Lecture 14

Segmentation and clustering

Administrative

A2 grades are out

A3 is out

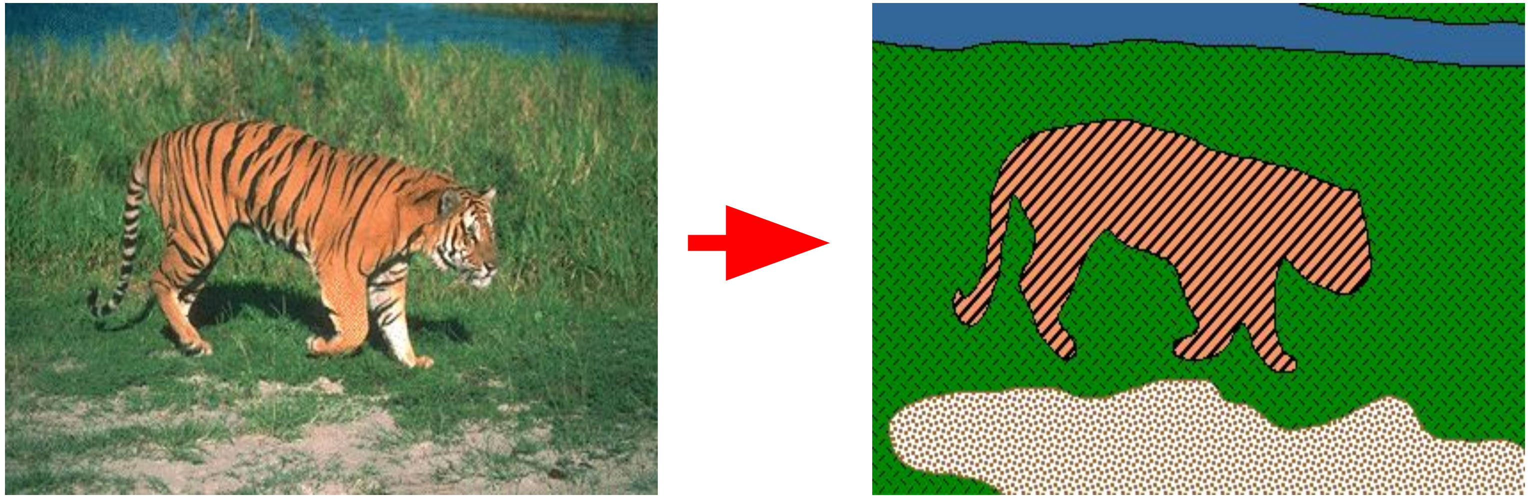
- Due May 19

CS455 Roadmap

Pixels	Video	Camera	Segment	ML
Convolutions Edges Descriptors	Motion Tracking	Camera 3D Geometry	Segmentation Clustering Detection	Linear Models (Conv) Neural networks

What is image segmentation?

- Identify groups of pixels that go together and are meaningful in some sense.

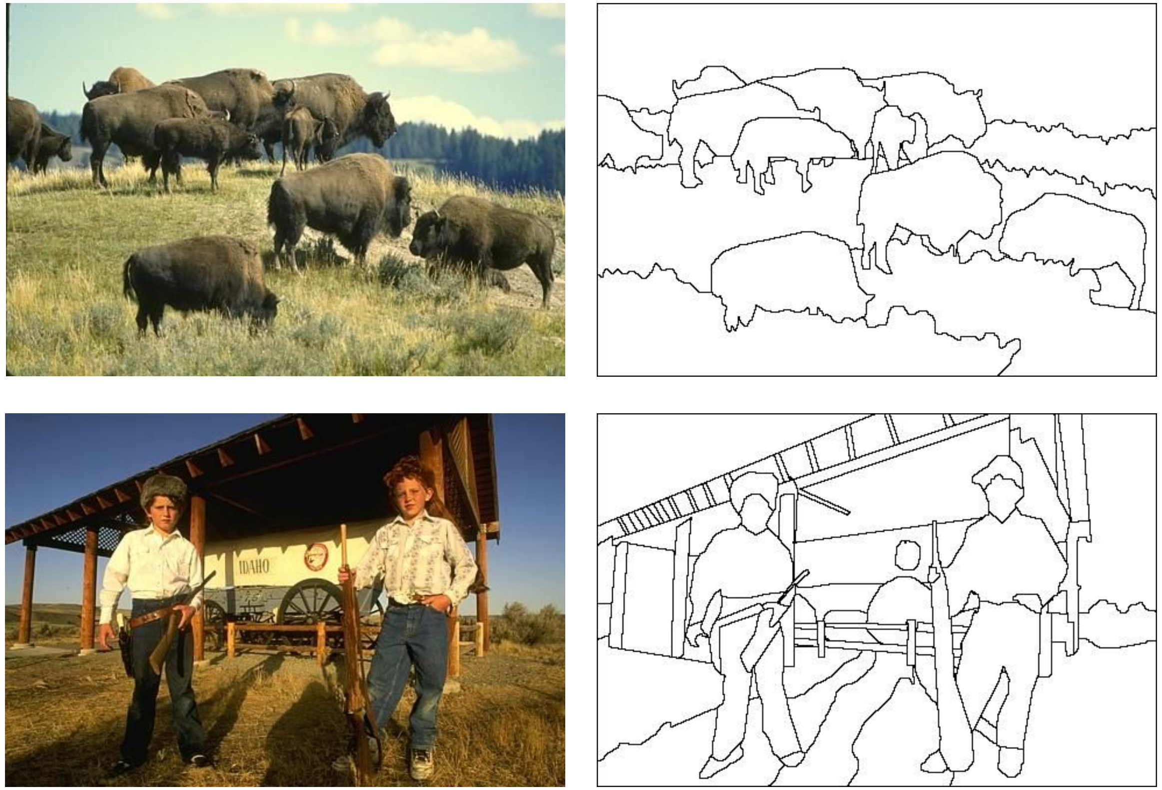


Slide credit: Steve Seitz, Kristen Grauman

Why do we segment?

- Separate image into coherent “objects”

Image | Human segmentation

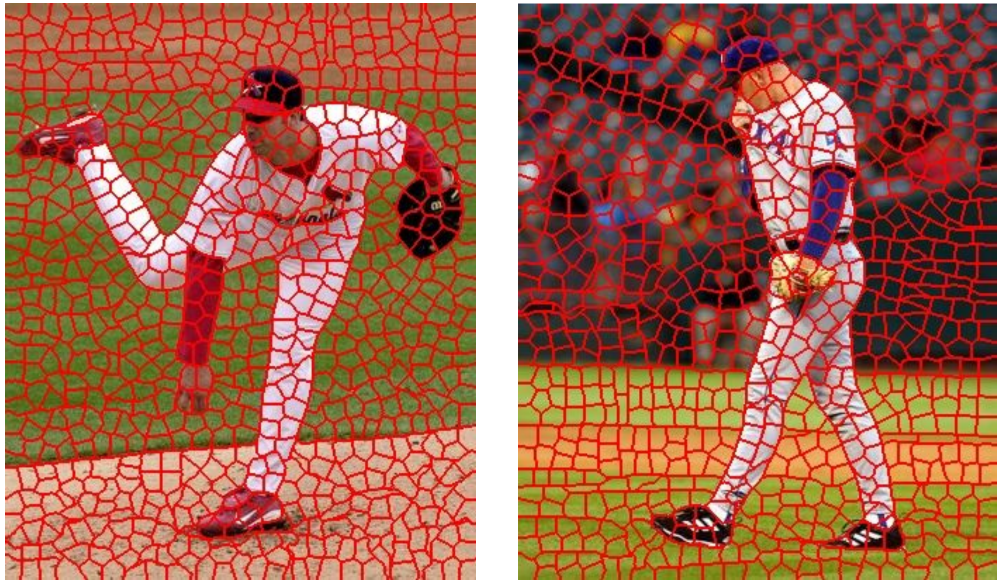


Slide credit: Svetlana Lazebnik

Why do we segment?

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

“superpixels”



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

Why do we segment?

