CS4442/9542b: Artificial Intelligence II
Prof. Olga Veksler

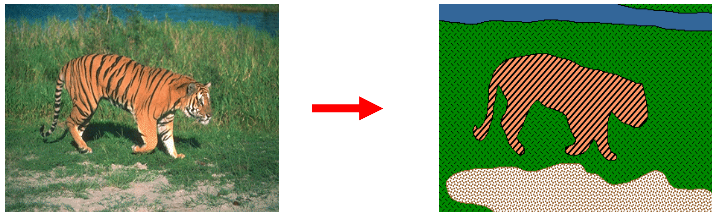
Lecture 15: Computer Vision
Image Segmentation

Slides are from Steve Seitz (UW), David Jacobs (UMD), Octavia Camps, Yaron Ukrainitz, Bernard Sarel

Today

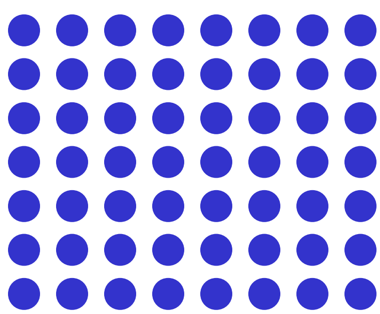
- Perceptual Grouping in humans
  - Gestalt perceptual grouping laws, describe grouping cues of humans
- Image segmentation ("Pixel Grouping")
  - Clustering
    - simple agglomerative algorithm
    - K-means
  - Histogram based
    - Thresholding
    - Mode-finding
    - Mean shift

From Images to Objects



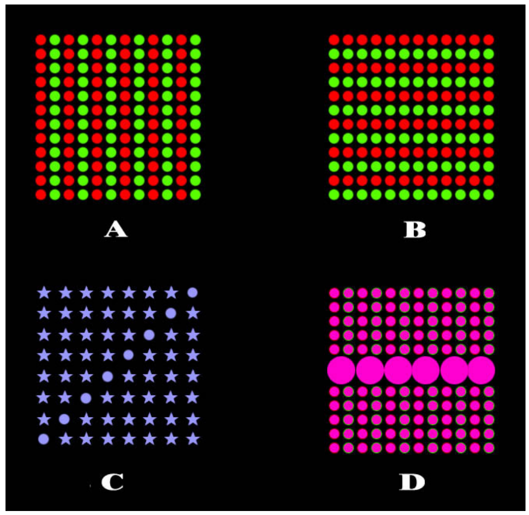
- Humans do not perceive the world as a collection of individual “pixels” but rather as a collection of objects and surfaces
- For many applications, it is useful to **segment** or **group** image pixels into blobs which are perceptually meaningful
  - hopefully belong to the same “object” or surface
- How to do this without (necessarily) object recognition?
  - Subjective problem, but has been well-studied
  - Gestalt Laws seek to formalize this
    - proximity, similarity, continuation, closure, common fate

Grouping



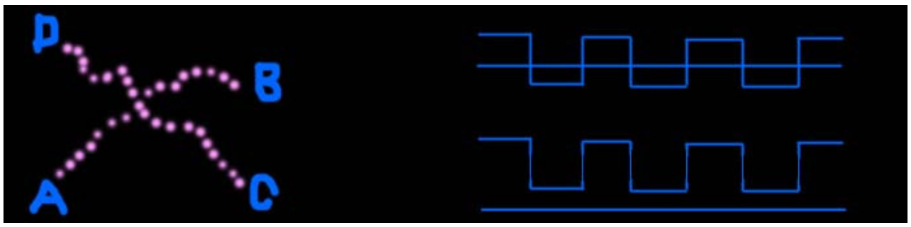
Most human observers would report no particular grouping

Gestalt Principles of Grouping: Common Form (includes color and texture)

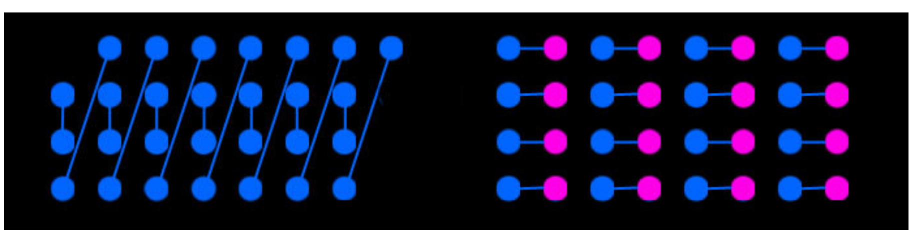


Gestalt Principles of Grouping: Proximity

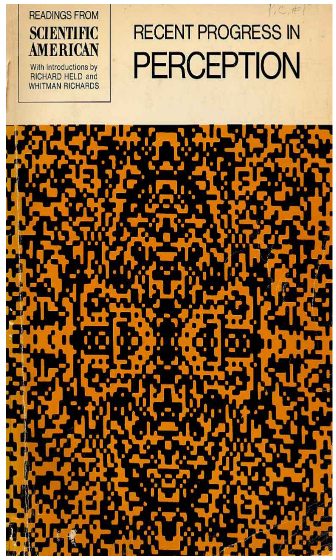
Gestalt Principles of Grouping: Good Continuation



