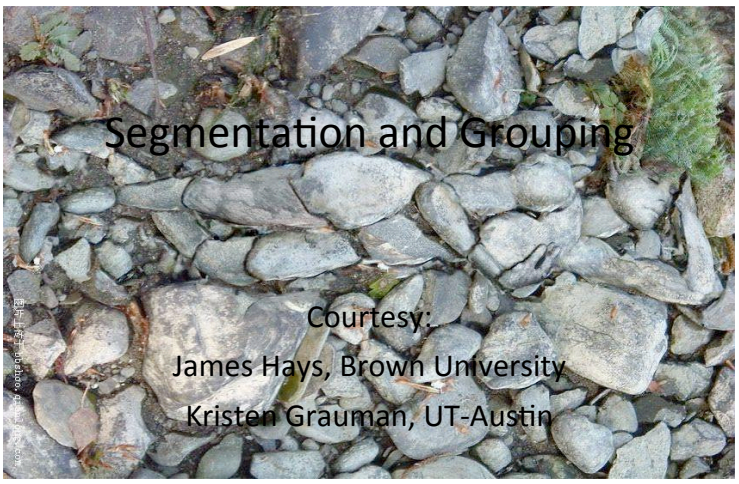


Slides: Hoiem and others

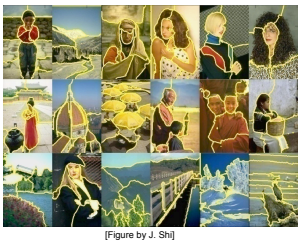
Outline

- What are grouping problems in vision?
- Inspiration from human perception
  - Gestalt properties
- Bottom-up segmentation via clustering
  - Algorithms:
    - Mode finding and mean shift: k-means, mean-shift
    - Graph-based: normalized cuts

Grouping in vision

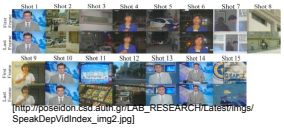
- Goals:
  - Gather features that belong together
  - Obtain an intermediate representation that compactly describes key image or video parts

Examples of grouping in vision



[Figure by J. Shi]

Determine image regions

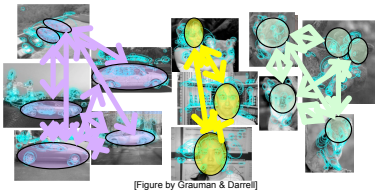


[http://poseidon.csd.auth.gr/LAB_RESEARCH/Latest/Imgs/SpeakDepVidIndex_img2.jpg]

Group video frames into shots

[Figure by Wang & Suter]

Figure-ground



[Figure by Grauman & Darrell]

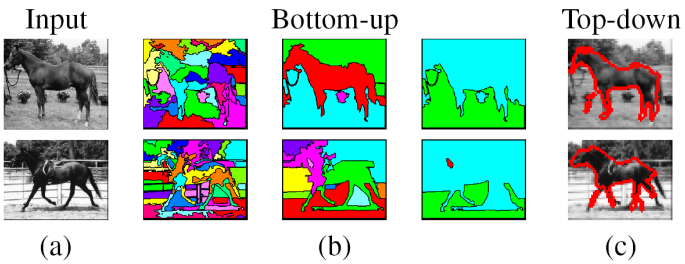
Object-level grouping

Grouping in vision

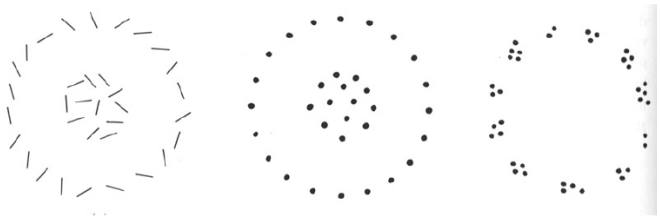
- Goals:
  - Gather features that belong together
  - Obtain an intermediate representation that compactly describes key image (video) parts
- Top down vs. bottom up segmentation
  - Top down: pixels belong together because they are from the same object
  - Bottom up: pixels belong together because they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

Major processes for segmentation

- Bottom-up: group tokens with similar features
- Top-down: group tokens that likely belong to the same object



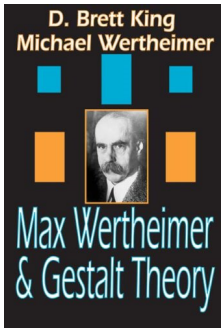
[Levin and Weiss 2006]