- **Summarizing data**
  - Look at large amounts of data
    - Find group of pixels
    - Represent each group of pixels with feature vectors e.g., HoG

- **Counting**
  - Histograms of texture, color, SIFT vectors

- **Foreground-background separation**
  - Separate the image into different regions

- **Prediction**
  - Images in the same group may have the same labels

Slide: Derek Hoiem

Segmentation is used in Adobe photoshop to remove background

Rother et al. 2004

[Segment Anything](#) [2023]

Segmentation is ill-defined problem, “correct” segments depends on the context

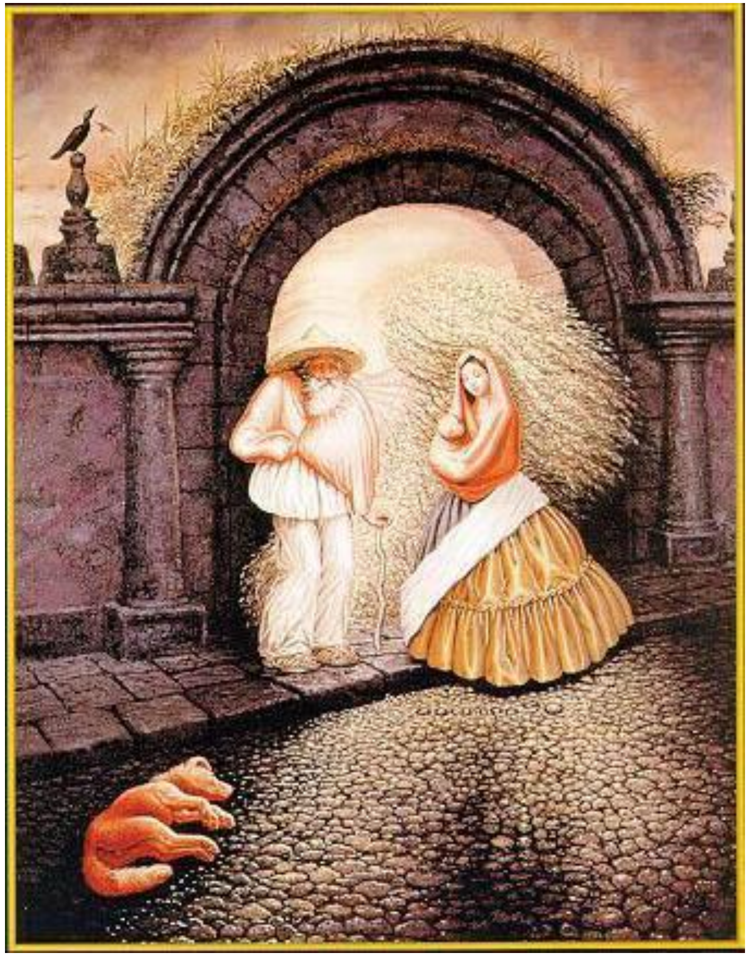
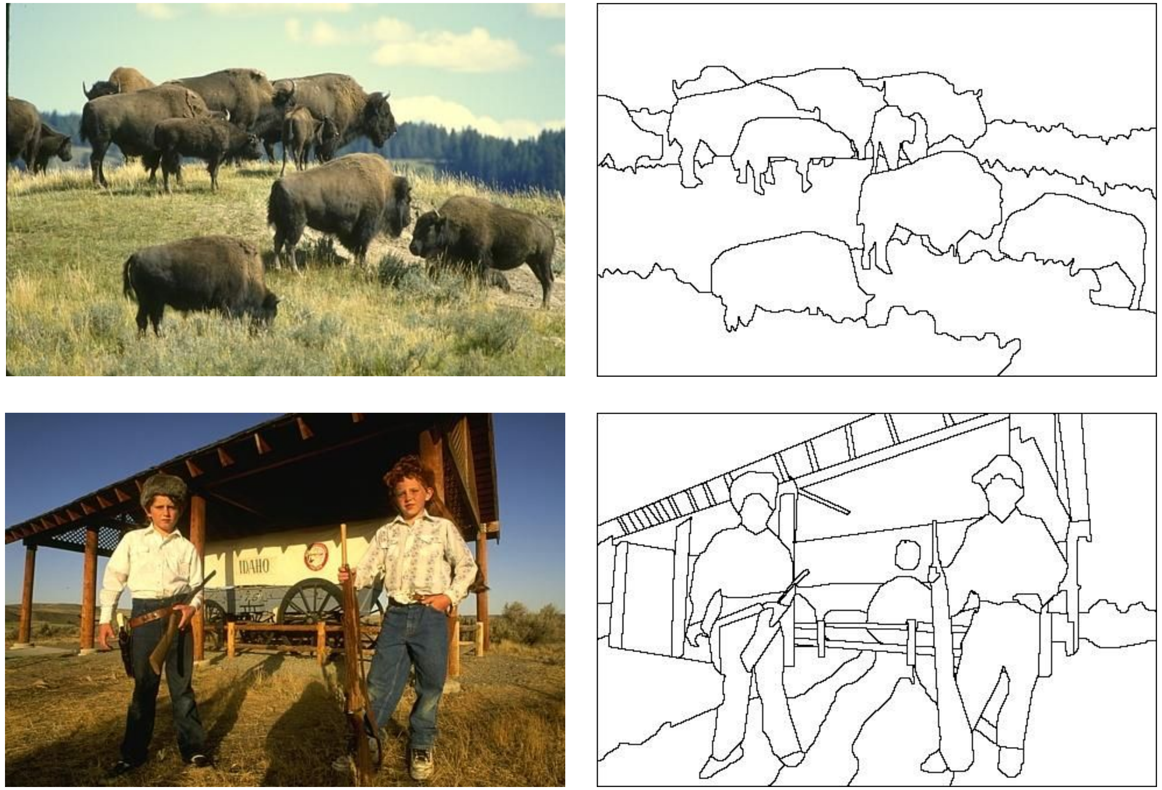
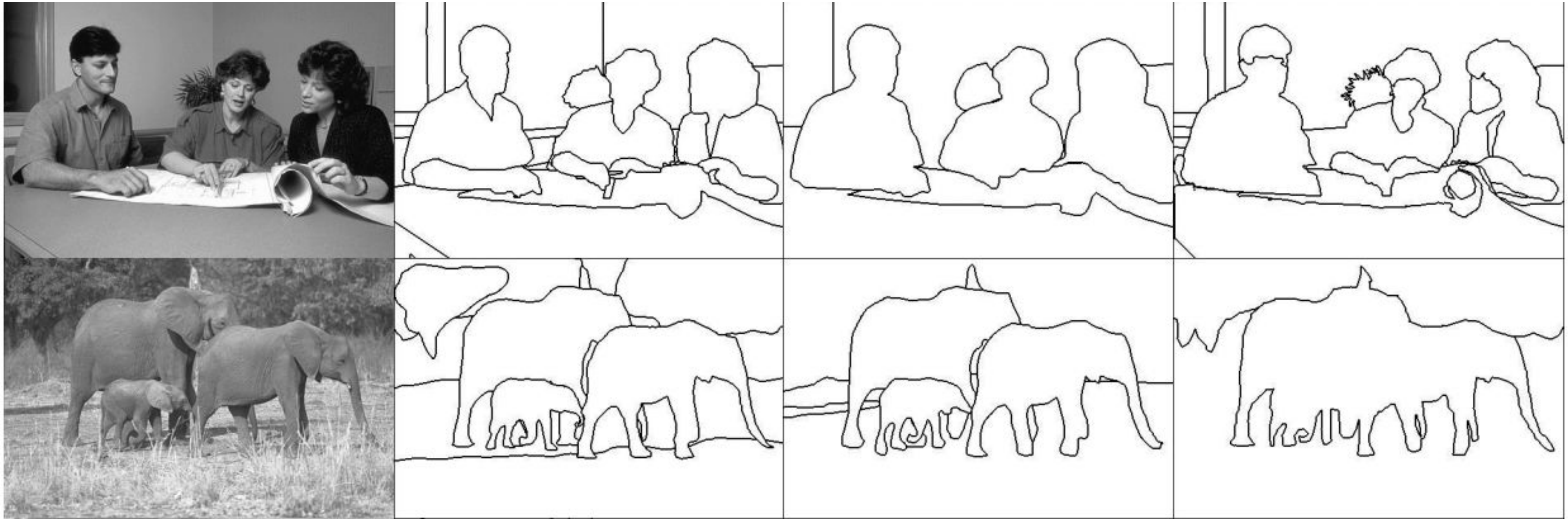


