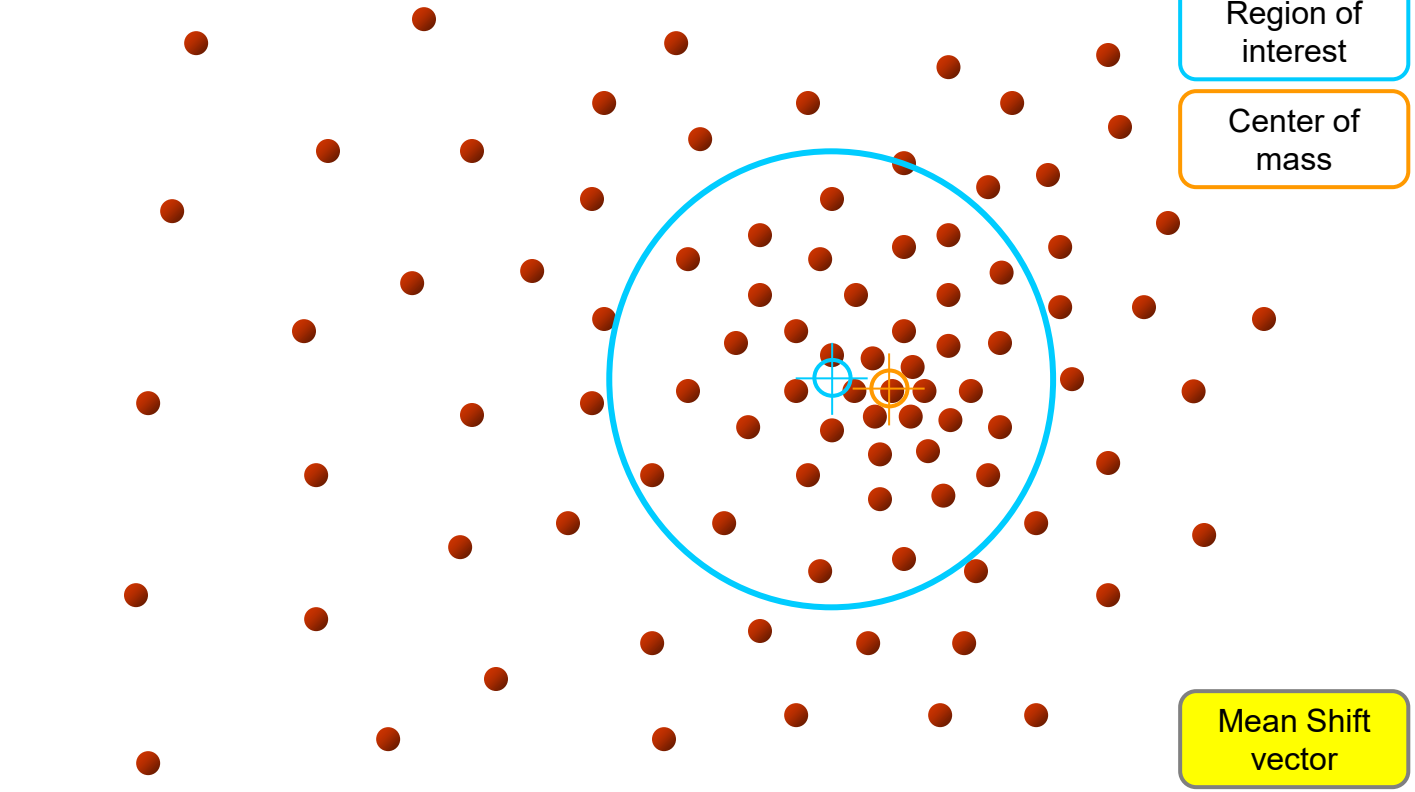
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



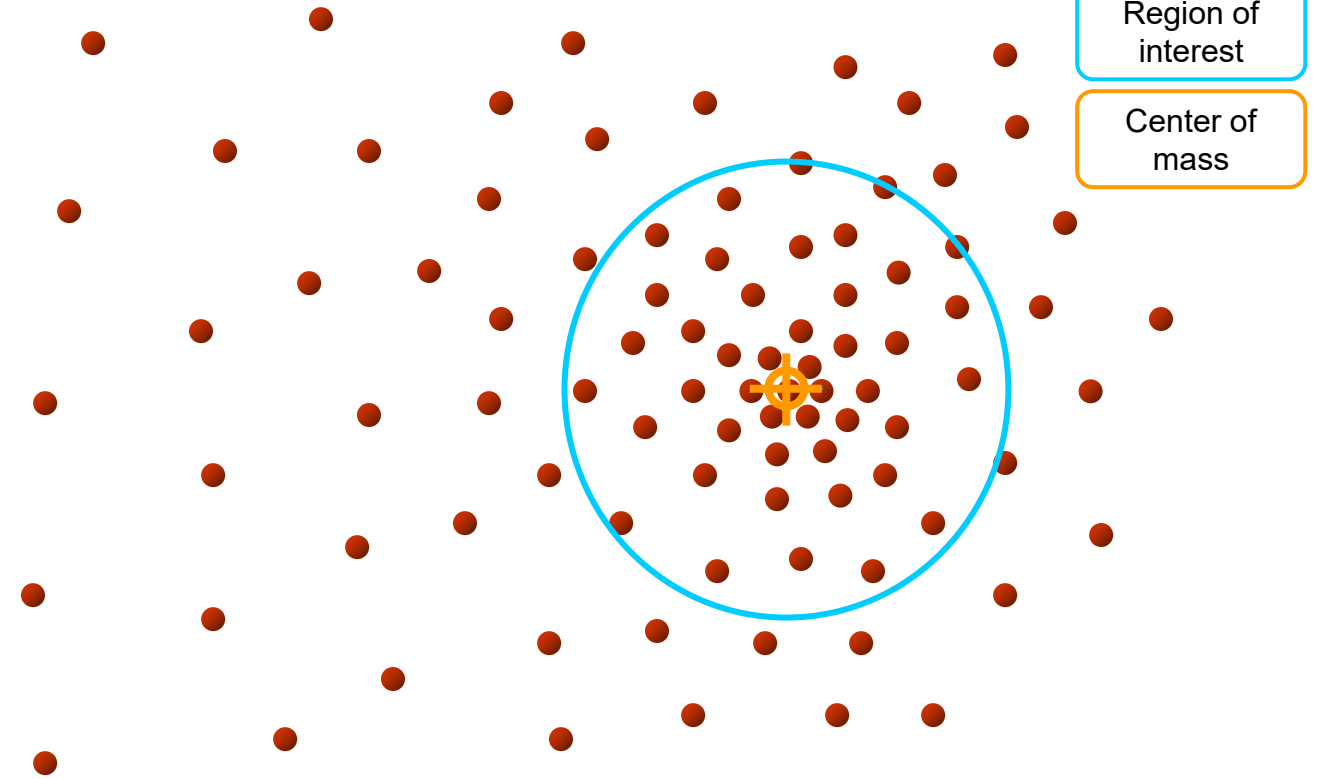
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



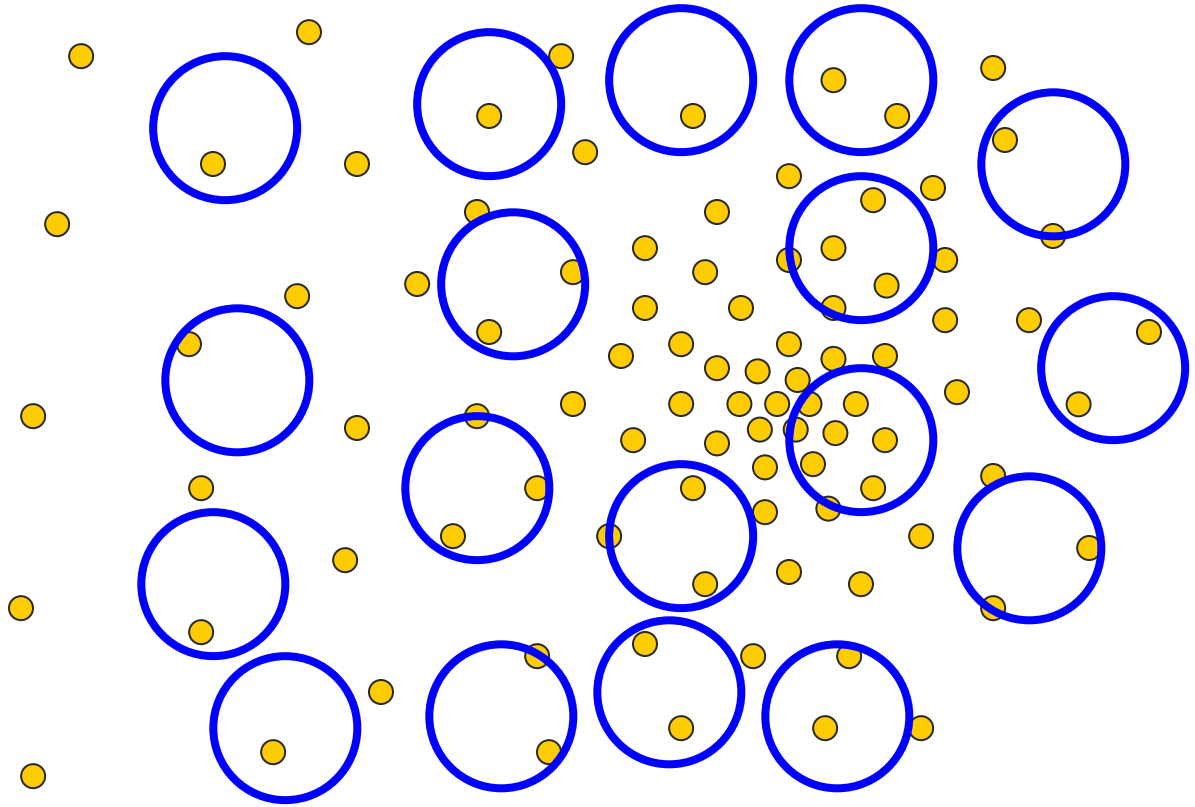
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