Gestalt Principles of Grouping: Connectivity



Gestalt Principles of Grouping: Symmetry



Gestalt Principles of Grouping: Symmetry

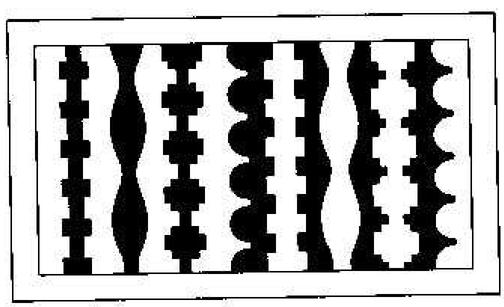
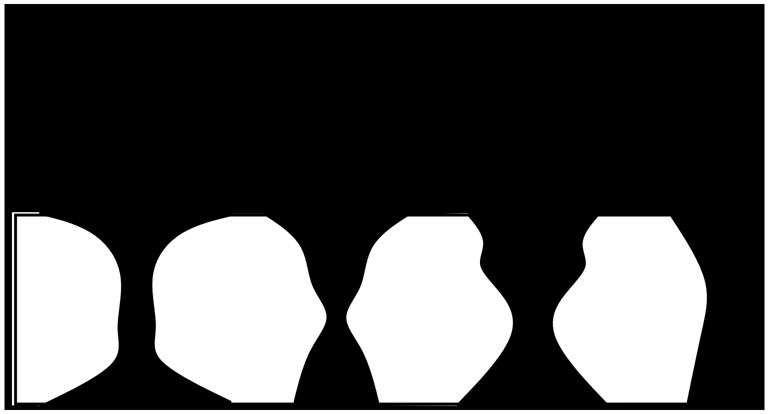


Figure 7.25

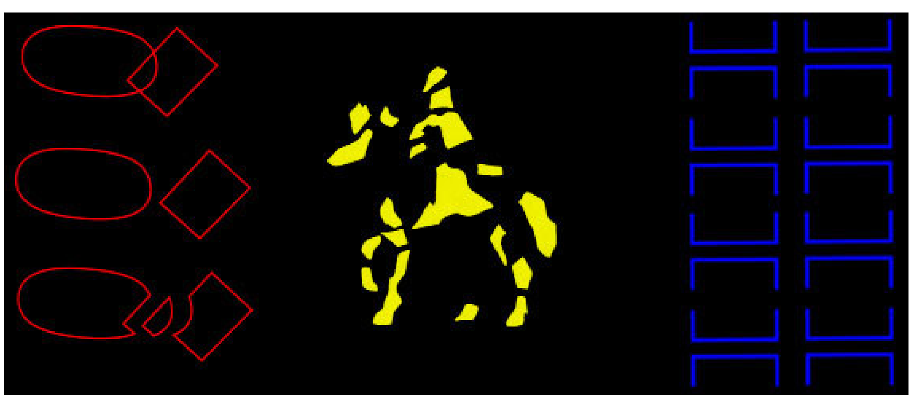
Symmetry and figure ground. Look to the left and to the right, and observe which colors become figure and which become ground. (Adapted from Hochberg, 1971.)

Gestalt Principles of Grouping: Convexity

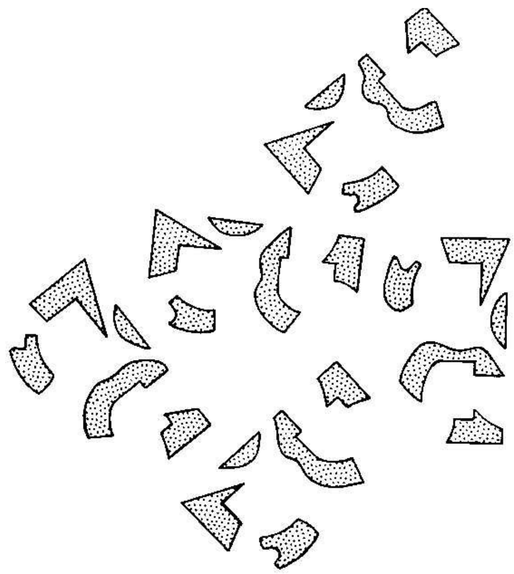


stronger than symmetry?

Gestalt Principles of Grouping: Closure

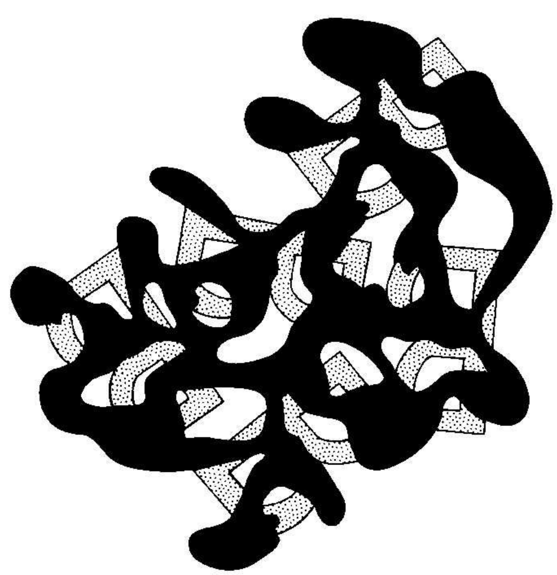


Gestalt Principles of Grouping: Closure



(Bregman)

Gestalt Principles of Grouping: Closure



Gestalt Principles of Grouping:Common Motion

Higher level Knowledge

Other Perceptual Grouping Factors

- Common depth
- Parallelism
- Collinearity

Take Home message

- We perceive the world in terms of objects, not pixels
- What forms an object is determined by regularities and non-trivial inference