Image | Human segmentation



Segmentation in humans: subjective, intuitive



User 1 User 2 User 3

[Martin 2001]

Today's agenda

- Gestalt theory for perceptual grouping
- Segmentation as clustering
- Agglomerative clustering

Reading:
Szeliski, 2^{nd} edition, Chapter 7.5

Today's agenda

- Gestalt theory for perceptual grouping
- Segmentation as clustering
- Agglomerative clustering

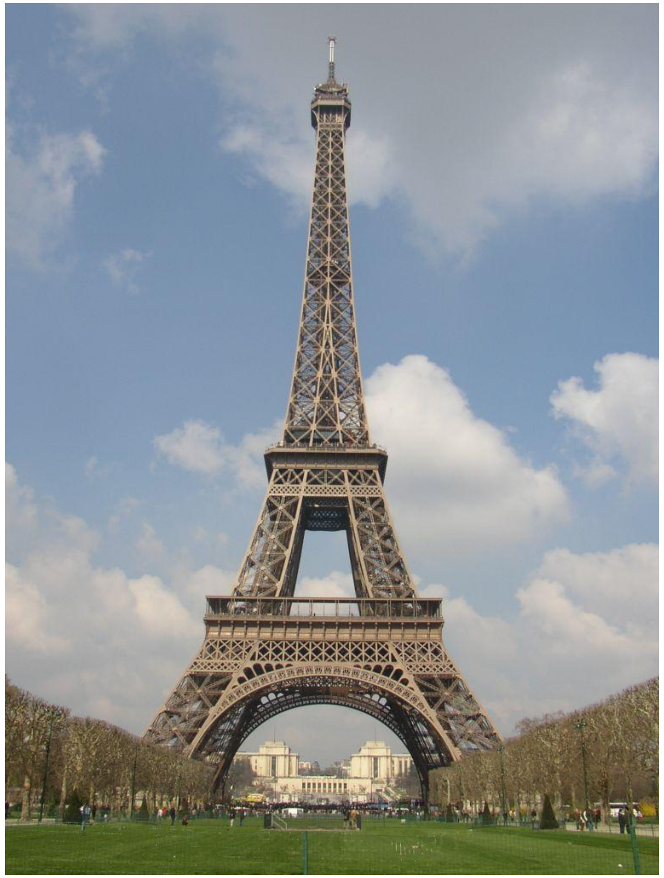
Reading:
Szeliski, 2nd edition, Chapter 7.5

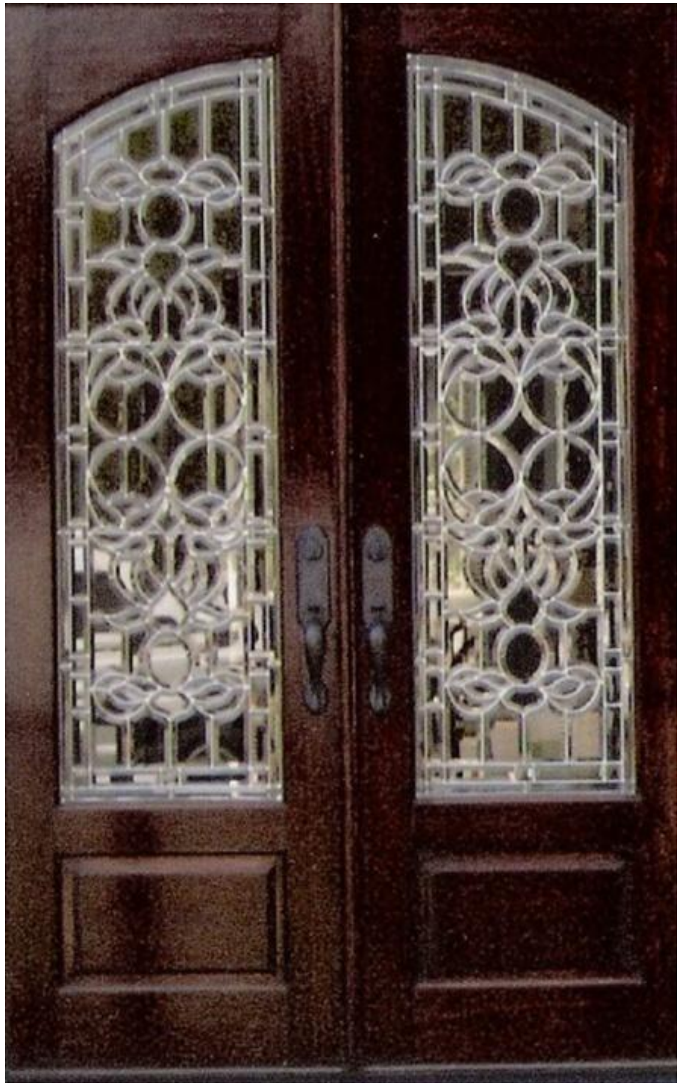
Similarity

What things should be grouped?

What cues indicate groups?

Symmetry





Slide credit: Kristen Grauman

Common Fate

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

Slide credit: Kristen Grauman

Proximity

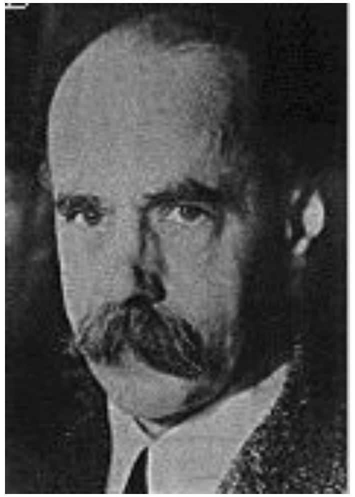
Gestalt Theory

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)



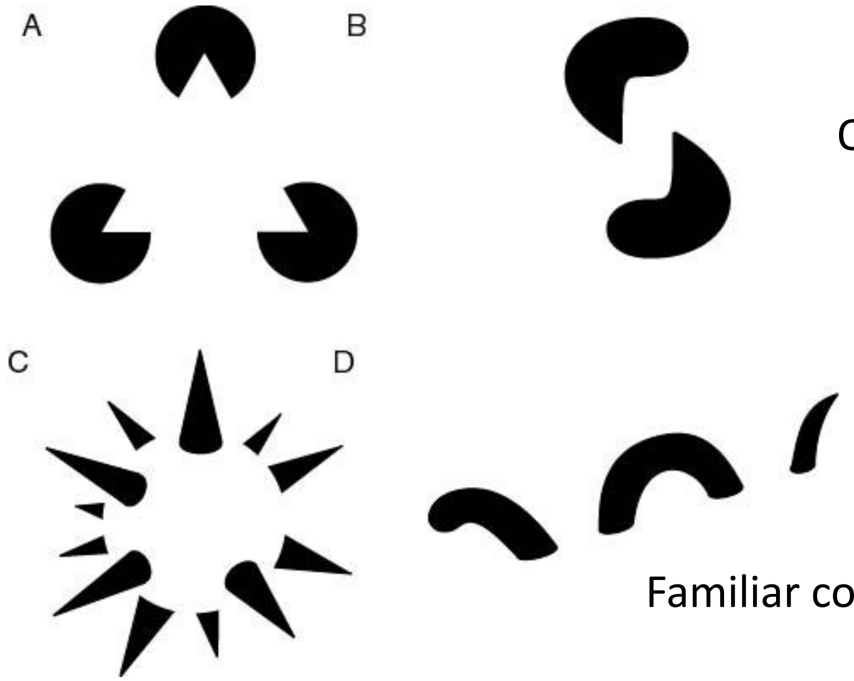
Untersuchungen zur Lehre von der Gestalt,
Psychologische Forschung, Vol. 4, pp. 301-350, 1923
http://psy.ed.asu.edu/~classics/Wertheimer/Forms/forms.htm

Gestalt Theory

- 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”