Slide by Y. Ukrainitz & B. Sarel

Real Modality Analysis

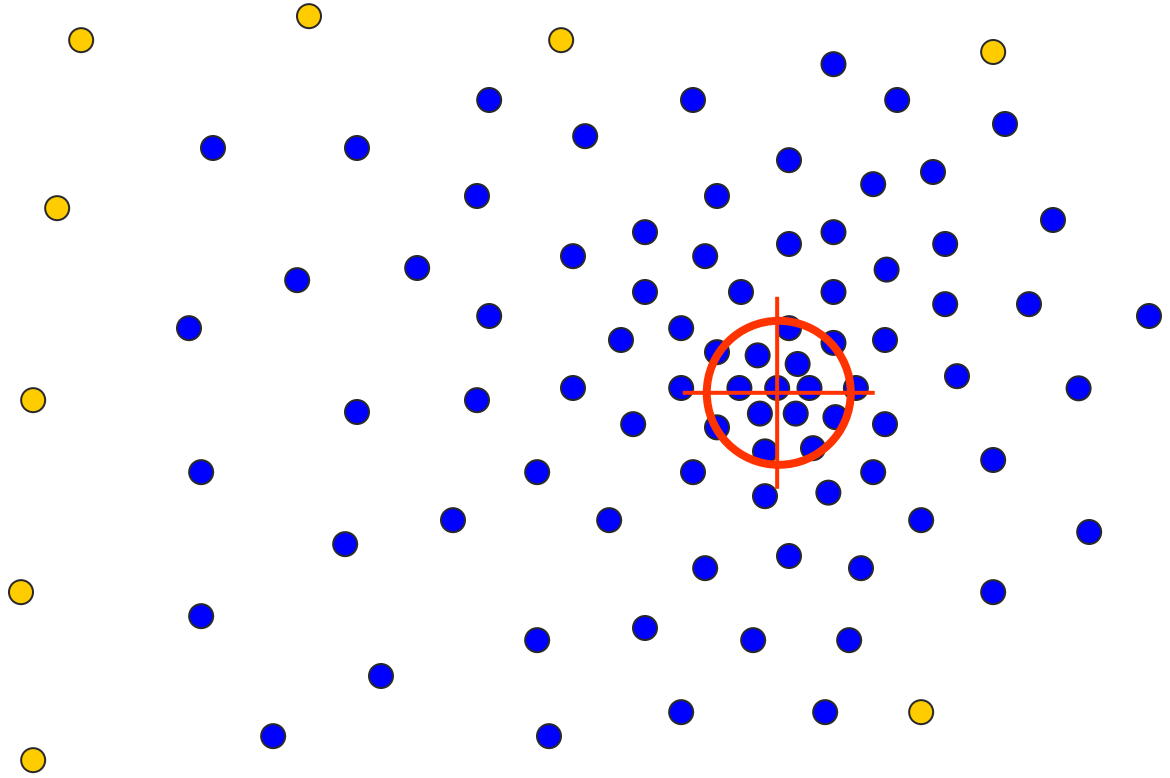


Tessellate the space with windows

Run the procedure in parallel

Slide by Y. Ukrainitz & B. Sarel

Real Modality Analysis

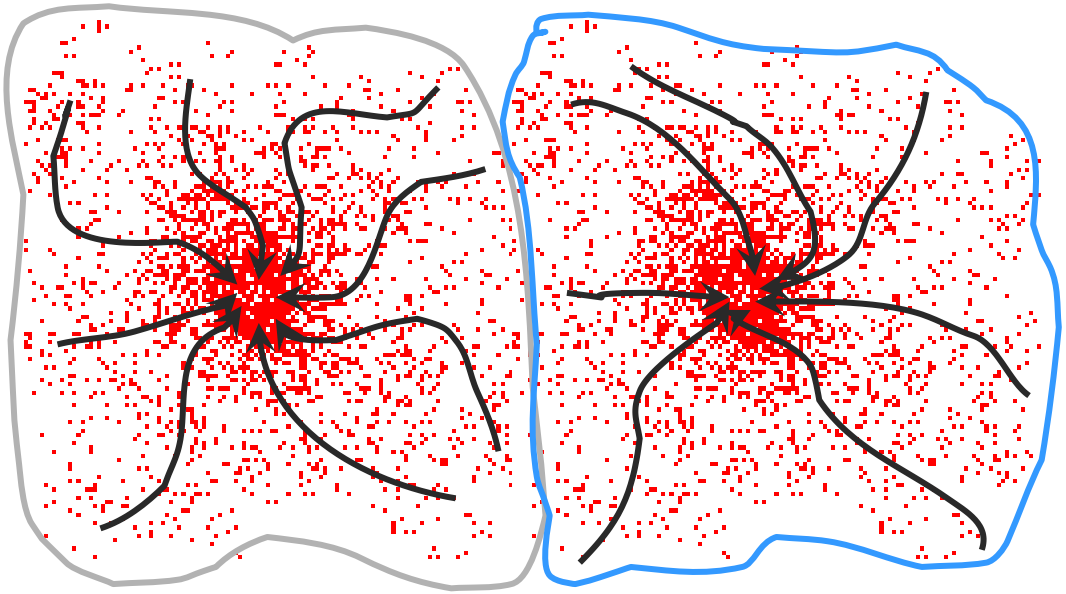


The blue data points were traversed by the windows towards the mode.

Slide by Y. Ukrainitz & B. Sarel

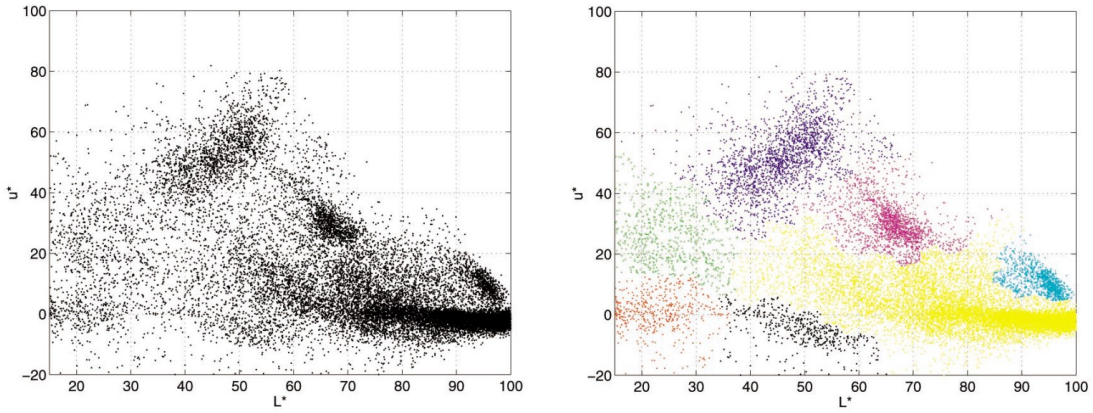
Attraction basin

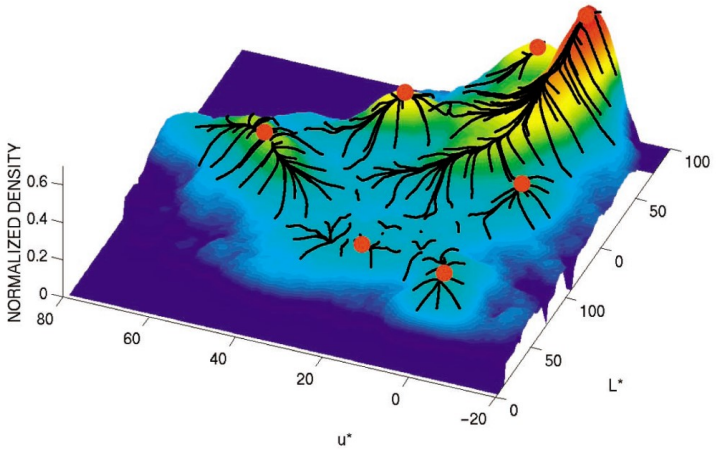
- Attraction basin: the region for which all trajectories lead to the same mode
- Cluster: all data points in the attraction basin of a mode



Mean shift algorithm

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





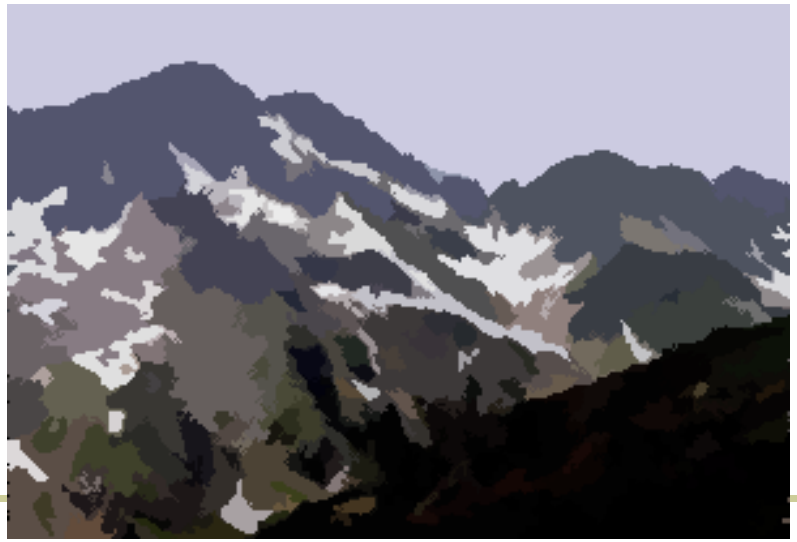
Mean shift clustering

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

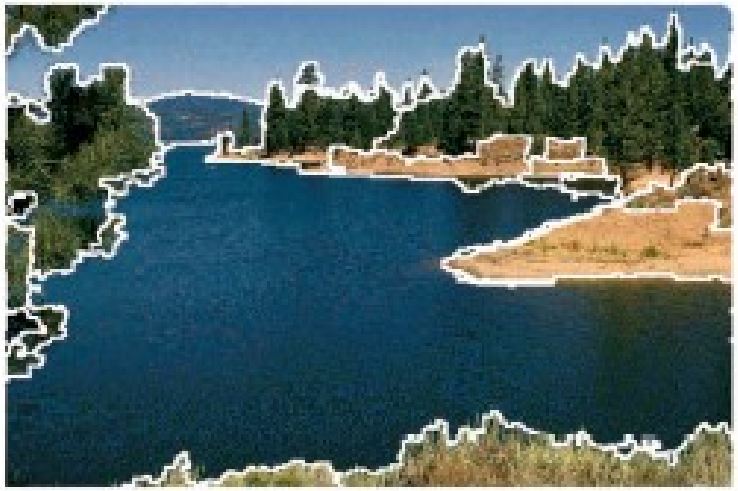
Segmentation by Mean Shift

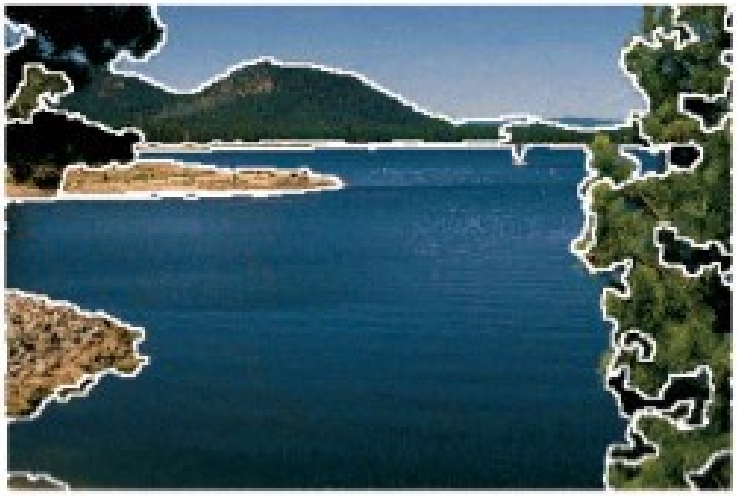
- Compute features for each pixel (color, gradients, texture, etc); also store each pixel's position
- Set kernel size for features K_f and position K_s
- Initialize windows at individual pixel locations
- Perform mean shift for each window until convergence
- Merge modes that are within width of K_f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

Mean shift segmentation results



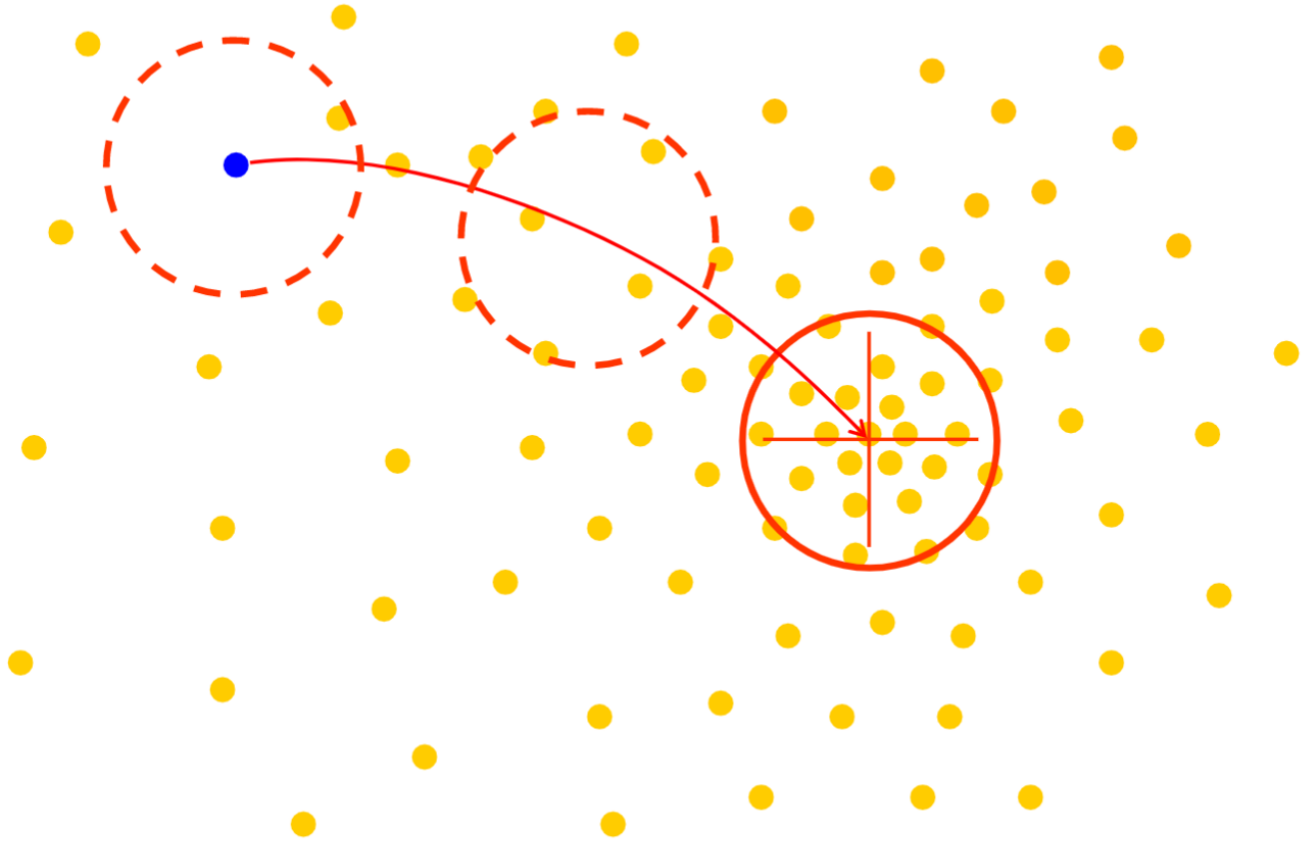
http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html