Human perceptual grouping

- Perceptual grouping has been significant inspiration to computer vision
- Why?
  - Perceptual grouping seems to rely partly on the nature of objects in the world
  - This is hard to quantify, we hypothesize that human vision encodes the necessary knowledge

Computer Vision: Image Segmentation

- In vision, we typically refer to perceptual organization problem as **image segmentation** or **clustering**
- Image segmentation is the operation of partitioning an image into a collection of
  - regions, which usually cover the whole image
  - linear structures, such as
    - line segments
    - curve segments
  - into 2D shapes, such as
    - circles
    - ellipses
    - ribbons (long, symmetric regions)
- Clustering is a more general term than image segmentation
  - Can cluster all sorts of data (usually represented as feature vectors), not just image pixels
    - Web pages, financial records, etc.
  - Clustering is a large area of machine learning (not supervised, that is labels of feature vectors are not known)

Example 1: Region Segmentation

Example 2: Lines and Circular Arcs Segmentation

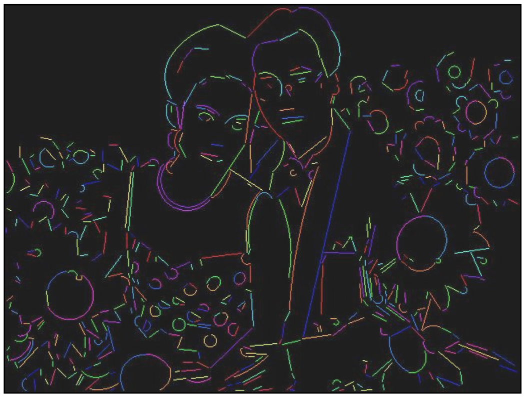


Image Segmentation: Cues for Grouping

Image cues are used for grouping/segmentation:

- Pixel-based cues:
  - color
  - depth (for stereo pairs)
  - motion (for video sequences)
- Region-based cues:
  - texture
  - region shape
- contour-based cues:
  - curvature

Image Segmentation Approaches

Approaches can be roughly divided into two groups:

1. Parametric: We have a description of what we want, with parameters:

   Examples: lines, circles, constant intensity regions, constant intensity regions + Gaussian noise

2. Non-parametric: have constraints the group should satisfy, or optimality criteria.

   Example: SNAKES. Find the closed curve that is smoothest and that also best follows strong image gradients.

Clustering Algorithms

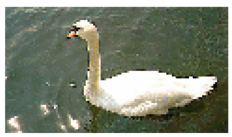
- Agglomerative
  - Start with each pixel in its own cluster
  - Iteratively merge clusters together according to some pre-defined criterion
  - Stop when reached some stopping condition
- Divisive
  - Start with all pixels in one cluster
  - Iteratively choose and split a cluster into two according to some pre-defined criterion
  - Stop when reached some stopping condition
- There are clustering methods which are both agglomerative and divisive

Simplest Agglomerative Clustering based on Color/Intensity

Initialize: Each pixel is a cluster (region)

Loop

- Find two adjacent regions with most similar color (or intensity)
- Merge to form new region with:
  - all pixels of these regions
  - average color (or intensity) of these regions
- Several possibilities for stopping condition:
  1. No regions similar (color or intensity differences between all neighboring regions is larger than some threshold, etc.)



Example: Agglomerative Intensity Based Clustering

23	25	19	21	23
18	22	24	25	24
20	19	26	28	22
3	3	7	8	26
1	3	5	4	24

23	25	19	21	23
18	22	24	25	24
20	19	26	28	22
3	3	7	8	26
1	3	5	4	24

Example: Agglomerative Intensity Based Clustering

23	25	19	21	23
18	22	24	25	24
20	19	26	28	22
3	3	7	8	26
1	3	5	4	24

→

23	25	19	21	23
18	22	24	25	24
20	19	26	28	22
3	3	7	8	26
1	3	5	4	24

Example: Agglomerative Intensity Based Clustering

23	25	19	21	23
18	22	24	25	24
20	19	26	28	22
3	3	7	8	26
1	3	5	4	24

→

23	25	19	21	23
18	22	24.5	24.5	24
20	19	26	28	22
3	3	7	8	26
1	3	5	4	24

Example: Agglomerative Intensity Based Clustering

23	25	19	21	23
18	22	24.5	24.5	24
20	19	26	28	22
3	3	7	8	26
1	3	5	4	24

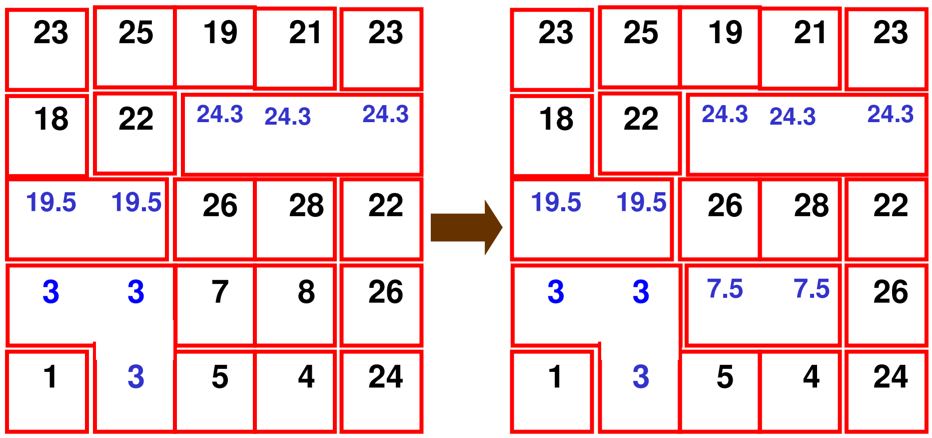
23	25	19	21	23
18	22	24.3	24.3	24.3
20	19	26	28	22
3	3	7	8	26
1	3	5	4	24

Example: Agglomerative Intensity Based Clustering

23	25	19	21	23
18	22	24.3	24.3	24.3
20	19	26	28	22
3	3	7	8	26
1	3	5	4	24

23	25	19	21	23
18	22	24.3	24.3	24.3
19.5	19.5	26	28	22
3	3	7	8	26
1	3	5	4	24

Example: Agglomerative Intensity Based Clustering



Example: Agglomerative Intensity Based Clustering

. . .