Illusory/subjective contours

Occlusion

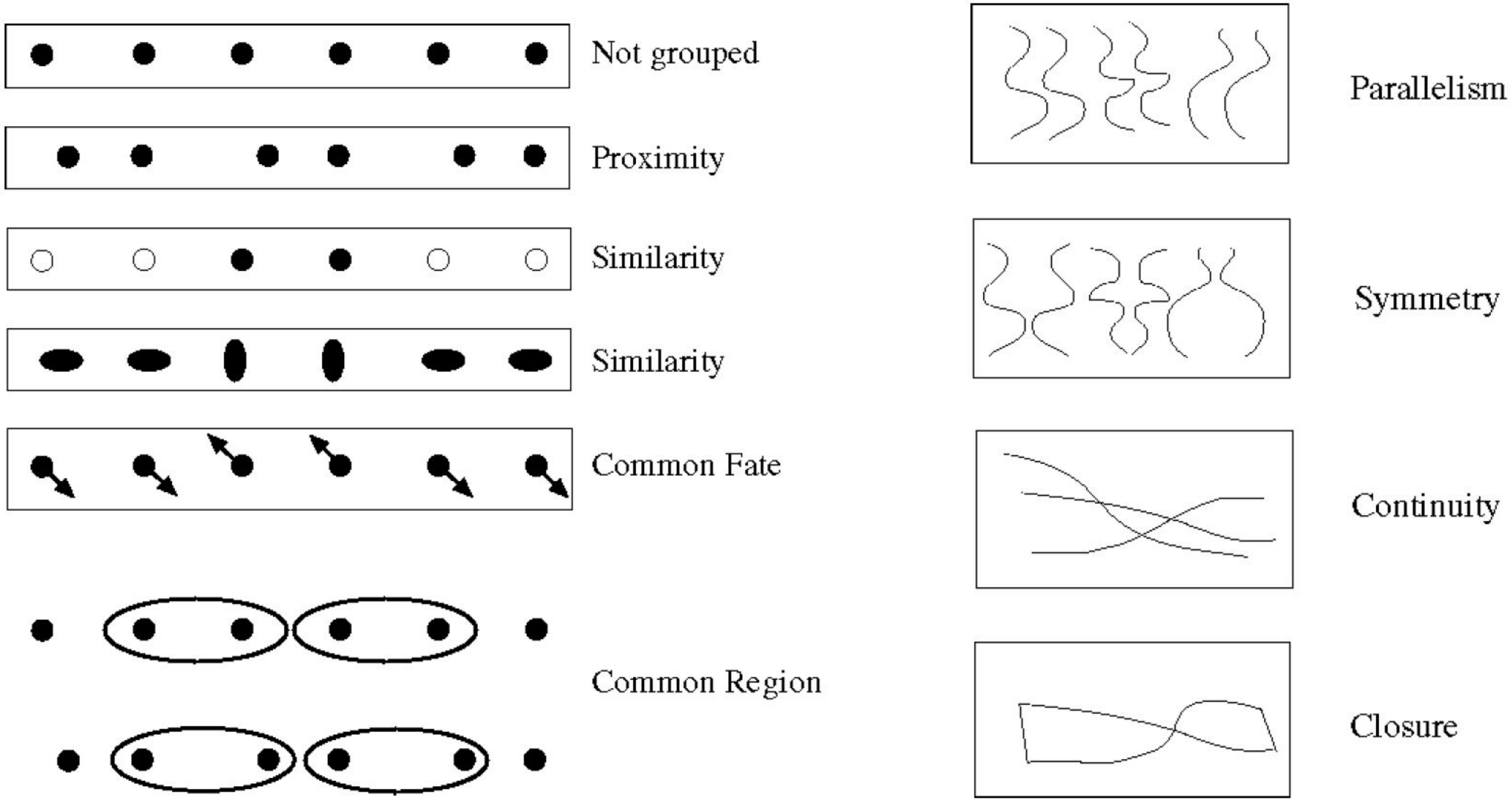


Familiar configuration

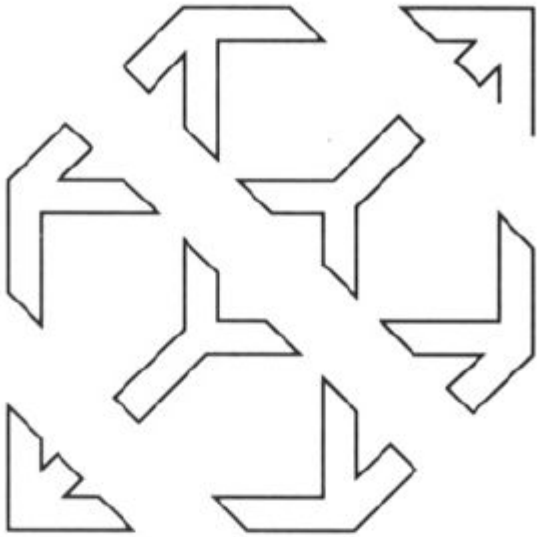
http://en.wikipedia.org/wiki/Gestalt_psychology

Gestalt Factors

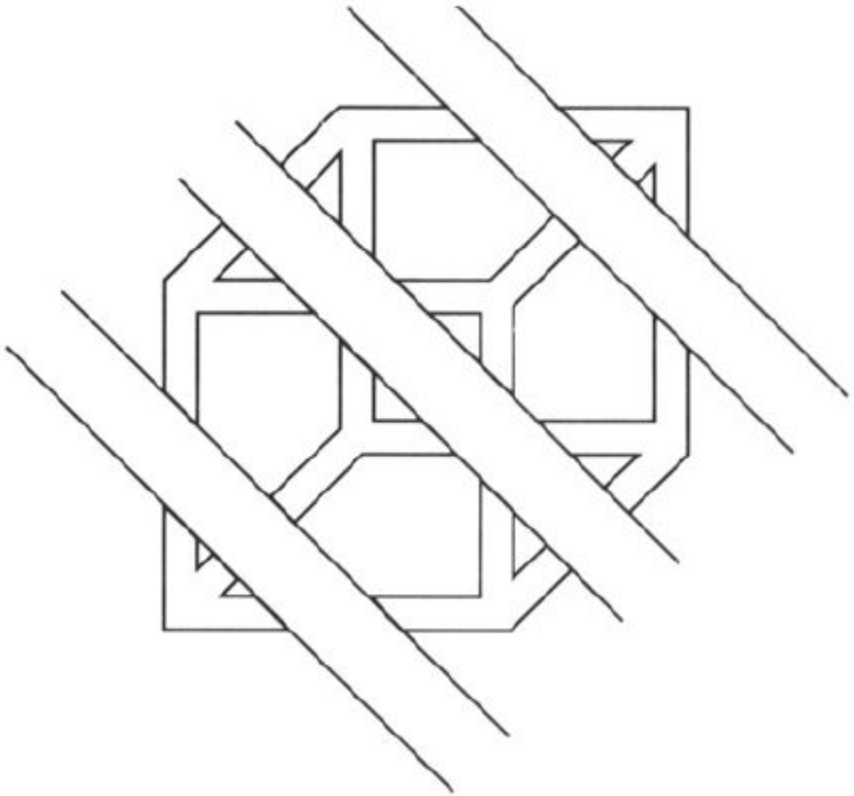
These factors make intuitive sense, but are very difficult to translate into algorithms.



Continuity through Occlusion Cues



Continuity through Occlusion Cues



Continuity, explanation by occlusion

Today's agenda

- Gestalt theory for perceptual grouping
- Segmentation as clustering
- Agglomerative clustering