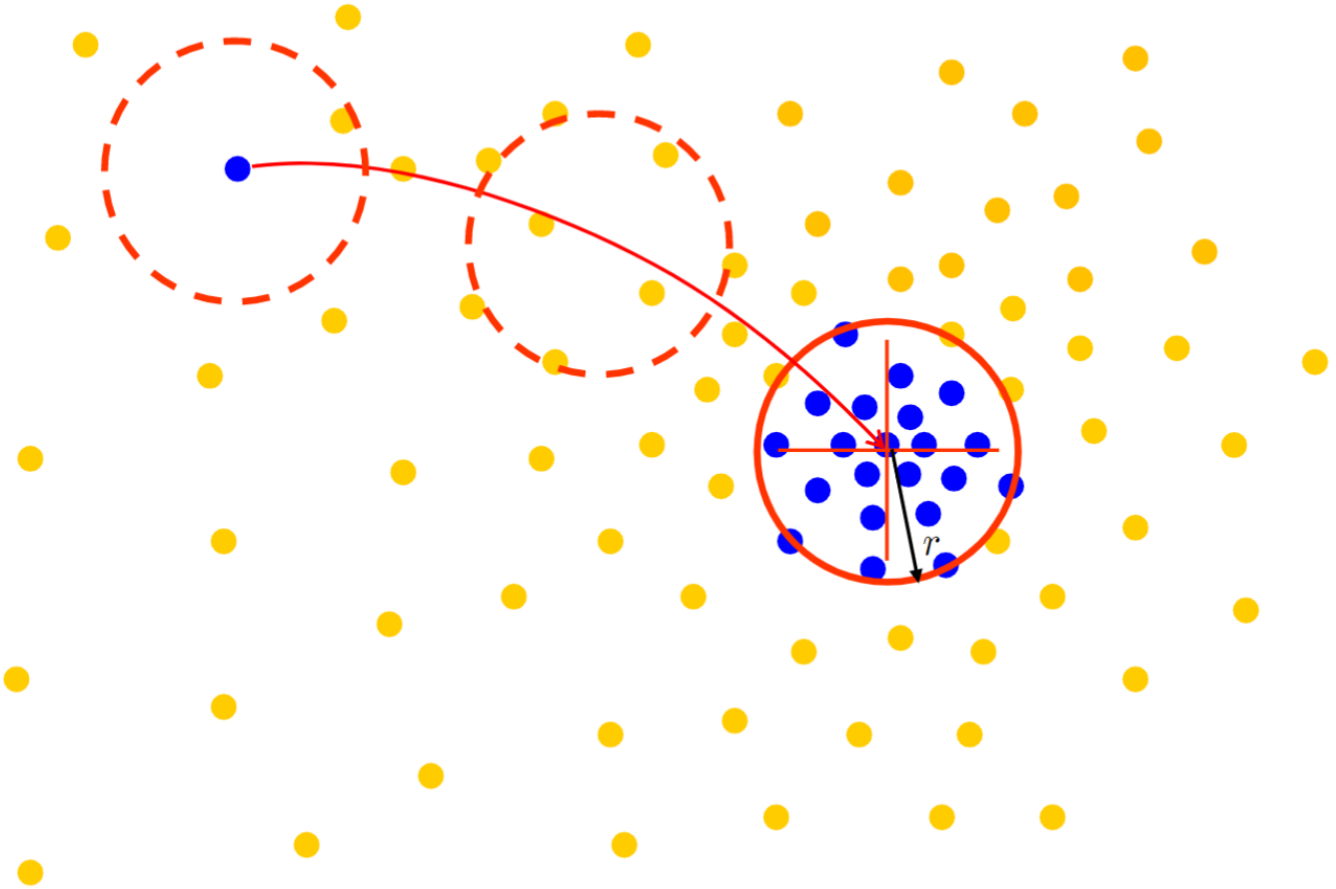
http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

Problem: Computational Complexity



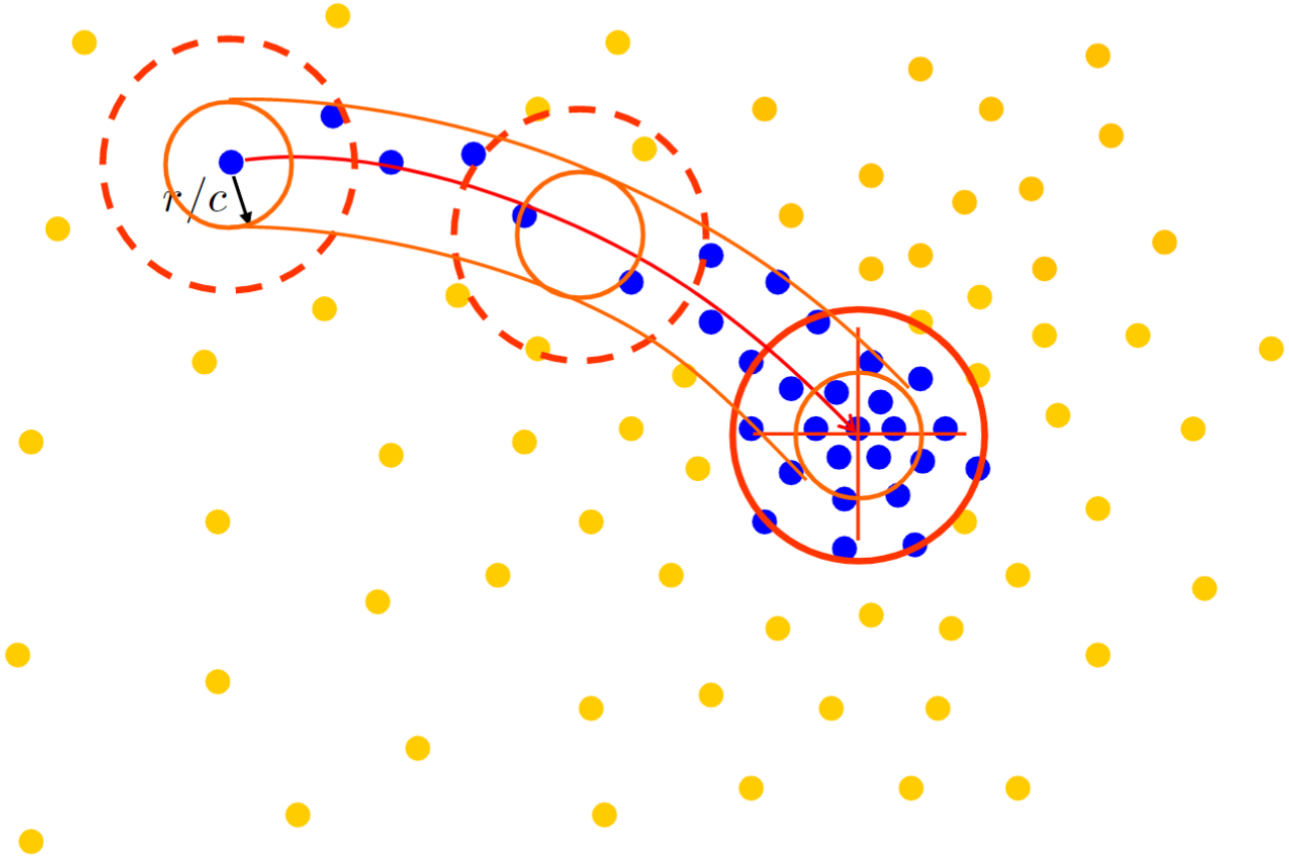
- Need to shift many windows…
- Many computations will be redundant.

Speed Up



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

Speed Up



2. Assign all points within radius r/c of the search path to the mode.

Mean-shift: other issues

- Speedups
  - Binned estimation – replace points within some “bin” by point at center with mass
  - Fast search of neighbors – e.g., k-d tree or approximate NN
  - Update all windows in each iteration (faster convergence)
- Other tricks
  - Use kNN to determine window sizes adaptively
- Lots of theoretical support

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

Mean shift pros and cons

- Pros
  - Good general-purpose segmentation
  - Flexible in number and shape of regions
  - Robust to outliers
  - General mode-finding algorithm (useful for other problems such as finding most common surface normals)
- Cons
  - Have to choose kernel size in advance
  - Not suitable for high-dimensional features
- When to use it
  - Oversegmentation
  - Multiple segmentations
  - Tracking, clustering, filtering applications
    - D. Comaniciu, V. Ramesh, P. Meer: *Real-Time Tracking of Non-Rigid Objects using Mean Shift*, *Best Paper Award*, IEEE Conf. Computer Vision and Pattern Recognition (CVPR'00), Hilton Head Island, South Carolina, Vol. 2, 142-149, 2000

Mean-shift reading

- Nicely written mean-shift explanation (with math)

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

  - Includes .m code for mean-shift clustering

- Mean-shift paper by Comaniciu and Meer

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

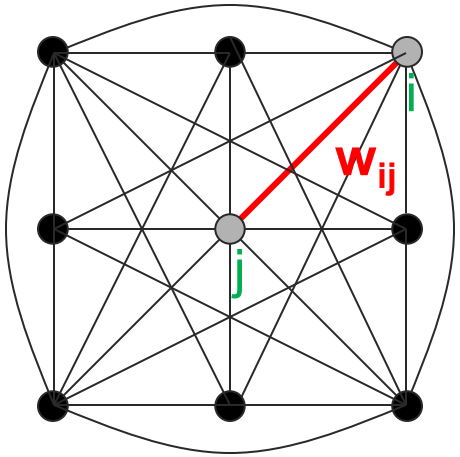
- Adaptive mean shift in higher dimensions

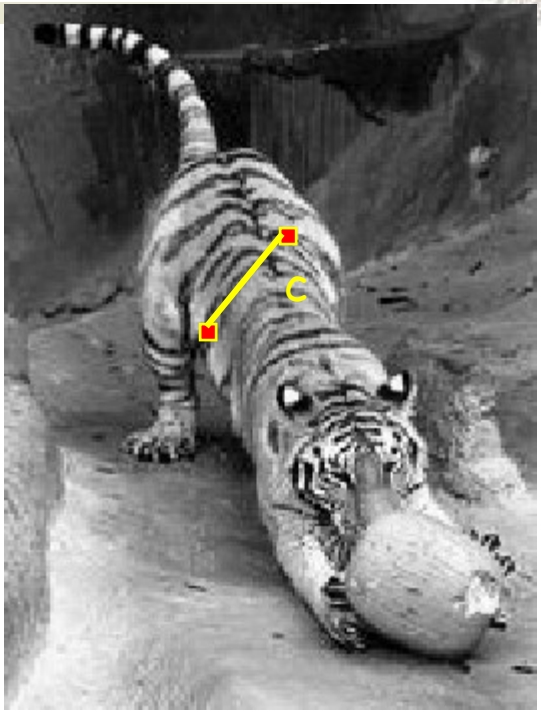
http://mis.hevra.haifa.ac.il/~ishimshoni/papers/chap9.pdf