Example: Agglomerative Intensity Based Clustering

23	25	19	21	23
18	22	24	25	24
20	19	26	28	22
3	3	7	8	26
1	3	5	4	24

22.9	22.9	22.9	22.9	22.9
22.9	22.9	22.9	22.9	22.9
22.9	22.9	22.9	22.9	22.9
4.25	4.25	4.25	4.25	22.9
4.25	4.25	4.25	4.25	22.9

Clustering complexity

- Assume image has n pixels
- Initializing:
  - O(n) time to compute regions
- Loop:
  - O(n) time to find 2 neighboring regions with most similar colors (could speed up)
  - O(n) time to update distance to all neighbors
- At most n times through loop so $O(n^2)$ time total

Agglomerative Clustering: Discussion

- Start with definition of good clusters
- Simple initialization
- Greedy: take steps that seem to most improve clustering
- This is a very general, reasonable strategy
- Can be applied to almost any problem
- But, not guaranteed to produce good quality answer

Clustering for Image Segmentation

- General clustering problem setting:
- have samples (or points, or feature vectors) $\boldsymbol{x_1},\ldots,\mathbf{x_n}$
  - **for segmentation, $x_1,\ldots,x_n$, correspond to n image pixels**
  - **each x_i can be**
    - **Intensity of pixel x_i (for gray image segmentation)**
    - **Color of pixel x_i (for color image segmentation)**
    - **Color of pixel x_i + coordinates of pixel x_i**
  - **For example:**

(2,44,55)	(22,4,5)	(32,5,6)
(4,4,25)	(6,14,6)	(7,8,91)

feature vectors for color based clustering

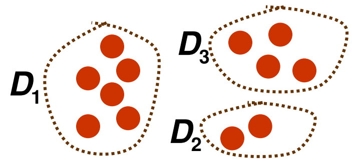
[2,44,55]
[22,4,5]
[32,5,6]
[4,4,25]
[6,14,6]
[7,8,91]

feature vectors for color and coordinates based clustering

[2,44,55,0,0]
[22,4,5,1,0]
[32,5,6,2,0]
[4,4,25,0,1]
[6,14,6,1,1]
[7,8,91,2,1]

Criterion Functions for Clustering

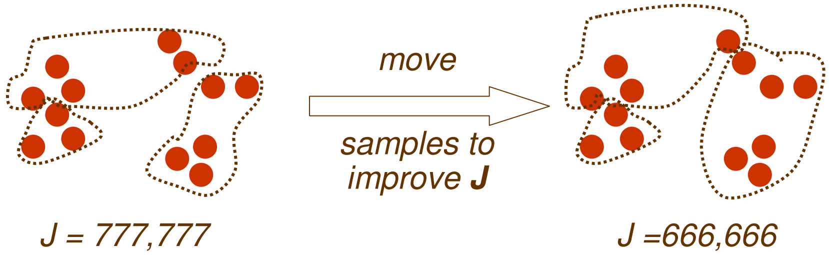
- Have samples (or points) $x_1,\dots,x_n$
- Suppose partitioned samples into k subsets $D_1,\dots,D_k$



- There are approximately $k^n/k!$ distinct partitions
- Can define a criterion function $J(D_1,\dots,D_k)$ which measures the quality of a partitioning $D_1,\dots,D_k$
- Then the clustering problem is a well defined problem
  - the optimal clustering is the partition which optimizes the criterion function

Iterative Optimization Algorithms

- Now have both proximity measure and criterion function, need algorithm to find the optimal clustering
- Exhaustive search is impossible, since there are approximately $k^n/k!$ possible partitions
- Usually some iterative algorithm is used
  - Find a reasonable initial partition
  - Repeat: *move samples from one group to another s.t. the objective function **J** is improved*



K-means Clustering

- Iterative clustering algorithm
- Want to optimize the J_{SSE} objective function

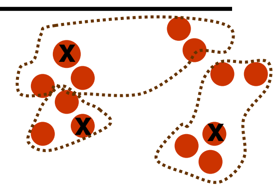
$$J_{SSE} = \sum_{i=1}^{k} \sum_{x \in D_i} || x - \mu_i ||^2$$

  - for a different objective function, we need a different optimization algorithm, of course
- Fix number of clusters to k
- k-means is probably the most famous clustering algorithm
  - it has a smart way of moving from current partitioning to the next one

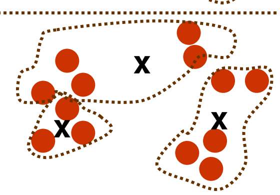
K-means Clustering

k = 3

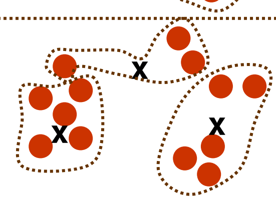
1. Initialize
   - pick k cluster centers arbitrary
   - assign each example to closest center