Segmentation strategies

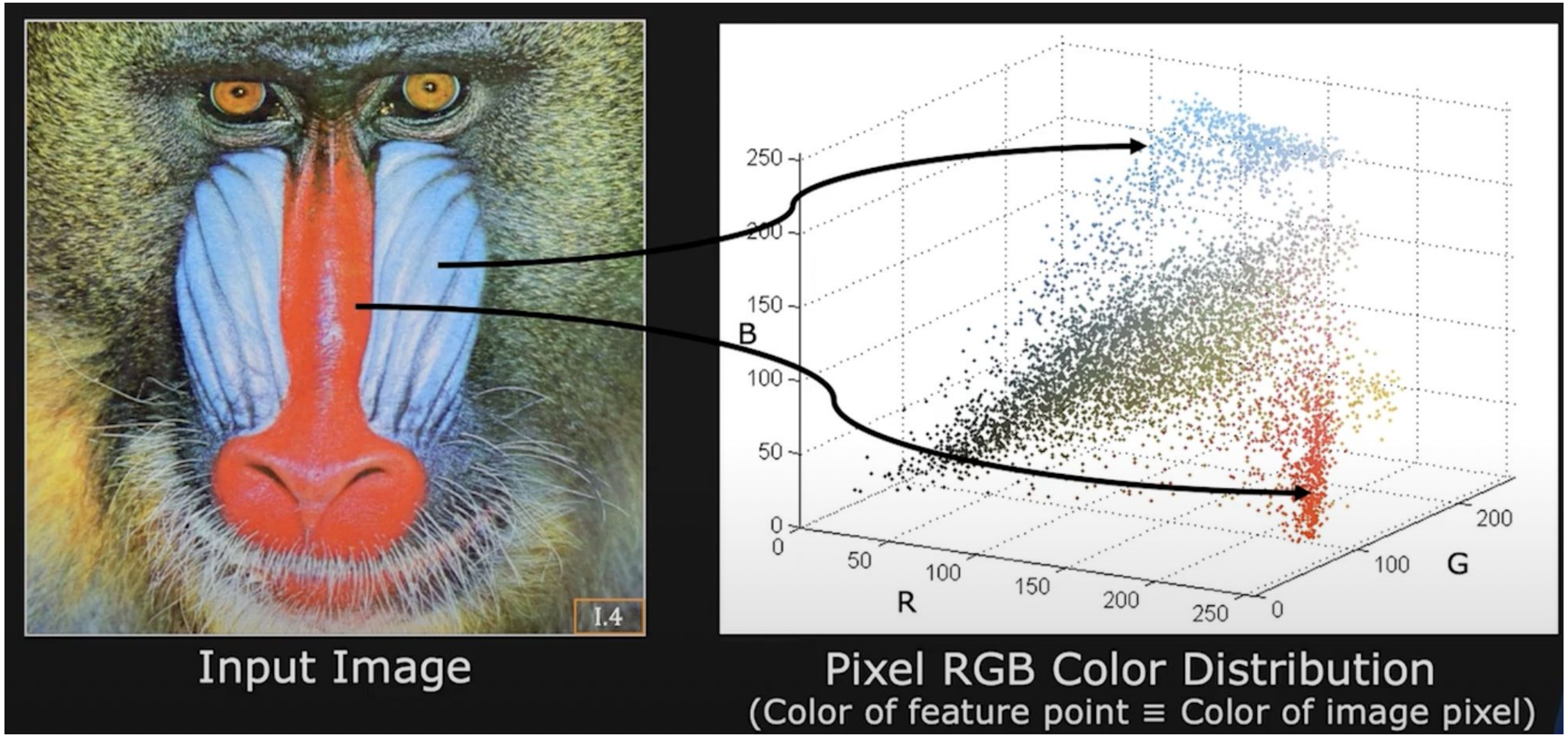
- Top down clustering
  - pixels belong together because they lie on the same visual entity (object, scene…)

- Bottom up clustering
  - pixels belong together because they look similar

These two are not mutually exclusive!

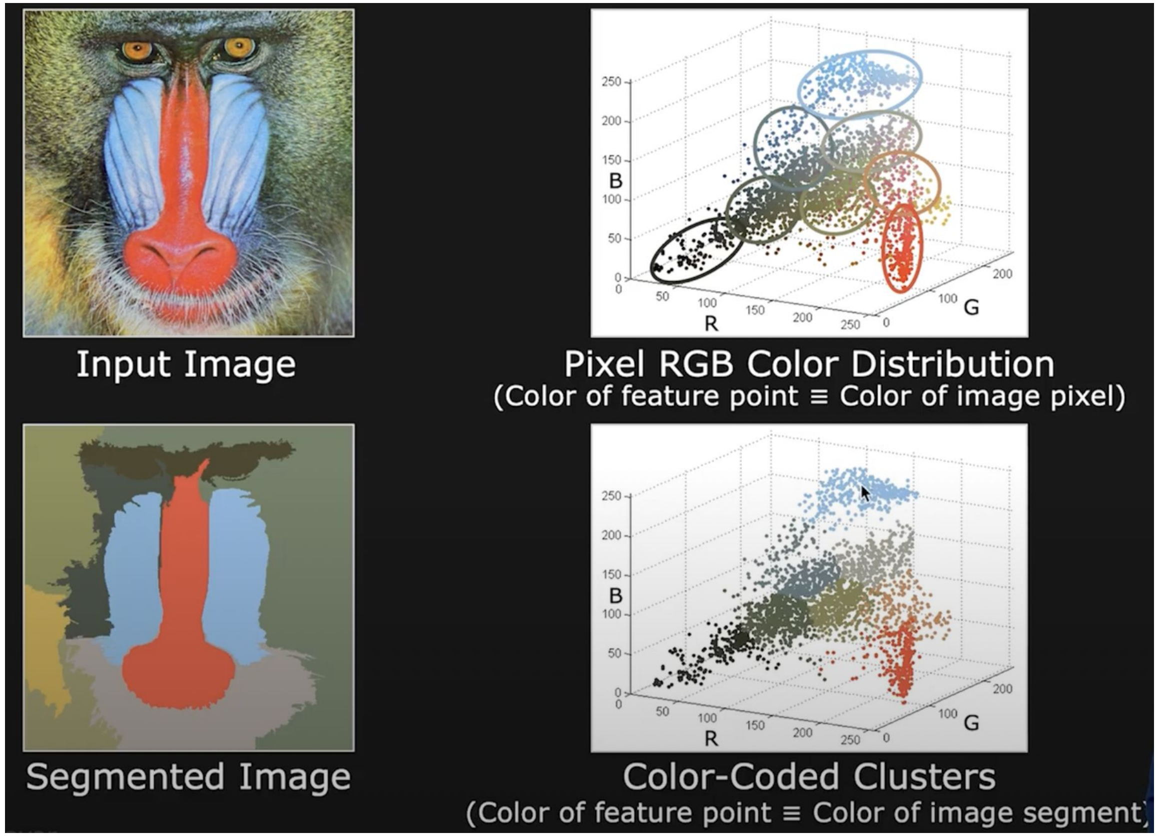
Segmentation as clustering

Clustering: group together similar data points, usually in feature space



Segmentation as clustering

Clustering: group together similar data points, usually in 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

Segmentation as clustering

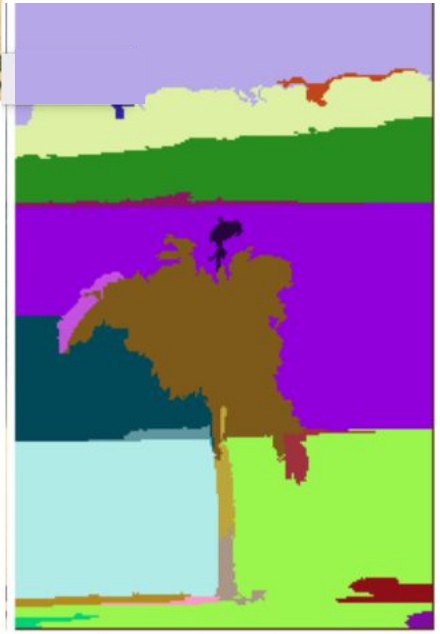
Clustering: group together similar data points, usually in feature space

Key Challenges:

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

Slide: Derek Hoiem

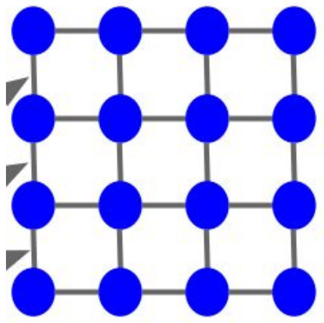
Over-segmenting images



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

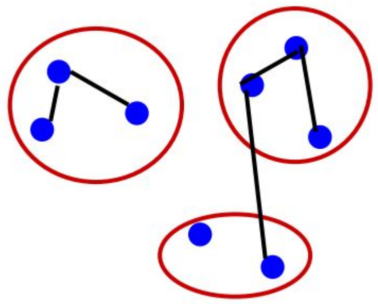
Image as a Graph - Features and weights

- 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 dist(x, x') distance in feature space.
  - where x and x' are two neighboring pixels



- Q. What is a good feature space?

Problem Formulation



- Graph G = (V, E)
- V is set of nodes (i.e. pixels)
- E is a set of undirected edges between pairs of pix
- 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’ ⊂ 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.