Today's Class

- What are grouping problems in vision?
- Inspiration from human perception
  - Gestalt properties
- Bottom-up segmentation via clustering
  - Mode finding and mean shift: k-means, GMM, mean-shift
- **Graph-based segmentation: Normalized Cut**
- Oversegmentation
  - Watershed algorithm, Felzenszwalb and Huttenlocher graph-based
- Multiple segmentation

Images as graphs





- *Fully-connected* graph
  - node for every pixel
  - link between *every* pair of pixels, **i**,**j**
  - similarity $\mathbf{w_{ij}}$ for each link

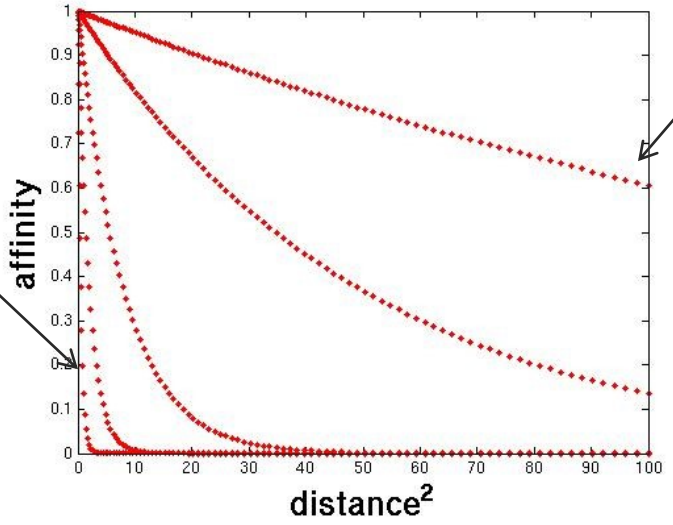
Source: Seitz

Measuring affinity

- One possibility:

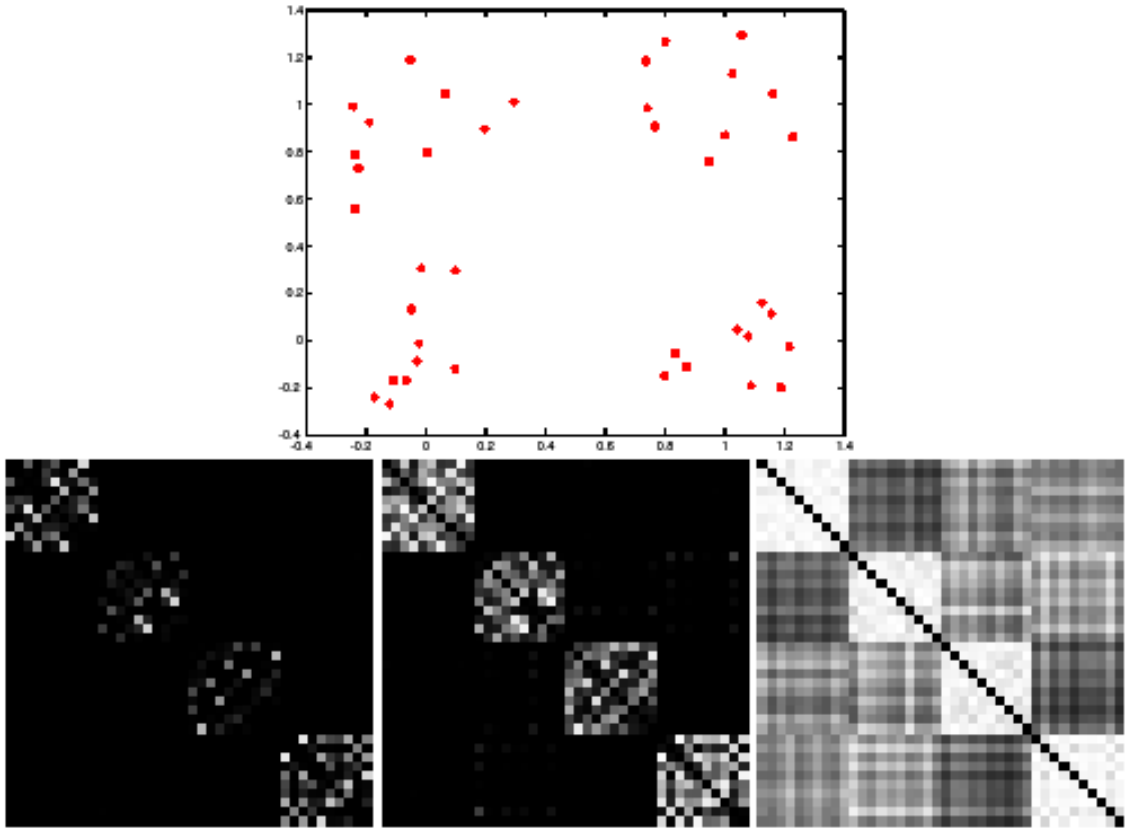
$$aff(x,y) = \exp\left\{-\left(1/2\sigma_d^2\right)\left(\|x-y\|^2\right)\right\}$$

Small sigma: group only nearby points



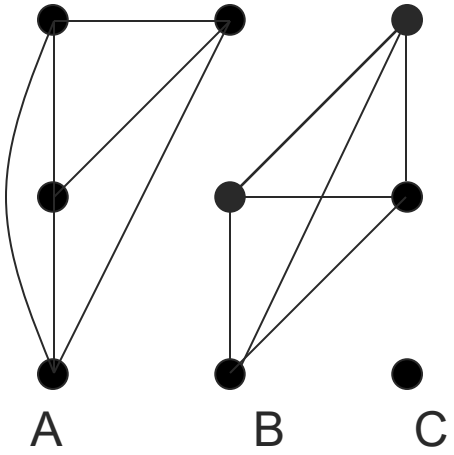
Large sigma: group distant points

Similarity matrix



Increasing sigma

Segmentation by Graph Cuts



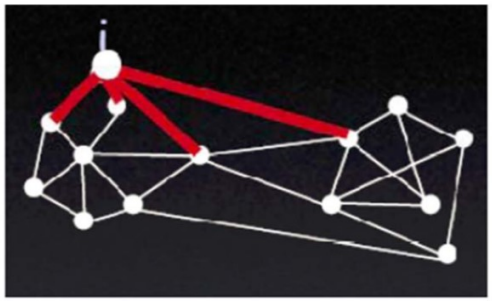
- Break Graph into Segments
  - Delete links that cross between segments
  - Easiest to break links that have low cost (low similarity)
    - similar pixels should be in the same segments
    - dissimilar pixels should be in different segments

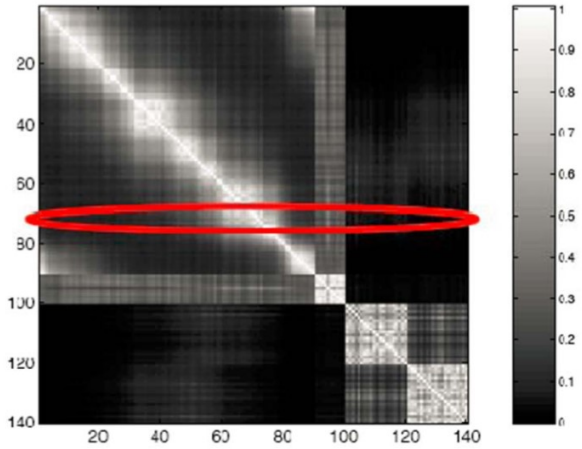
Source: Seitz

Graph Terminologies

- Degree of nodes

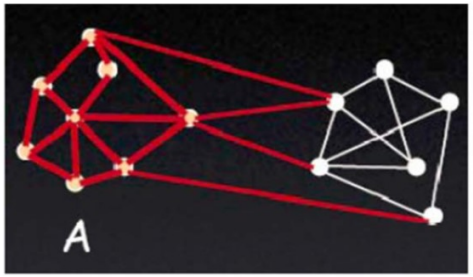
$$d_i = \sum_j w_{i,j}$$

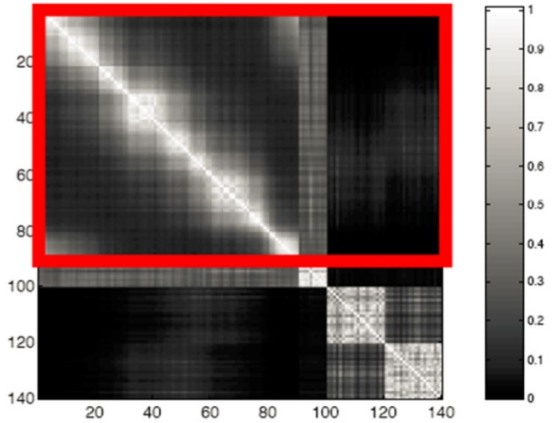




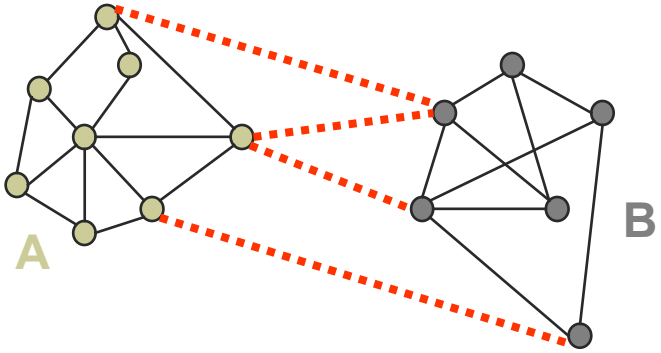
- Volume of a set

$$vol(A) = \sum_{i \in A} d_i, A \subseteq V$$





Cuts in a graph



- Link Cut
  - set of links whose removal makes a graph disconnected
  - cost of a cut:

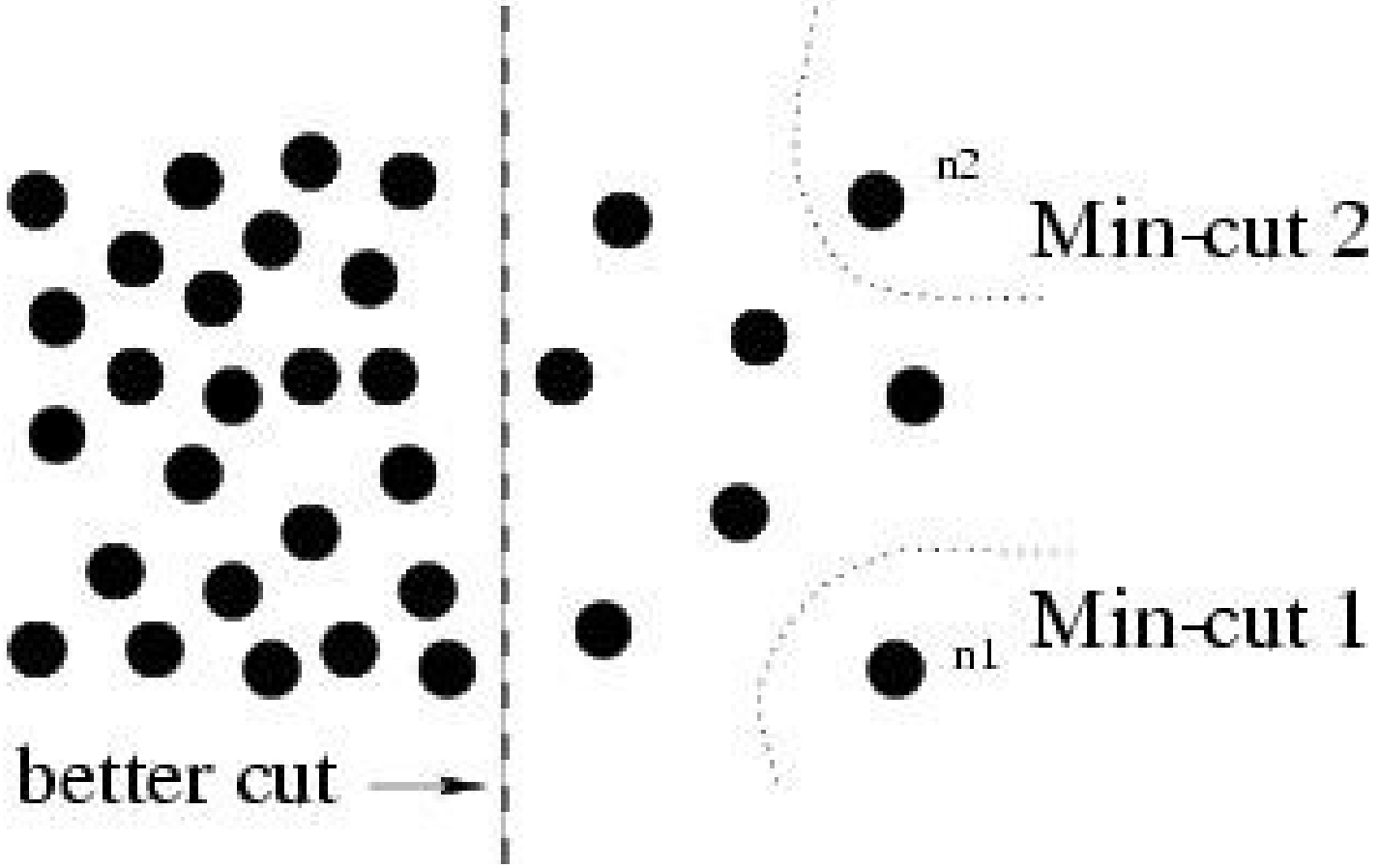
$$cut(A, B) = \sum_{p \in A, q \in B} c_{p,q}$$