2. compute sample means for each cluster



3. reassign all samples to the closest mean

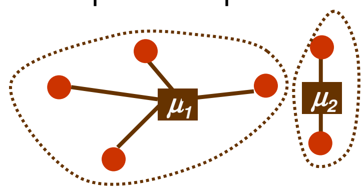


4. if clusters changed at step 3, go to step 2

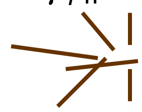
K-means Clustering

- Consider steps *2* and *3* of the algorithm

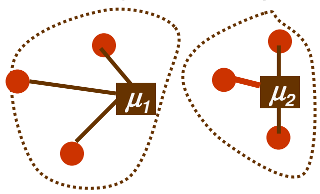
2. compute sample means for each cluster



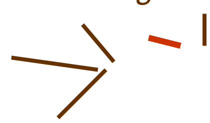
$$J_{SSE} = \sum_{i=1}^{k}\sum_{x \in D_i} || x - \mu_i ||^2$$

= sum of 

3. reassign all samples to the closest mean

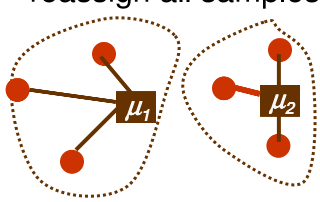


If we represent clusters by their old means, the error has gotten smaller



K-means Clustering

3. reassign all samples to the closest mean



If we represent clusters by their old means, the error has gotten smaller

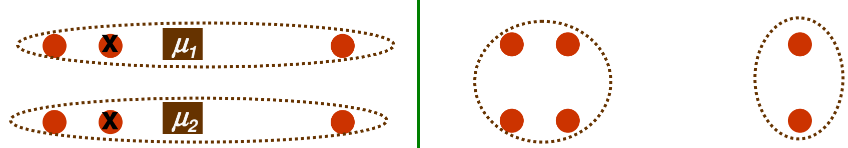
- However we represent clusters by their new means, and mean is always the smallest representation of a cluster

$$\frac{\partial}{\partial z}\sum_{x \in D_i}\frac{1}{2}|| x - z ||^2 = \frac{\partial}{\partial z}\sum_{x \in D_i}\frac{1}{2}\left(|| x ||^2 - 2x^t z + || z ||^2\right) = \sum_{x \in D_i}(-x + z) = 0$$

$$\Rightarrow z = \frac{1}{n_i}\sum_{x \in D_i} x$$

K-means Clustering

- We just proved that by doing steps *2* and *3*, the objective function goes down
  - in two step, we found a “smart “ move which decreases the objective function
- Thus the algorithm converges after a finite number of iterations of steps *2* and *3*
- However the algorithm is not guaranteed to find a global minimum



2-means gets stuck here

global minimum of $\boldsymbol{J_{SSE}}$

K-means clustering Example

- k = 2 and initial cluster centers are at pixels (0,0) and (1,2)

feature vectors for color based clustering

[2,4,5]
[6,8,8]
[3,5,6]
[7,9,5]
[1,4,6]
[7,8,9]

(2,4,5)	(6,8,8)	(3,5,6)
(7,9,5)	(1,4,6)	(7,8,9)

- distance between (6,8,8) and (2,4,5) is

$$(6-2)^2 + (8-4)^2 + (8-5)^2 = 41$$

- distance between (6,8,8) and (7,8,9) is

$$(6-7)^2 + (8-8)^2 + (8-9)^2 = 2$$

- Therefore sample (6,8,8) is assigned to the same cluster as (7,8,9)
- Repeat for the other 5 samples

K-means clustering Example

- k = 2 and initial cluster centers are at pixels (0,0) and (1,2)

feature vectors for color based clustering

[2,4,5]
[6,8,8]
[3,5,6]
[7,9,5]
[1,4,6]
[7,8,9]

(2,4,5)	(6,8,8)	(3,5,6)
(7,9,5)	(1,4,6)	(7,8,9)

after 1st iteration:

(2,4,5)	(6,8,8)	(3,5,6)
(7,9,5)	(1,4,6)	(7,8,9)

after 1st iteration new means are:

$$\frac{(2,4,5)+(1,4,6)+(3,5,6)}{3}=(2,4.33,5.66)$$

$$\frac{(7,9,5)+(6,8,8)+(7,8,9)}{3}=(6.66,8.33,7.33)$$

K-means clustering Example

- k = 2 and initial cluster centers are at pixels (0,0) and (1,2)

after 1st iteration:

(2,4,5)	(6,8,8)	(3,5,6)
(7,9,5)	(1,4,6)	(7,8,9)

after 1st iteration new means are:

$$\frac{(2,4,5)+(1,4,6)+(3,5,6)}{3}=(2,4.33,5.66)$$

$$\frac{(7,9,5)+(6,8,8)+(7,8,9)}{3}=(6.66,8.33,7.33)$$

- samples (2,4,5), (1,4,6), (7,8,9) are closest to mean (2,4.33,5.66)
- samples (7,9,5), (6,8,8) and (7,8,9) are closest to the mean (6.66,8.33,7.33)
- Therefore no change after second iteration, k-means converges

K-means clustering Example

- k = 2 and initial cluster centers are at pixels (0,0) and (1,2)