Distance Measures

Clustering is an unsupervised learning method. Given items $v_1, v_2, \ldots, v_n \in \mathcal{R}^D$, 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.*

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

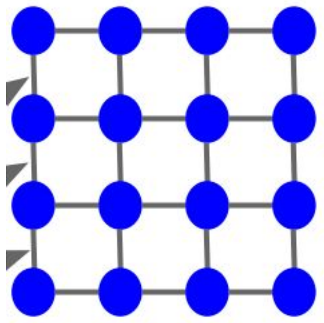
$$\begin{aligned} dist(v_1, v_2) &= 1 - cos(v_1, v_2) \\ &= 1 - \frac{v_1^T v_2}{||v_1|| \cdot ||v_2||} \end{aligned}$$

How do we cluster?

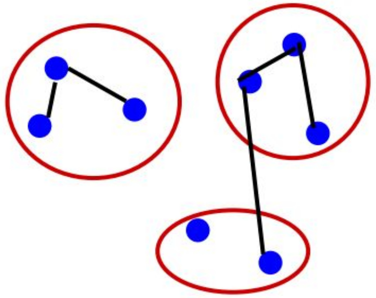
- **Agglomerative clustering**
  - Start with each point as its own cluster and iteratively merge the closest clusters
- **K-means**
  - Iteratively re-assign points to the nearest cluster center
- **Mean-shift clustering**
  - Estimate modes of pdf

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 dist(x, x’) distance in feature space.
  - where x and x’ are two neighboring pixels



Segmentation using graph-cut



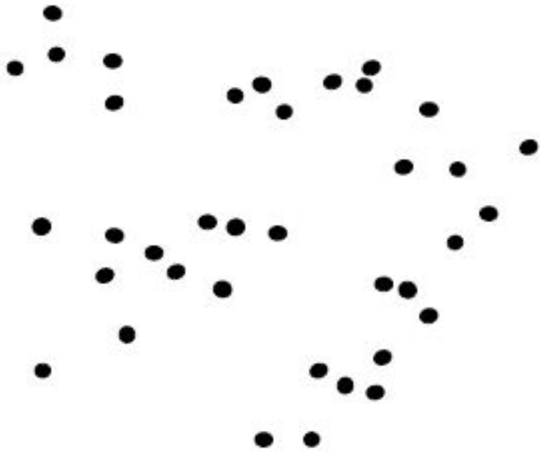
- Graph G = (V, E)
- V is set of nodes (i.e. pixels)
- E is a set of undirected edges between pairs of pix
- dist(v_i , v_j) is the weight/distance of the edge between nodes v_i and v_j.

- S is a segmentation of a graph G such that G' = (V, E') where E' ⊂ 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.

Today's agenda

- Gestalt theory for perceptual grouping
- Segmentation as clustering
- Agglomerative clustering

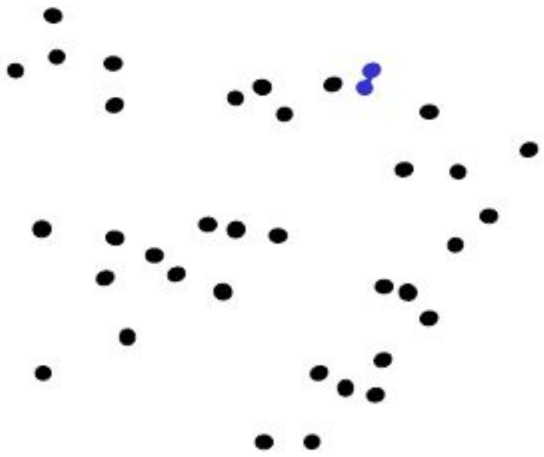
Agglomerative clustering



1. Say "Every point is its own cluster"

Slide credit: Andrew Moore

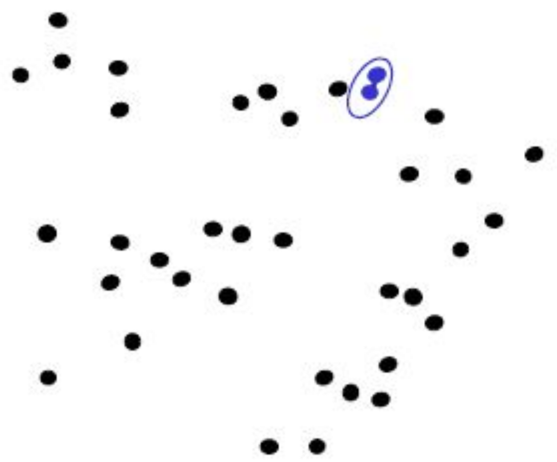
Agglomerative clustering



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

Slide credit: Andrew Moore

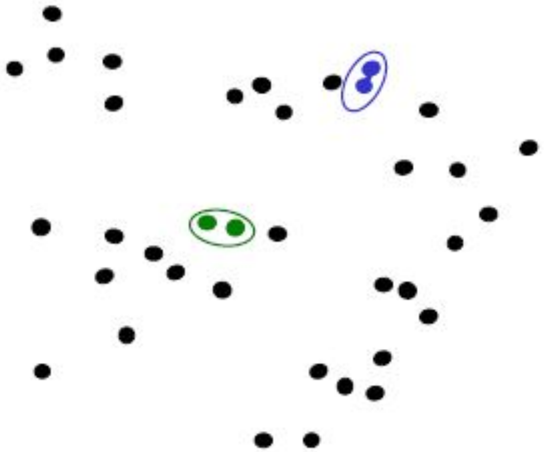
Agglomerative clustering



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

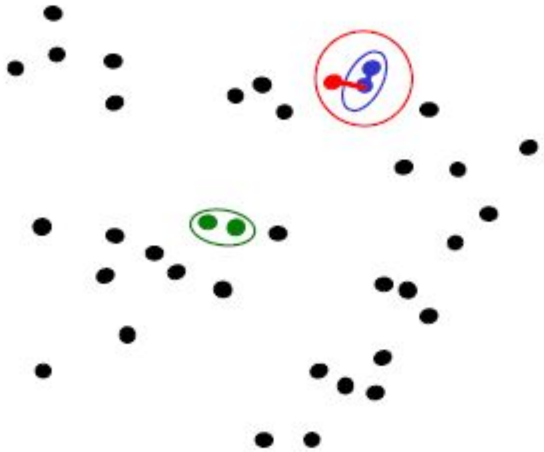
Agglomerative clustering



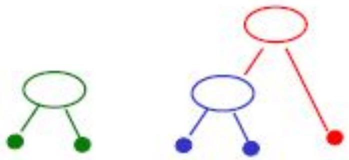
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

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

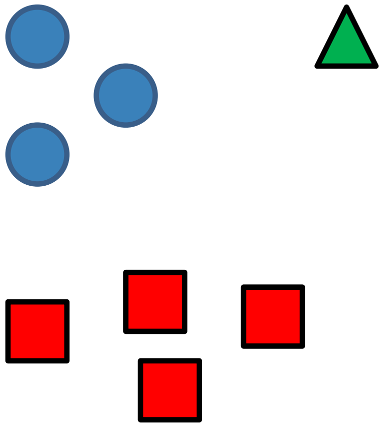


Slide credit: Andrew Moore

Agglomerative clustering

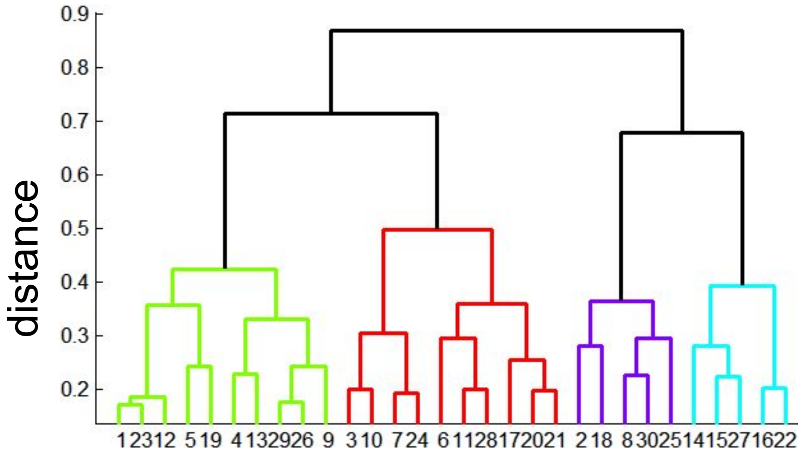
How to define cluster similarity?