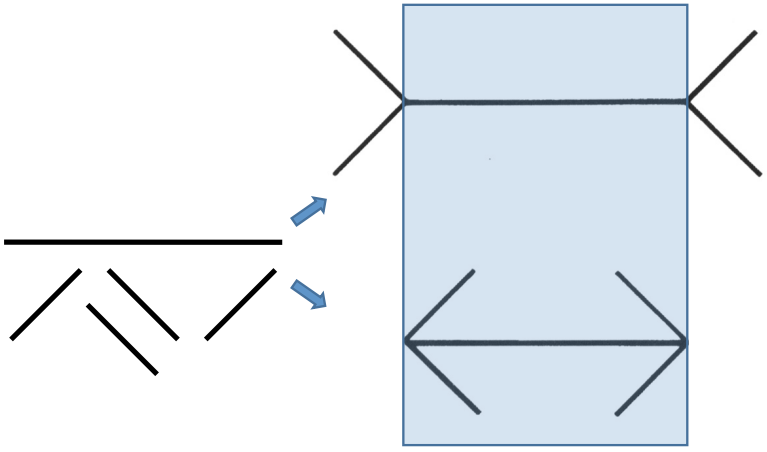
What things should be grouped?
What cues indicate groups?

Gestalt psychology or **Gestaltism**

- German: *Gestalt* - "form" or "whole"
- Berlin School, early 20th century
  - Kurt Koffka, Max Wertheimer, and Wolfgang Köhler
- 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)



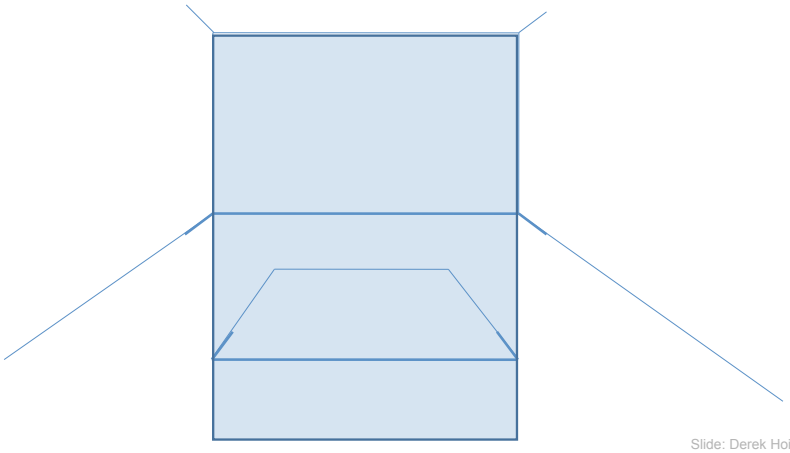
Gestaltism



The Muller-Lyer illusion

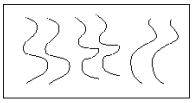
Slide: Derek Hoiem

We perceive the interpretation, not the senses

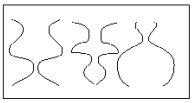


Slide: Derek Hoiem

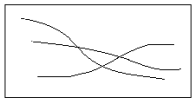
Principles of perceptual organization



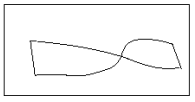
Parallelism



Symmetry



Continuity

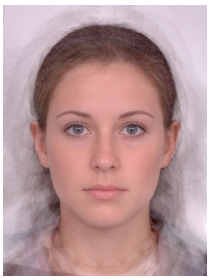


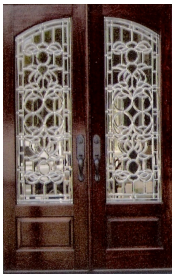
Closure

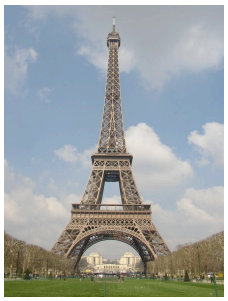
Similarity

http://chicagoist.com/attachments/chicagoist_alicia/GEESE.jpg, http://wwwdelivery.superstock.com/WI/223/1532/PreviewComp/SuperStock_1532R-0831.jpg