One idea: Find minimum cut

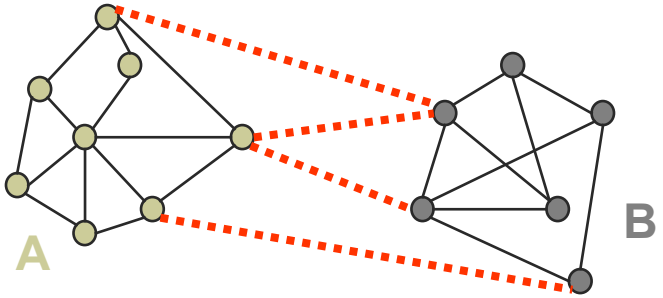
- gives you a segmentation
- fast algorithms exist for doing this

Source: Seitz

But min cut is not always the best cut...



Normalized Cuts in a graph



Normalized Cut

- a cut penalizes large segments
- fix by normalizing for size of segments

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

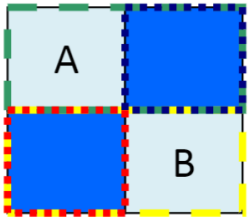
- volume(A) = sum of costs of all edges that touch A

Source: Seitz

Normalized Cut

- Consider the connectivity between groups relative to the volume of each group

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



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

Minimized when Vol(A) and Vol(B) are equal.
Thus encourage balanced cut

Solving the Normalized Cut problem

- Exact discrete solution to Ncut is NP-hard even on regular grid [Papadimitriou’ 97]
- We first transform to

$$min_{\boldsymbol{x}} Ncut(\boldsymbol{x}) = min_{\boldsymbol{y}} \frac{\boldsymbol{y}^T(\mathbf{D}-\mathbf{W})\boldsymbol{y}}{\boldsymbol{y}^T\mathbf{D}\boldsymbol{y}}$$

$$\text{with the condition } y(i) \in \{1, -b\} \text{ and } \boldsymbol{y}^T\mathbf{D}\mathbf{1} = 0$$

- Drawing on spectral graph theory, good approximation can be obtained by solving a generalized eigenvalue problem.

$$(\mathbf{D}-\mathbf{W})\boldsymbol{y} = \lambda\mathbf{D}\boldsymbol{y}.$$

Recursive normalized cuts

1. Given an image or image sequence, set up a weighted graph: G=(V, E)
   1. Vertex for each pixel
   2. Edge weight for nearby pairs of pixels

2. Solve for eigenvectors with the smallest eigenvalues: $(D - W)y = \lambda Dy$
   1. Use the eigenvector with the second smallest eigenvalue to bipartition the graph
   2. Note: this is an approximation

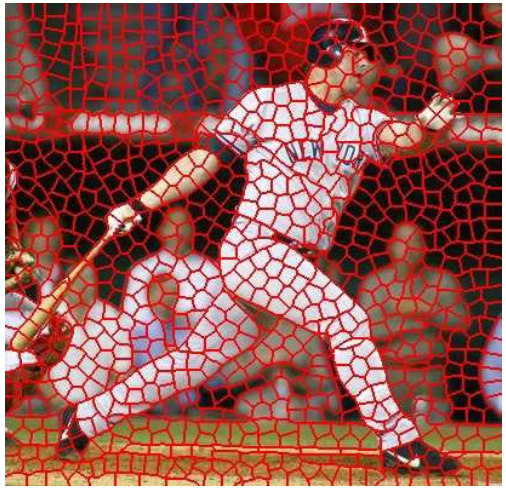
3. Recursively repartition the segmented parts if necessary

Details: http://www.cs.berkeley.edu/~malik/papers/SM-n**cut**.pdf

Normalized cuts results

Normalized cuts: Pro and con

- Pros
  - Generic framework, can be used with many different features and affinity formulations
  - Provides regular segments
- Cons
  - Need to chose number of segments
  - High storage requirement and time complexity
  - Bias towards partitioning into equal segments
- Usage
  - Use for oversegmentation when you want regular segments



Today's Class

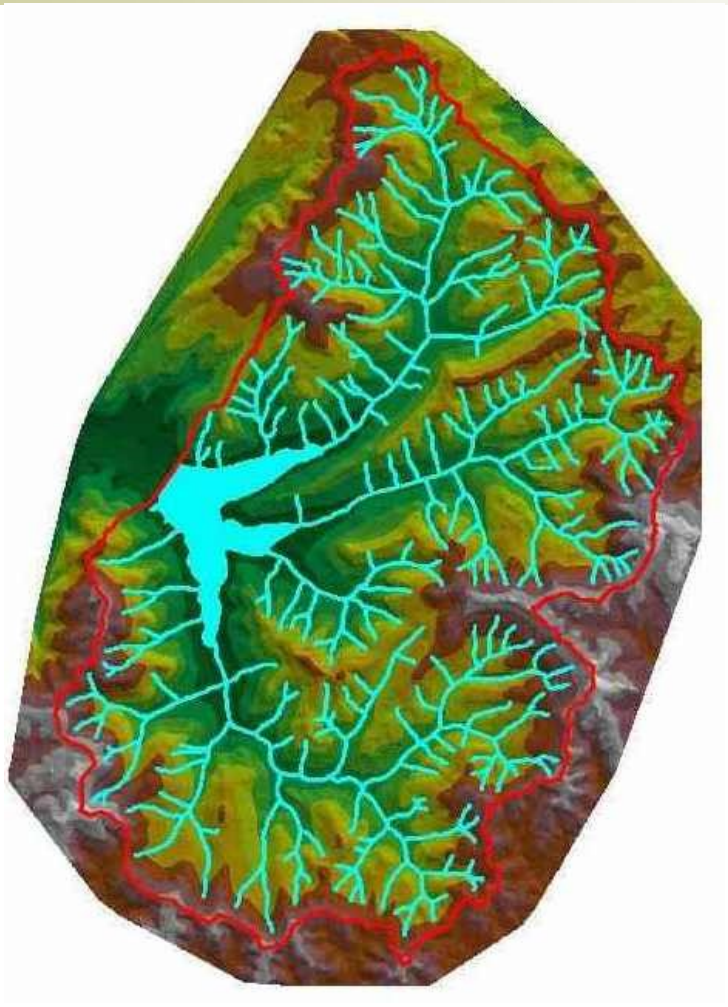
- What are grouping problems in vision?
- Inspiration from human perception
  - Gestalt properties
- Bottom-up segmentation via clustering
  - Mode finding and mean shift: k-means, GMM, mean-shift
- Graph-based segmentation: Normalized Cut
- **Oversegmentation**
  - Watershed algorithm, Felzenszwalb and Huttenlocher graph-based etc.
- Multiple segmentation

Superpixel algorithms

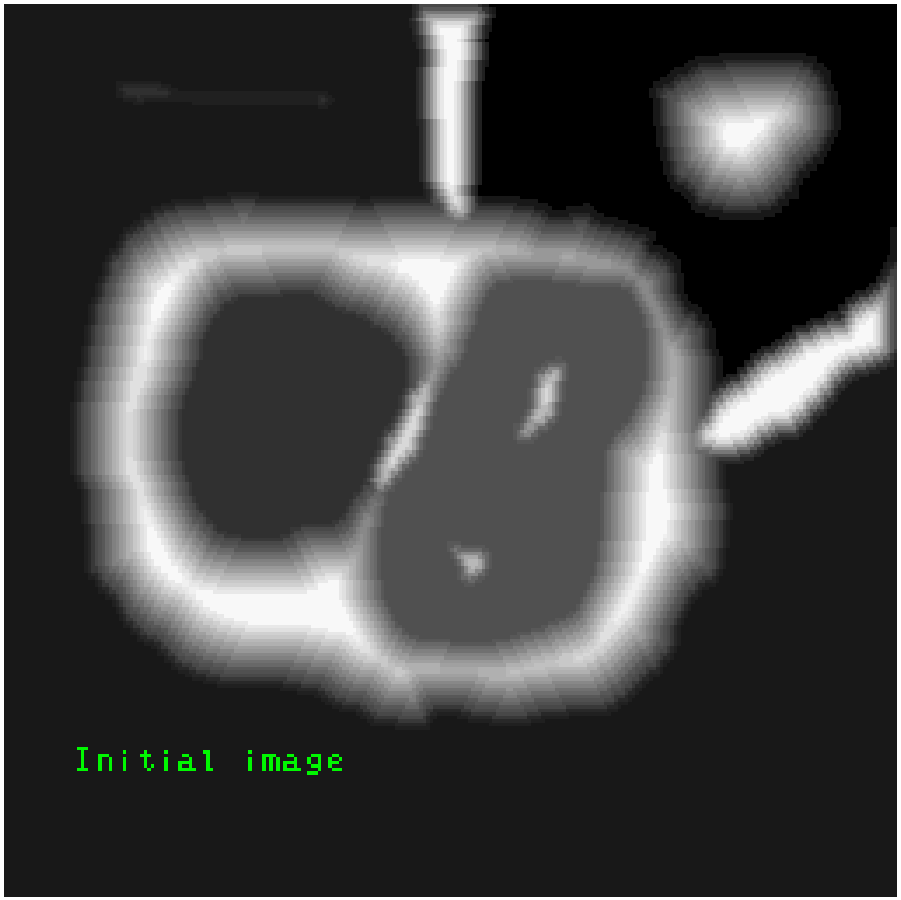
- Goal is to divide the image into a large number of regions, such that each regions lie within object boundaries

- Examples
  - Watershed
  - Felzenszwalb and Huttenlocher graph-based
  - Turbopixels
  - SLIC