- 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 based on distance between merges

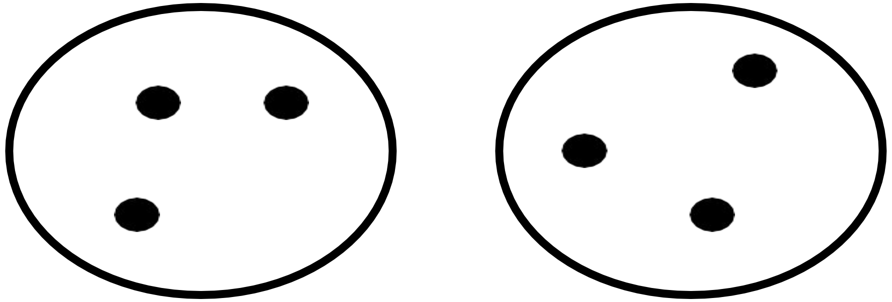


Agglomerative Hierarchical Clustering - Algorithm

Inputs:
- An input image
- Feature representation for each pixel
- Distance metric dist(-,-)

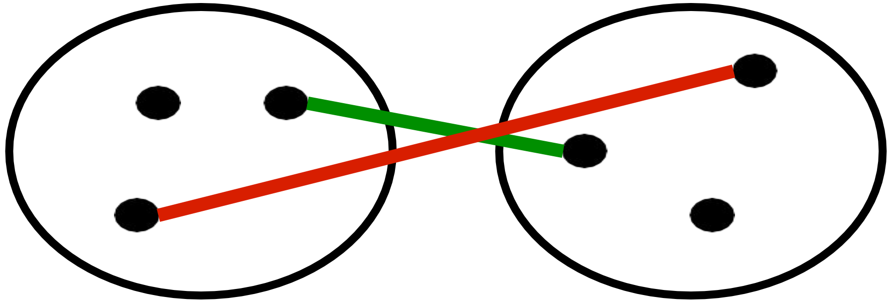
➢ Initially, each pixel v_1, …, v_n is its own cluster C_1, …, C_n
➢ While True:
  - Find two nearest clusters according to dist(C_i, C_j)
  - Merge C = (C_i, C_j)
  - If only 1 cluster is left:
    - break

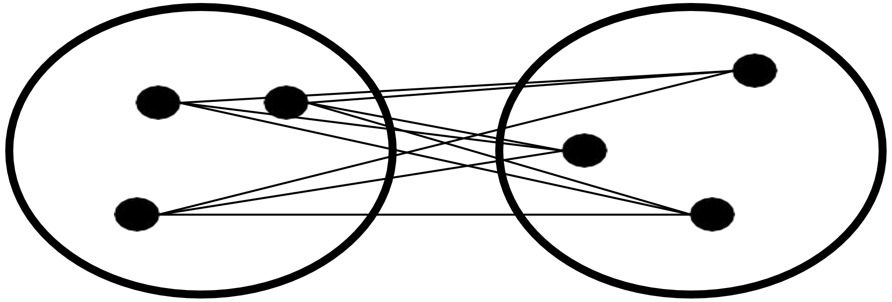
How should we define “closest” for clusters with multiple pixels already in it?



How should we define "closest" for clusters with multiple pixels already in it?

- Closest pair
  (single-link clustering)
- Farthest pair
  (complete-link clustering)
- Average of all pairs

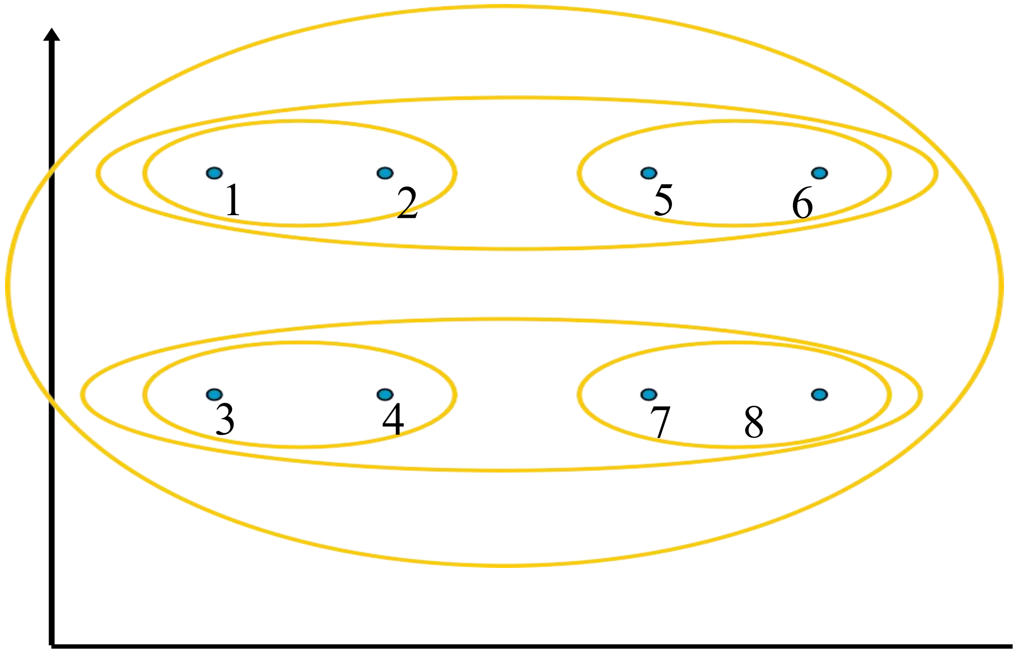




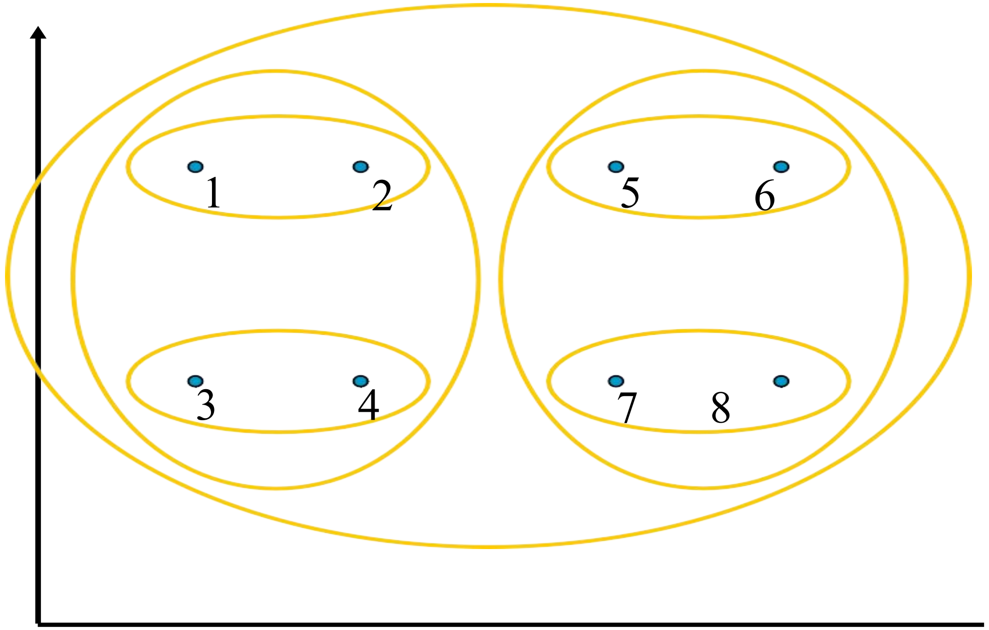
Different choices create different clustering behaviors

How should we define "closest" for clusters with multiple pixels already in it?