(2,4,5)	(6,8,8)	(3,5,6)
(7,9,5)	(1,4,6)	(7,8,9)

feature vectors for color and coordinates based clustering

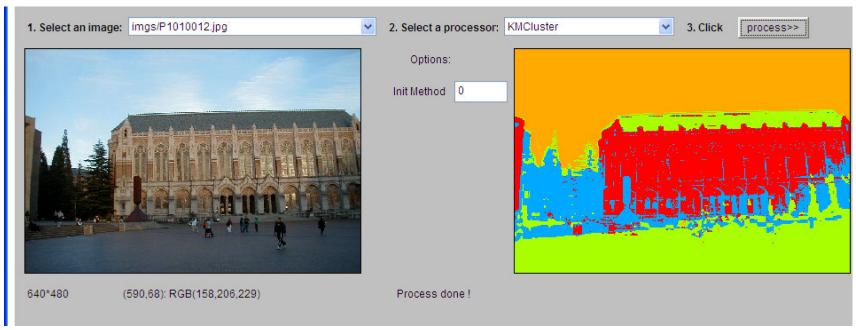
[2,4,5,0,0]
[6,8,8,1,0]
[3,5,6,2,0]
[7,9,5,0,1]
[1,4,6,1,1]
[7,8,9,2,1]

- The procedure is identical to the color-only based clustering, except samples are 5-dimensional now

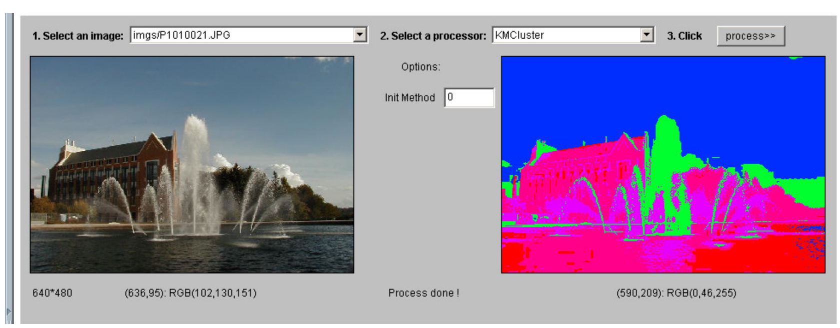
K-means Clustering

- Finding the optimum of $\boldsymbol{J_{SSE}}$ is NP-hard
- In practice, **k**-means clustering performs usually well
- It is very efficient
- Its solution can be used as a starting point for other clustering algorithms
- Still 100's of papers on variants and improvements of **k**-means clustering every year

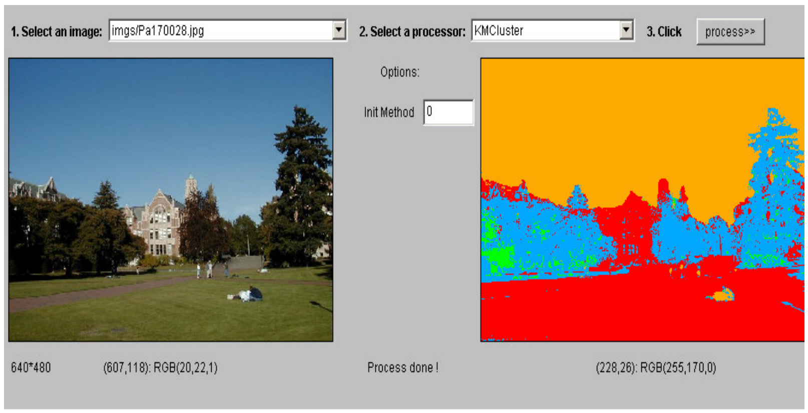
K-Means Example 1



K-Means Example 2



K-Means Example 3



Histogram-Based Segmentation

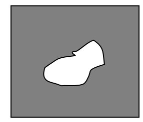
Segmentation by Histogram Processing

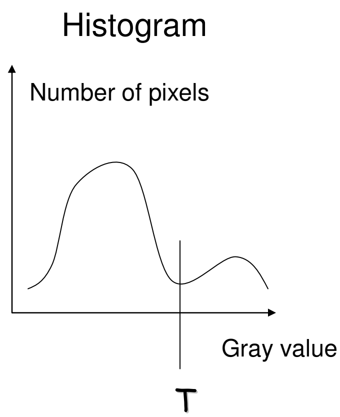
- Given image with N colors, choose K
- Each of the K colors defines a region
  – not necessarily contiguous
- Performed by computing color histogram, looking for modes

- This is what happens when you downsample image color range, for instance in Photoshop

Histogram-based Segmentation

Ex: bright object on dark background:





Select threshold

Create binary image:

- $I(x,y) < T \Rightarrow O(x,y) = 0$
- $I(x,y) > T \Rightarrow O(x,y) = 1$

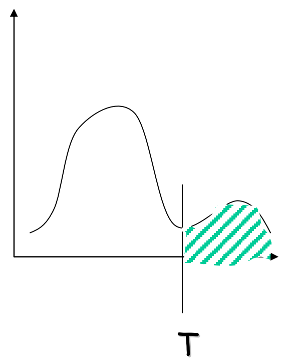
How do we select a Threshold?