Watershed algorithm



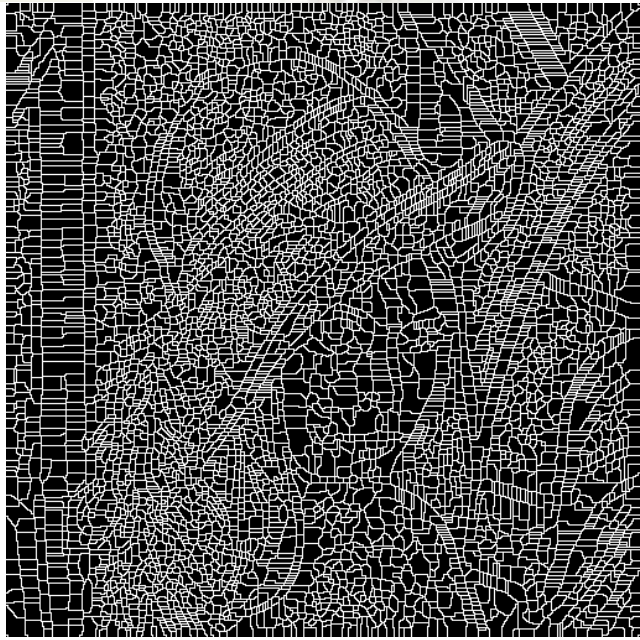
Watershed algorithm: demo



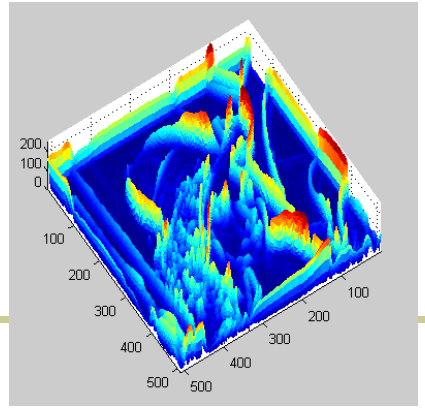
Watershed segmentation

Image

Gradient

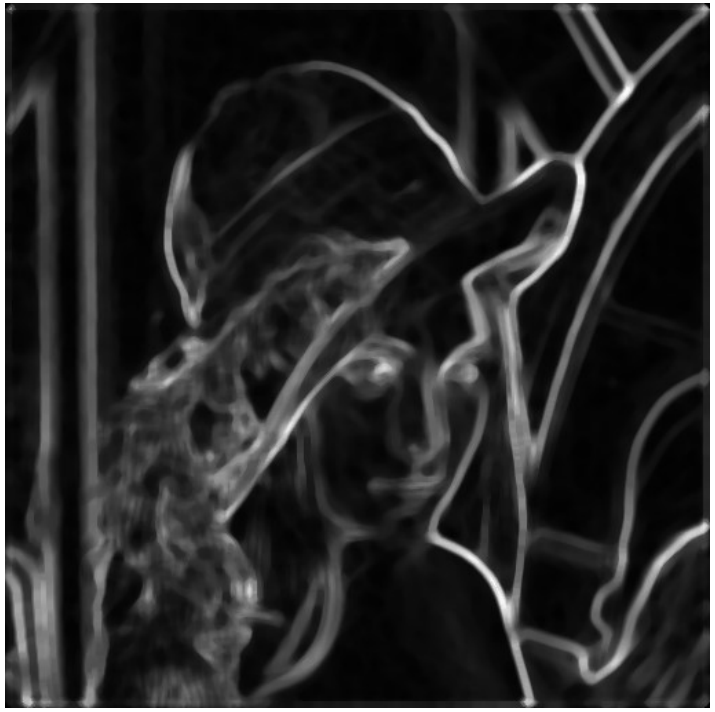


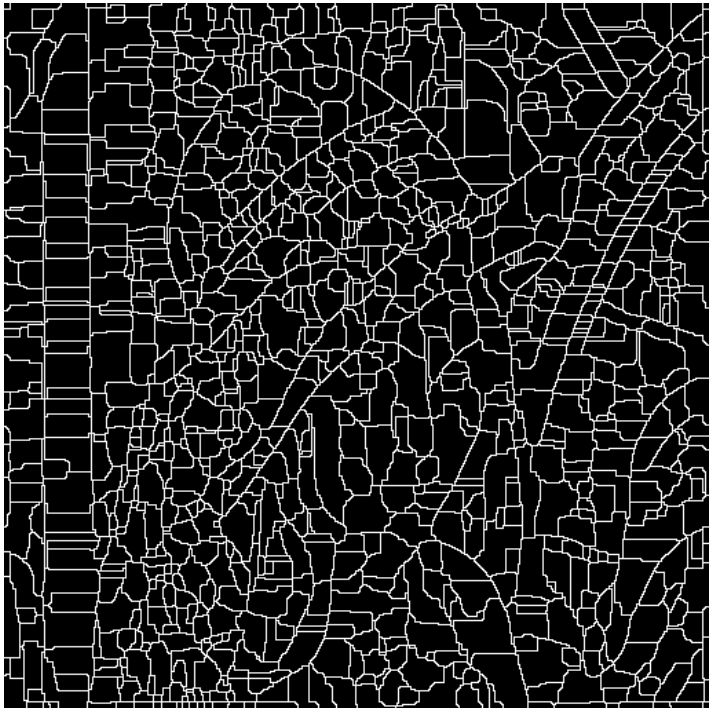
Watershed boundaries



Simple trick

- Use Gaussian or median filter to reduce number of regions





Meyer's watershed segmentation

1. Choose local minima as region seeds
2. Add neighbors to priority queue, sorted by value
3. Take top priority pixel from queue
    1. If all labeled neighbors have same label, assign that label to pixel
    2. Add all non-marked neighbors to queue
4. Repeat step 3 until finished (all remaining pixels in queue are on the boundary)

Matlab: seg = watershed(bnd_im)

Meyer 1991

Watershed usage

- Use as a starting point for hierarchical segmentation
  - Ultrametric contour map (Arbelaez 2006)

- Works with any soft boundaries
  - Pb (w/o non-max suppression)
  - Canny (w/o non-max suppression)
  - Etc.

Watershed usage

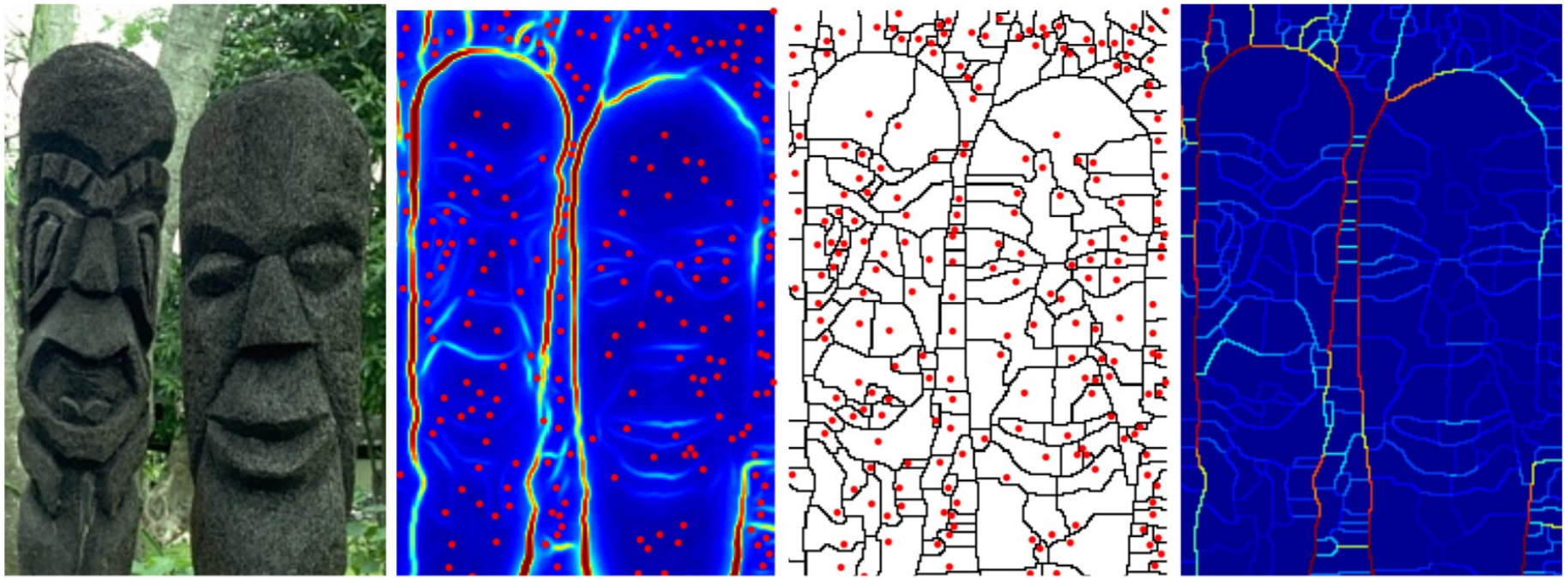


Fig. 11. **Watershed Transform.** ***Left:*** Image. ***Middle Left:*** Boundary strength $E(x, y)$. We regard $E(x, y)$ as a topographic surface and flood it from its local minima. ***Middle Right:*** This process partitions the image into catchment basins $\mathcal{P}_0$ and arcs $\mathcal{K}_0$. There is exactly one basin per local minimum and the arcs coincide with the locations where the floods originating from distinct minima meet. Local minima are marked with red dots. ***Right:*** Each arc weighted by the mean value of $E(x, y)$ along it. This weighting scheme produces artifacts, such as the strong horizontal contours in the small gap between the two statues.

Watershed pros and cons

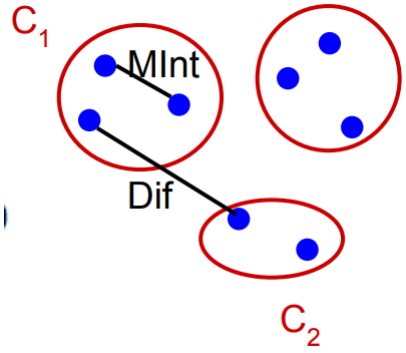
- Pros
  - Fast (< 1 sec for 512x512 image)
  - Preserves boundaries
- Cons
  - Only as good as the soft boundaries (which may be slow to compute)
  - Not easy to get variety of regions for multiple segmentations
- Usage
  - Good algorithm for superpixels, hierarchical segmentation

Graph-Based Segmentation

http://www.cs.brown.edu/~pff/segment/, by Felzenszwalb and Huttenlocher

+ Good for thin regions
+ Fast
+ Easy to control coarseness of segmentations
+ Can include both large and small regions
- Often creates regions with strange shapes
- Sometimes makes very large errors

Predicate for segmentation



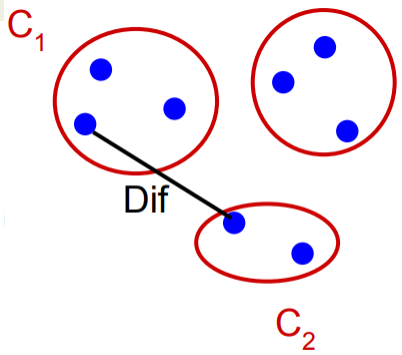
- Predicate D determines whether there is a boundary for segmentation.

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

Where

- dif(C1 , C2) is the difference between two clusters.
- in(C1 , C2) is the internal different in the clusters C1 and C2

Predicate for Segmentation



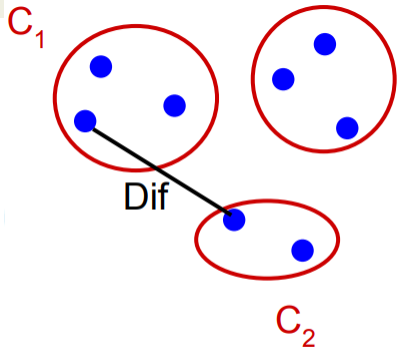
- Predicate D determines whether there is a boundary for segmentation.

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

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

The different between two components is the minimum weight edge that connects a node vi in clusters C1 to node vj in C2

Predicate for Segmentation



- Predicate D determines whether there is a boundary for segme

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

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

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

In(C1, C2) 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.