Closest pair
(single-link clustering)



Farthest pair
(complete-link clustering)

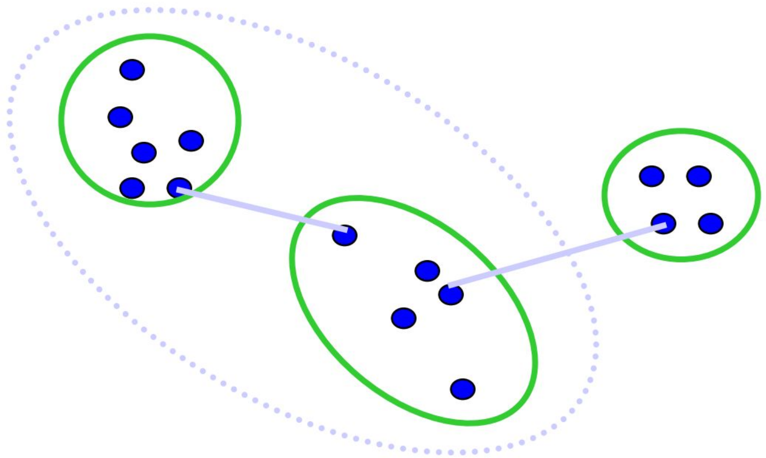


[Pictures from Thorsten Joachims]

Single Linkage distance measure

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

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

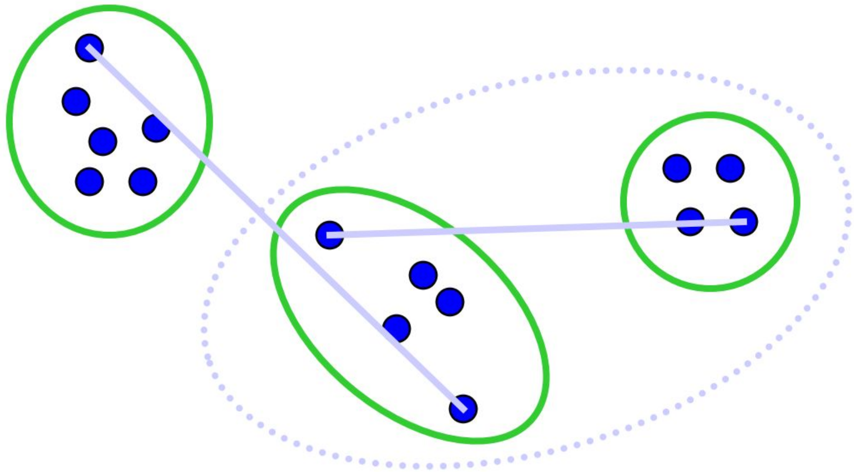


Long, skinny clusters

Complete Link distance measure

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

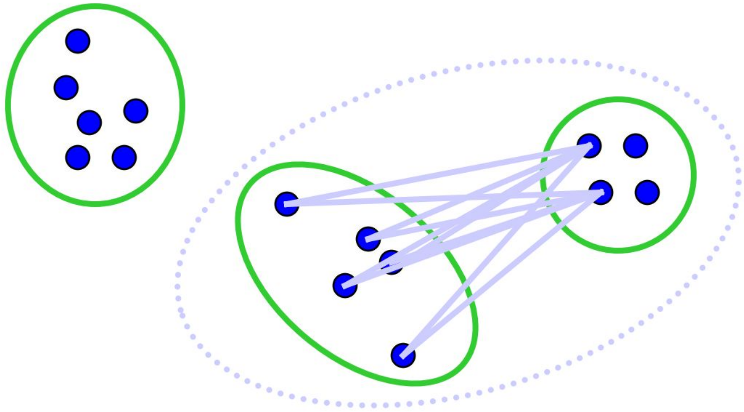
Produces compact clusters that are similar in diameter



Tight clusters

Average Link distance measures

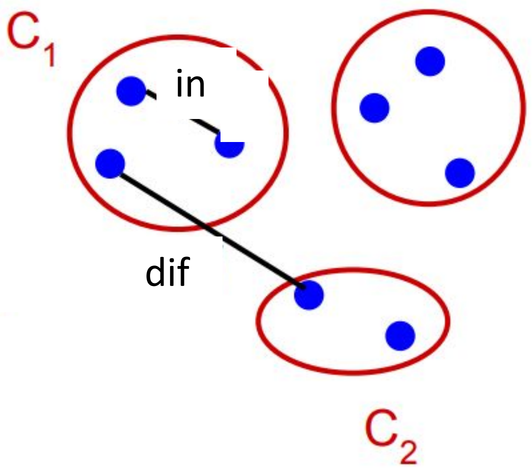
$$dist(C_i, C_j) = \frac{\sum_{v_i \in C_i, v_j \in C_j, (C_i, C_j) \in E} dist(v_i, v_j)}{|C_i||C_j|}$$



Robust against noise.

Inlier-outlier linkage distance measure

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



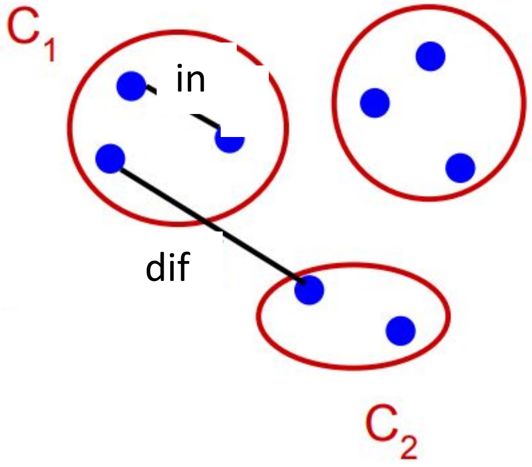
Where

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

Inlier-outlier linkage distance measure

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

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



Where

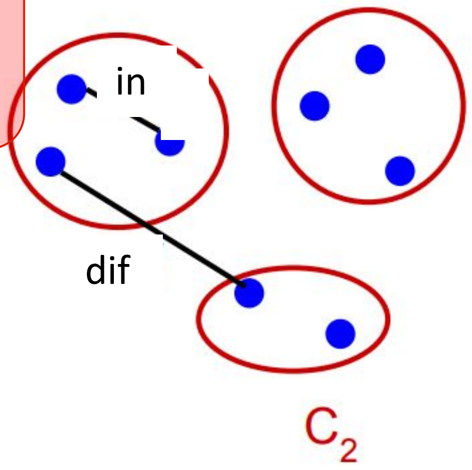
- dif(C1 , C2) is the difference between two clusters.
- in(C1 , C2) is the internal difference in the clusters C1 and C2

Inlier-outlier linkage distance measure

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

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

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



Where

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

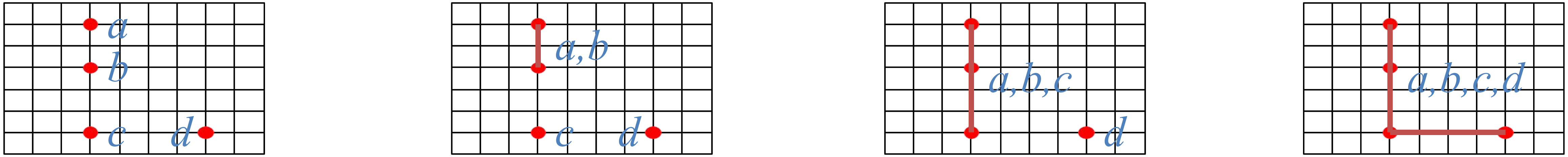
inlier-outlier linkage for Segmentation

- 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

How to implement single-linkage efficiently

Euclidean Distance



(1) (2) (3)

	b	c	d
a	2	5	6
b		3	5
c			4

Distance Matrix

(1)

	b	c	d
a	2	5	6
b		3	5
c			4

(2)

	c	d
a, b	3	5
c		4

(3)

	d
a, b, c	4

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 atleast $O(n^2)$.
- Can get stuck at a local optima.

Next time

K-means and mean shift

Other Kernels

A kernel is a function that satisfies the following requirements :

1. $\int_{R^d} \phi(x) = 1$
2. $\phi(x) \geq 0$

Some examples of kernels include :

1. Rectangular $\phi(x) = \begin{cases} 1 & a \leq x \leq b \\ 0 & else \end{cases}$
2. Gaussian $\phi(x) = e^{-\frac{x^2}{2\sigma^2}}$
3. Epanechnikov $\phi(x) = \begin{cases} \frac{3}{4}(1-x^2) & if\ |x| \leq 1 \\ 0 & else \end{cases}$

source

Technical Details

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

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

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

- Term1: this is proportional to the density estimate at 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 x_t is:

1. Compute the mean shift vector **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

Technical Details

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

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

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

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

where c_k represents a normalization constant.

Comaniciu & Meer, 2002