Automatic thresholding

- P-tile method
- Mode method
- Peakiness detection
- Mean-shift

P-Tile Method

If the *size* and brightness range of the object is approximately known, pick T s.t. the area under the histogram corresponds to the size of the object:

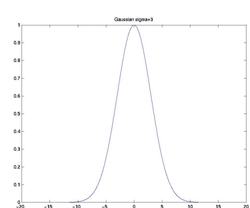


Mode Method

- Model each region as “constant” + noise
- Usually noise is modeled as $N(0,\sigma_i)$:

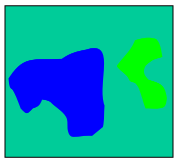
If $(x,y) \in R_i$ then, $I(x,y) = \mu_i + n_i(x,y)$

$$p(n_i) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{1}{2}\frac{n_i^2}{\sigma_i^2}}$$

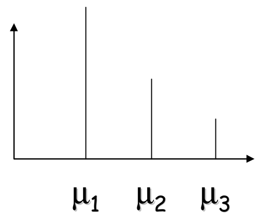


$$E(n_i) = 0 \qquad E(n_i^2) = \sigma_i^2$$

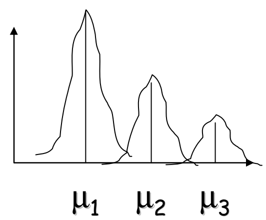
Example: Image with 3 regions



Ideal histogram:

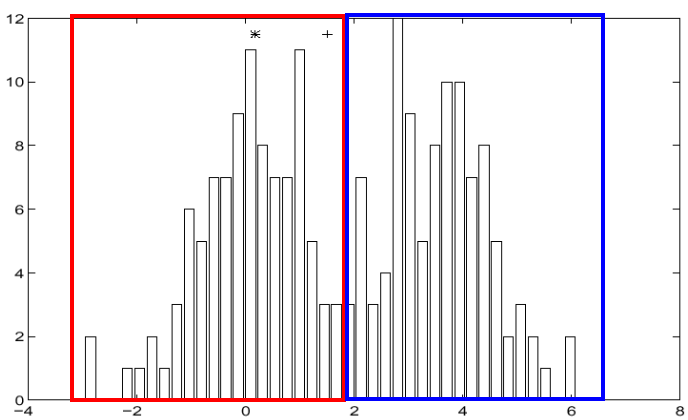


Add noise:



The valleys are good places for thresholding to separate regions.

Finding Modes in a Histogram



How Many Modes Are There?

- Easy to see, hard to compute
- Not a trivial problem

“Peakiness” Detection Algorithm

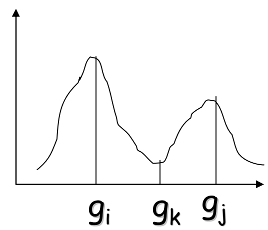
Find the two **HIGHEST LOCAL MAXIMA** at a **MINIMUM DISTANCE APART**: g_i and g_j

Find **lowest point** between them: g_k

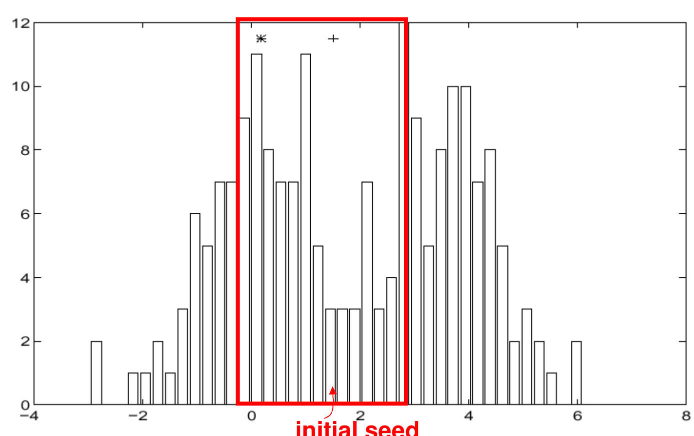
Measure “peakiness”:

- $\min(H(g_i),H(g_j))/H(g_k)$

Find (g_i,g_j,g_k) with highest peakiness



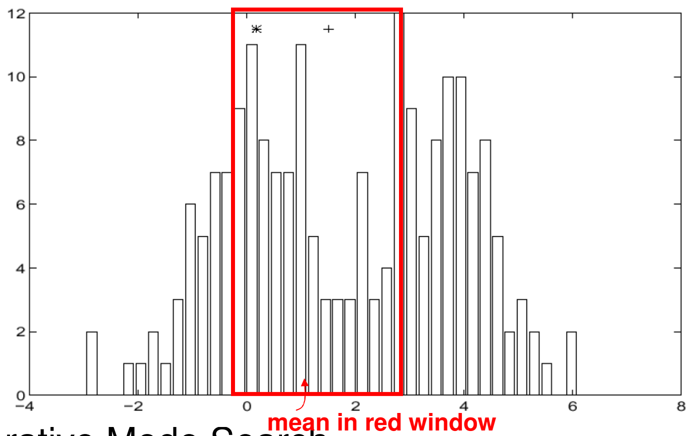
Mean Shift [Comaniciu & Meer]



Iterative Mode Search

1. Initialize random seed, and fixed window
2. Calculate center of gravity of the window (the “mean”)
3. Translate the search window to the mean
4. Repeat Step 2 until convergence