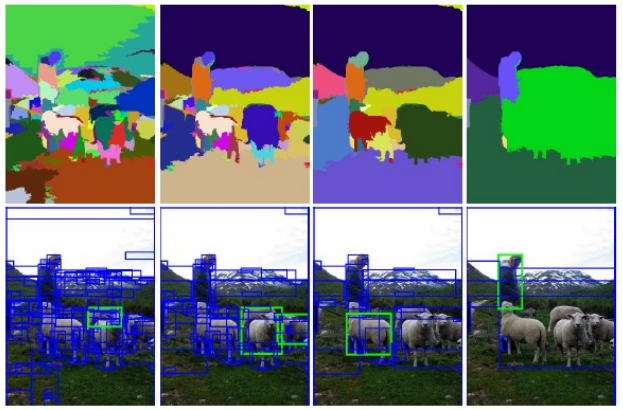
Today's Class

- What are grouping problems in vision?
- Inspiration from human perception
  - Gestalt properties
- Bottom-up segmentation via clustering
  - Mode finding and mean shift: k-means, GMM, mean-shift
- Graph-based segmentation: Normalized Cut
- Oversegmentation
  - Watershed algorithm, Felzenszwalb and Huttenlocher graph-based etc.
- **Multiple segmentation**

Multiple segmentations

- When creating regions for pixel classification or object detection, don't commit to one partitioning

- Strategies:
  - Hierarchical segmentation
    - Occlusion boundaries hierarchy: Hoiem et al. IJCV 2011 (uses trained classifier to merge)
    - Pb+watershed hierarchy: Arbeleaz et al. CVPR 2009
    - Selective search: FH + agglomerative clustering
  - Vary segmentation parameters
    - E.g., multiple graph-based segmentations or mean-shift segmentations
  - Region proposals
    - Propose seed superpixel, try to segment out object that contains it (Endres Hoiem ECCV 2010, Carreira Sminchisescu CVPR 2010)



Pb+watershed hierarchy

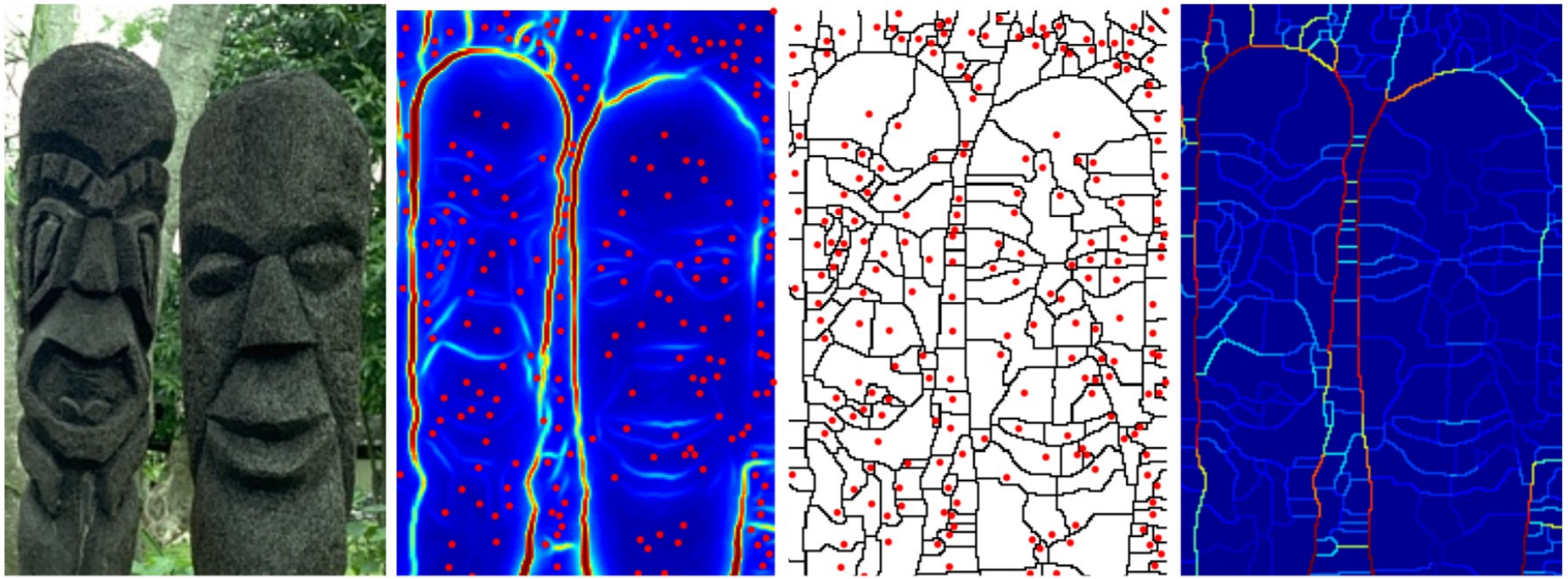


Fig. 11. **Watershed Transform.** ***Left:*** Image. ***Middle Left:*** Boundary strength $E(x, y)$. We regard $E(x, y)$ as a topographic surface and flood it from its local minima. ***Middle Right:*** This process partitions the image into catchment basins $\mathcal{P}_0$ and arcs $\mathcal{K}_0$. There is exactly one basin per local minimum and the arcs coincide with the locations where the floods originating from distinct minima meet. Local minima are marked with red dots. ***Right:*** Each arc weighted by the mean value of $E(x, y)$ along it. This weighting scheme produces artifacts, such as the strong horizontal contours in the small gap between the two statues.

Pb+watershed hierarchy

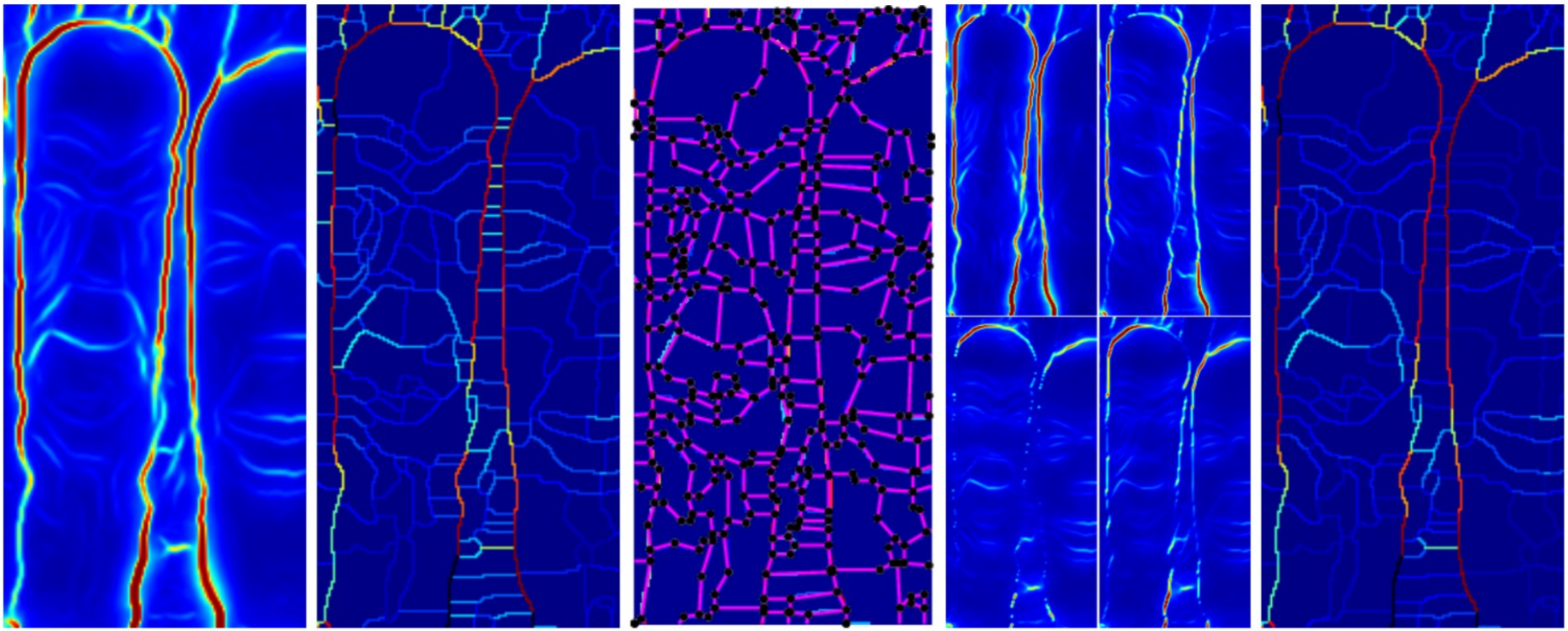


Fig. 12. **Oriented Watershed Transform.** ***Left:*** Input boundary signal $E(x, y) = \max_\theta E(x, y, \theta)$. ***Middle Left:*** Watershed arcs computed from $E(x, y)$. Note that thin regions give rise to artifacts. ***Middle:*** Watershed arcs with an approximating straight line segment subdivision overlaid. We compute this subdivision in a scale-invariant manner by recursively breaking an arc at the point maximally distant from the straight line segment connecting its endpoints, as shown in Figure 13. Subdivision terminates when the distance from the line segment to every point on the arc is less than a fixed fraction of the segment length. ***Middle Right:*** Oriented boundary strength $E(x, y, \theta)$ for four orientations θ. In practice, we sample eight orientations. ***Right:*** Watershed arcs reweighted according to E at the orientation of their associated line segments. Artifacts, such as the horizontal contours breaking the long skinny regions, are suppressed as their orientations do not agree with the underlying $E(x, y, \theta)$ signal.

Pb+watershed hierarchy

1) Select minimum weight contour:
   $C^* = \text{argmin}_{C \in \mathcal{K}_0} W(C)$.
2) Let $R_1, R_2 \in \mathcal{P}_0$ be the regions separated by C^*.
3) Set $R = R_1 \cup R_2$, and update:
   $\mathcal{P}_0 \leftarrow \mathcal{P}_0 \backslash \{R_1, R_2\} \cup \{R\}$ and $\mathcal{K}_0 \leftarrow \mathcal{K}_0 \backslash \{C^*\}$.
4) Stop if $\mathcal{K}_0$ is empty.
   Otherwise, update weights $W(\mathcal{K}_0)$ and repeat.

Pb+watershed hierarchy: Arbeleaz et al. CVPR 2009

Pb+watershed hierarchy

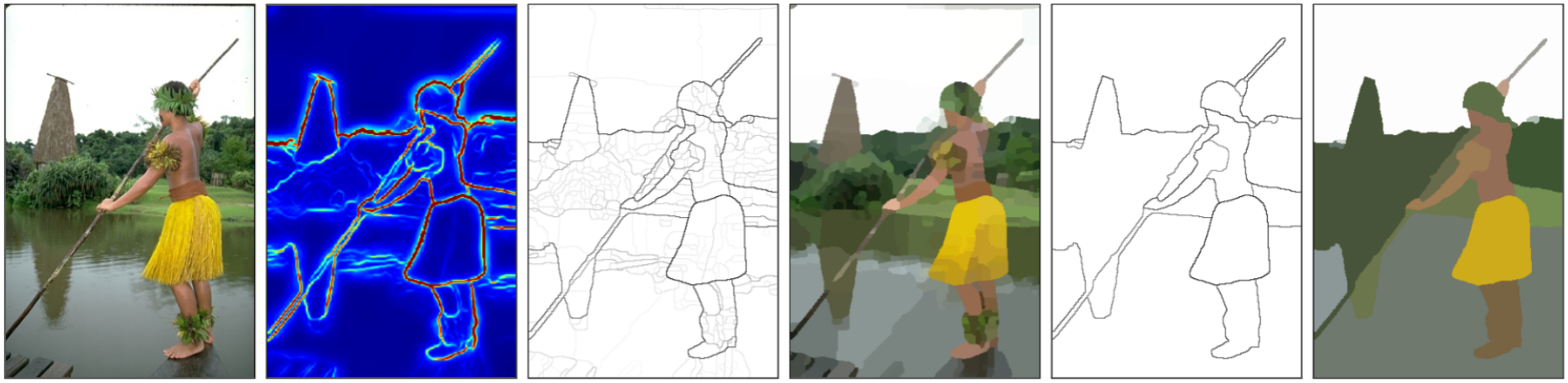


Fig. 14. **Hierarchical segmentation from contours.** ***Far Left:*** Image. ***Left:*** Maximal response of contour detector gPb over orientations. ***Middle Left:*** Weighted contours resulting from the Oriented Watershed Transform - Ultrametric Contour Map (OWT-UCM) algorithm using gPb as input. This single weighted image encodes the entire hierarchical segmentation. By construction, applying any threshold to it is guaranteed to yield a set of closed contours (the ones with weights above the threshold), which in turn define a segmentation. Moreover, the segmentations are nested. Increasing the threshold is equivalent to removing contours and merging the regions they separated. ***Middle Right:*** The initial oversegmentation corresponding to the finest level of the UCM, with regions represented by their mean color. ***Right and Far Right:*** Contours and corresponding segmentation obtained by thresholding the UCM at level 0.5.

Pb+watershed hierarchy

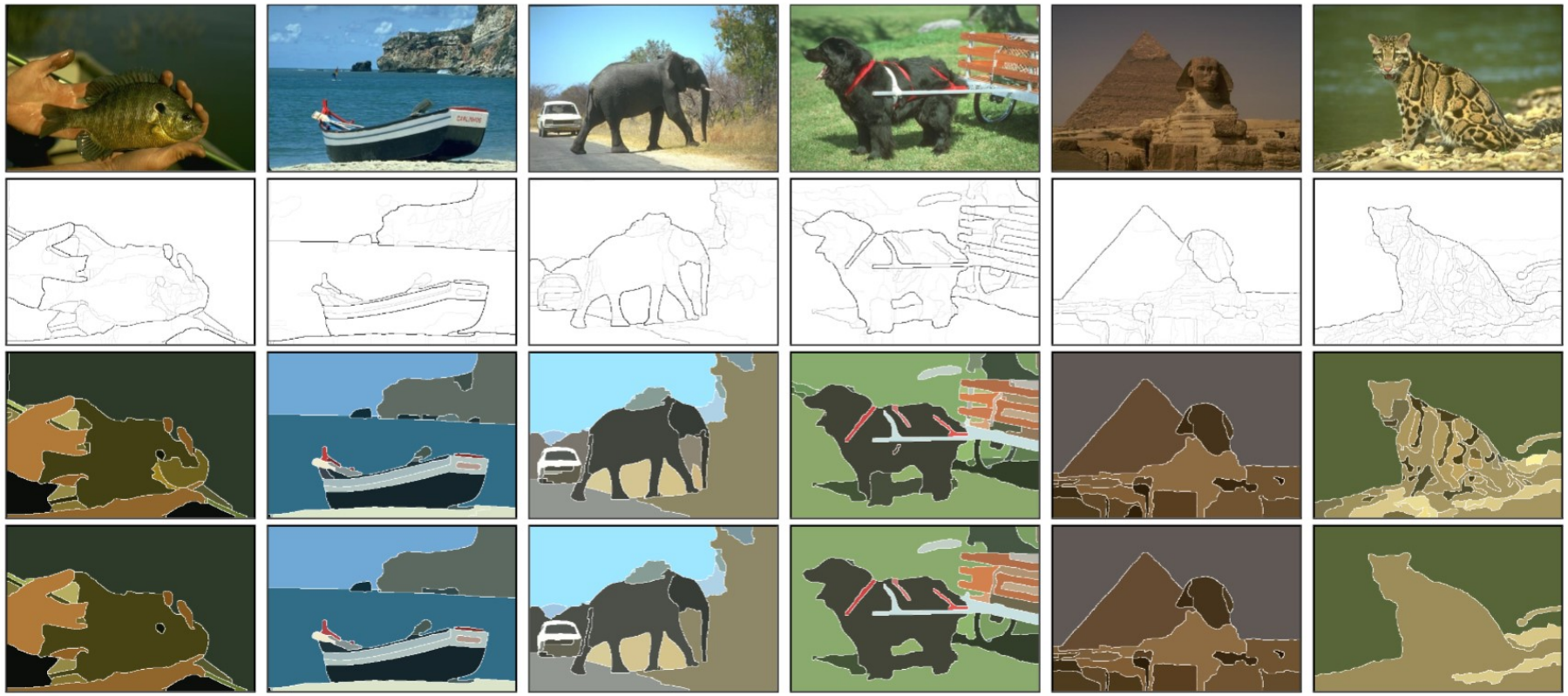


Fig. 16. **Additional hierarchical segmentation results on the BSDS500.** ***From Top to Bottom:*** Image, UCM produced by gPb-owt-ucm, and ODS and OIS segmentations. All images are from the test set.

Benchmark

	BSDS300			BSDS500		
	ODS	OIS	AP	ODS	OIS	AP
Human	0.79	0.79	—	0.80	0.80	—
gPb-owt-ucm	**0.71**	**0.74**	**0.73**	**0.73**	**0.76**	**0.73**
[34] Mean Shift	0.63	0.66	0.54	0.64	0.68	0.56
[33] NCuts	0.62	0.66	0.43	0.64	0.68	0.45
Canny-owt-ucm	0.58	0.63	0.58	0.60	0.64	0.58
[32] Felz-Hutt	0.58	0.62	0.53	0.61	0.64	0.56
[31] SWA	0.56	0.59	0.54	—	—	—
Quad-Tree	0.37	0.39	0.26	0.38	0.39	0.26
gPb	0.70	0.72	0.66	0.71	0.74	0.65
Canny	0.58	0.62	0.58	0.60	0.63	0.58

TABLE 1. **Boundary benchmarks on the BSDS.** Results for seven different segmentation methods (upper table) and two contour detectors (lower table) are given. Shown are the F-measures when choosing an optimal scale for the entire dataset (ODS) or per image (OIS), as well as the average precision (AP). Figures 1, 2, and 17 show the full precision-recall curves for these algorithms. Note that the boundary benchmark has the largest discriminative power among the evaluation criteria, clearly separating the Quad-Tree from all the data-driven methods.

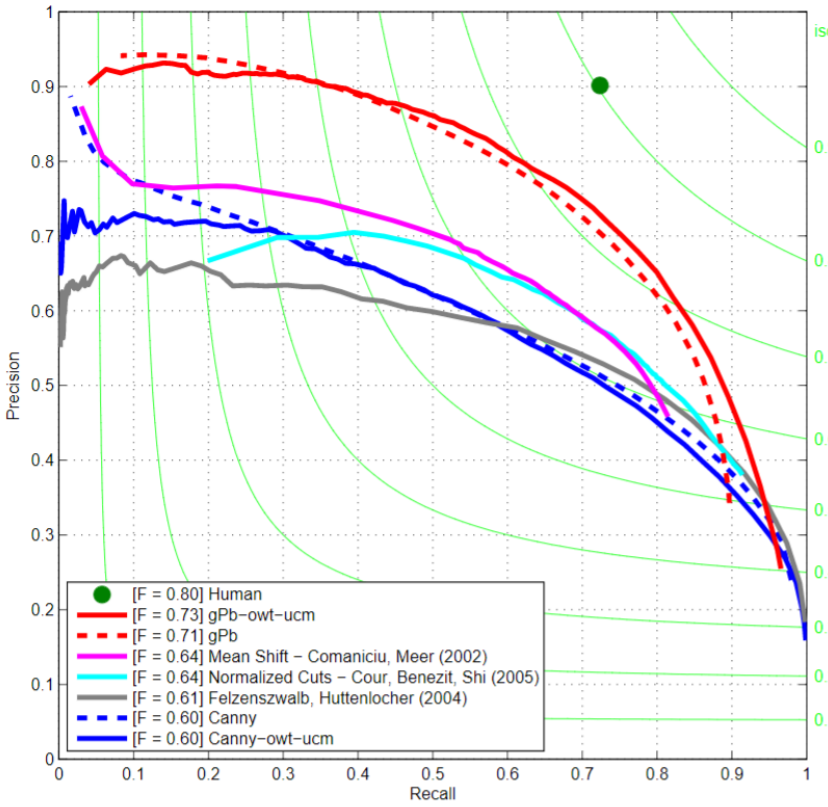


Fig. 17. **Boundary benchmark on the BSDS500.** Co
paring boundaries to human ground-truth allows us
evaluate contour detectors [3], [22] (dotted lines) and se
mentation algorithms [4], [32], [33], [34] (solid lines) in t
same framework. Performance is consistent when goi
from the BSDS300 (Figures 1 and 2) to the BSDS5
(above). Furthermore, the **OWT-UCM algorithm p
serves contour detector quality**. For both gPb a
Canny, comparing the resulting segment boundaries
the original contours shows that our OWT-UCM algorit
constructs hierarchical segmentations from contours wi
out losing performance on the boundary benchmark.

Benchmark

	BSDS300						
	Covering			PRI		VI	
	ODS	OIS	Best	ODS	OIS	ODS	OIS
Human	0.73	0.73	–	0.87	0.87	1.16	1.16
gPb-owt-ucm	**0.59**	**0.65**	**0.75**	**0.81**	**0.85**	**1.65**	**1.47**
[34] Mean Shift	0.54	0.58	0.66	0.78	0.80	1.83	1.63
[32] Felz-Hutt	0.51	0.58	0.68	0.77	0.82	2.15	1.79
Canny-owt-ucm	0.48	0.56	0.66	0.77	0.82	2.11	1.81
[33] NCuts	0.44	0.53	0.66	0.75	0.79	2.18	1.84
[31] SWA	0.47	0.55	0.66	0.75	0.80	2.06	1.75
[29] Total Var.	0.57	–	–	0.78	–	1.81	–
[70] T+B Encode	0.54	–	–	0.78	–	1.86	–
[30] Av. Diss.	0.47	–	–	0.76	–	2.62	–
[30] ChanVese	0.49	–	–	0.75	–	2.54	–
Quad-Tree	0.33	0.39	0.47	0.71	0.75	2.34	2.22

	BSDS500						
	Covering			PRI		VI	
	ODS	OIS	Best	ODS	OIS	ODS	OIS
Human	0.72	0.72	–	0.88	0.88	1.17	1.17
gPb-owt-ucm	**0.59**	**0.65**	**0.74**	**0.83**	**0.86**	**1.69**	**1.48**
[34] Mean Shift	0.54	0.58	0.66	0.79	0.81	1.85	1.64
[32] Felz-Hutt	0.52	0.57	0.69	0.80	0.82	2.21	1.87
Canny-owt-ucm	0.49	0.55	0.66	0.79	0.83	2.19	1.89
[33] NCuts	0.45	0.53	0.67	0.78	0.80	2.23	1.89
Quad-Tree	0.32	0.37	0.46	0.73	0.74	2.46	2.32

TABLE 2. **Region benchmarks on the BSDS.** For each segmentation method, the leftmost three columns report the score of the covering of ground-truth segments according to optimal dataset scale (ODS), optimal image scale (OIS), or Best covering criteria. The rightmost four columns compare the segmentation methods against ground-truth using the Probabilistic Rand Index (PRI) and Variation of Information (VI) benchmarks, respectively. Among the region benchmarks, the covering criterion has the largest dynamic range, followed by PRI and VI.

Summary

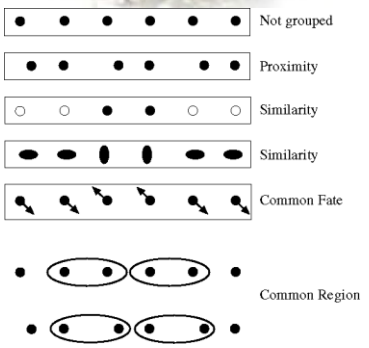
- What are grouping problems in vision?
- Inspiration from human perception
  - Gestalt properties
- Bottom-up segmentation via clustering
  - Mode finding and mean shift: k-means, GMM, mean-shift
- Graph-based segmentation: Normalized Cut
- Oversegmentation
  - Watershed algorithm, Felzenszwalb and Huttenlocher graph-based
- Multiple segmentation

Summary of segmentation algorithms

- Oversegmentation
  - Watershed + Pb
  - Felzenszwalb and Huttenlocher 2004
    http://www.cs.brown.edu/~pff/segment/
  - SLIC ← good recent option
  - Turbopixels
  - Mean-shift

- Larger regions
  - Hierarchical segmentation (e.g., from Pb)
  - Normalized cuts
  - Mean-shift
  - Seed + graph cuts

Things to remember



- Gestalt cues and principles of organization
- Uses of segmentation
  - Efficiency
  - Better features
  - Propose object regions
  - Want the segmented object
- Mean-shift segmentation
  - Good general-purpose segmentation method
  - Generally useful clustering, tracking technique
- Normalized cuts
  - Produces regular regions
  - Slow but good for oversegmentation
- Watershed segmentation
  - Good for hierarchical segmentation
  - Use in combination with boundary prediction