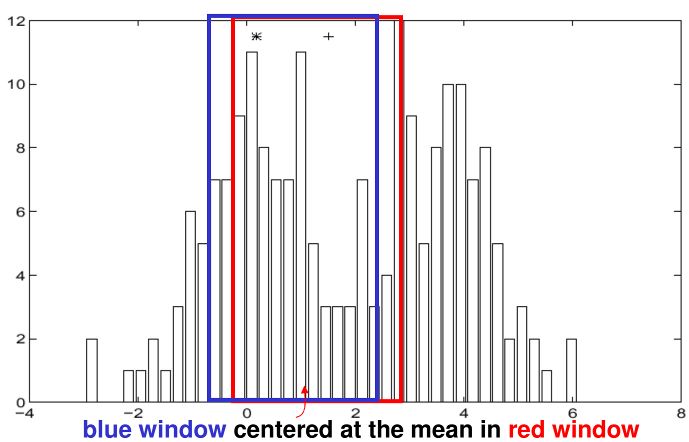
Mean Shift [Comaniciu & Meer]



Iterative Mode Search

1. Initialize random seed, and fixed window
2. Calculate center of gravity of the window (the "mean")
3. Translate the search window to the mean
4. Repeat Step 2 until convergence

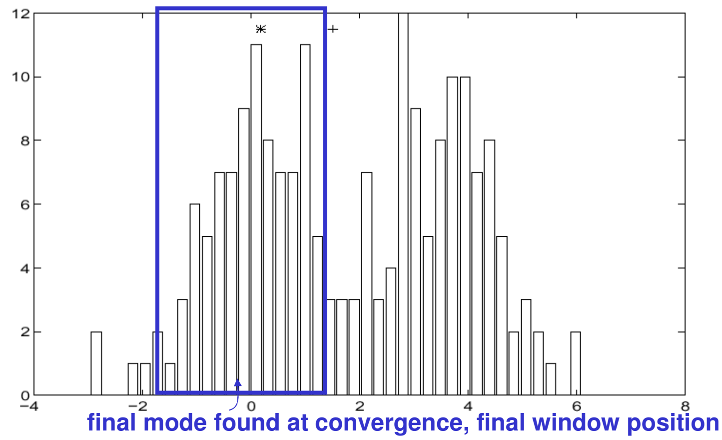
Mean Shift [Comaniciu & Meer]



Iterative Mode Search

1. Initialize random seed, and fixed window
2. Calculate center of gravity of the window (the "mean")
3. Translate the search window to the mean
4. Repeat Step 2 until convergence

Mean Shift [Comaniciu & Meer]



Iterative Mode Search

1. Initialize random seed, and fixed window
2. Calculate center of gravity of the window (the "mean")
3. Translate the search window to the mean
4. Repeat Step 2 until convergence

Algorithm MEAN SHIFT to find histogram PEAK

1. Choose a window size
   - for example 5
2. Choose the initial location of the search window
3. Compute the mean location in the search window
4. Center the window at the location computed in 3
5. Repeat steps 3 and 4 until convergence

Algorithm MEAN SHIFT for Image Segmentation

- Find image histogram, choose window size
- Choose initial location of search window:
  - Randomly select a number M of image pixels
  - Find the average value in a 3x3 window for each of these pixels
  - Set the center of the window to the value with largest histogram count
- Apply mean shift to find the window peak
- Remove pixels in the window from the image and the histogram
  - Say peak was at intensity 44 and window size is 5
  - Pixels with intensities between [39,49] become one group
  - Remove these pixels from further consideration
- Repeat steps 2 to 4 until no pixels are left

Algorithm MEAN SHIFT

- Previous slides assumed features are gray pixel values
  - Feature vectors are one dimensional
- Can do the same thing for color images
  - Feature vectors are 3 dimensional
- Can also include the (x,y) pixel coordinates
  - Feature vectors are 5 dimensional
- In all these cases, taking a window around feature vector ***y*** corresponds to taking all feature vectors ***x*** s.t.

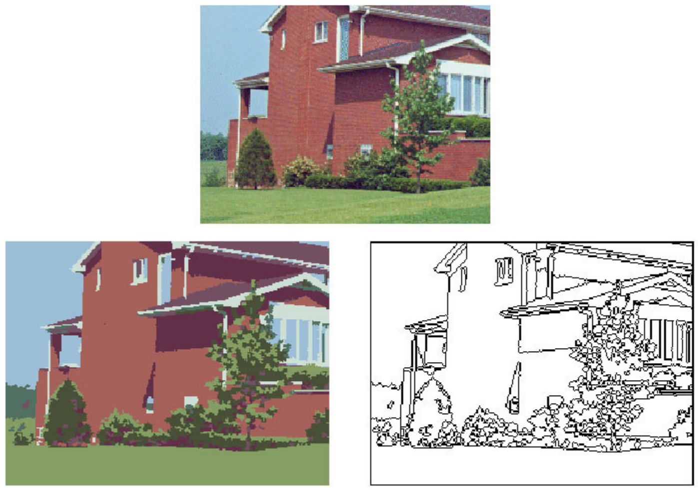
$$\|y - x\|^2 \leq r$$

- New window center is shifted from ***y*** to

$$\frac{1}{n}\sum_{x \in S} x$$

- Where S is the set of all feature vectors x s.t. $\|y - x\|^2 \leq r$, and n is the size of S

Mean Shift Segmentation: Examples



More Examples: http://www.caip.rutgers.edu/~comanici/segm_images.html

Mean Shift Segmentation: More Examples

Mean Shift Segmentation: More Examples

Mean Shift: Strengths & Weaknesses

Strengths :

- Does not assume any prior shape (e.g. elliptical) on data clusters
- Can handle arbitrary feature spaces
- Only ONE parameter to choose
    - *h* the window size

Weaknesses :

- The window size is not trivial
- Inappropriate window size can cause modes to be merged (giving too few segments) or generate additional “shallow” modes (giving too many segments)
- there are adaptive window size extentions