Symmetry







http://seedmagazine.com/news/2006/10/beauty_is_in_the_processingtim.php

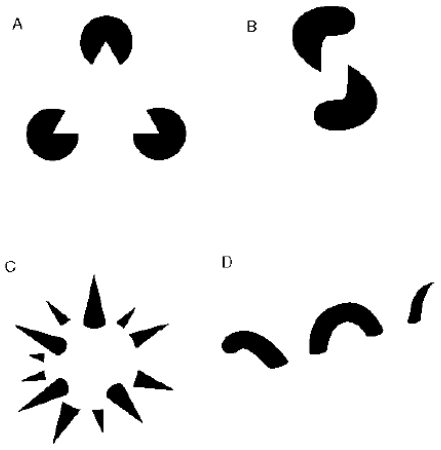
Common fate

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

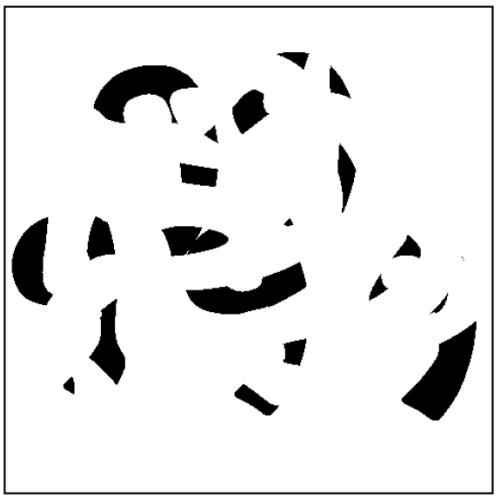
Proximity

http://www.capital.edu/Resources/Images/outside6_035.jpg

Grouping by invisible completion

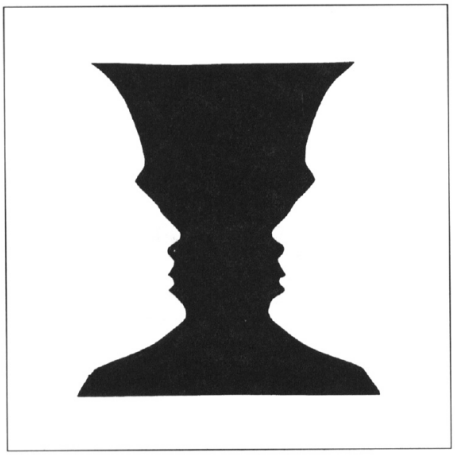


From Steve Lehar: The Constructive Aspect of Visual Perception



D. Forsyth

Figure-ground



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

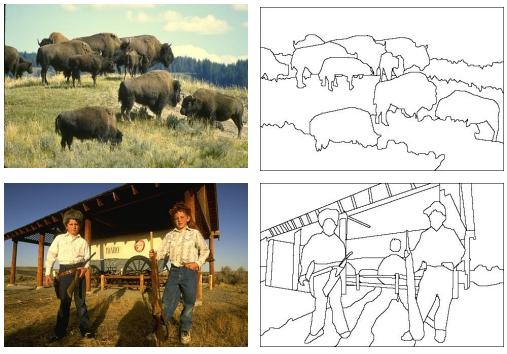
Outline

- What are grouping problems in vision?
- Inspiration from human perception
  - Gestalt properties
- Bottom-up segmentation via clustering
  - Algorithms:
    - Mode finding and mean shift: k-means, mean-shift
    - Graph-based: normalized cuts
  - Features: color, texture, …
    - Quantization for texture summaries

The goals of segmentation

Separate image into coherent “objects”

image | human segmentation



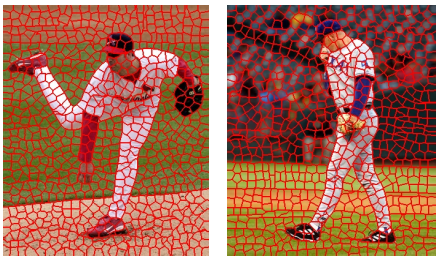
Source: Lana Lazebnik

The goals of segmentation

Separate image into coherent “objects”

Group together similar-looking pixels for efficiency of further processing

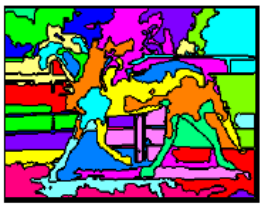
“superpixels”



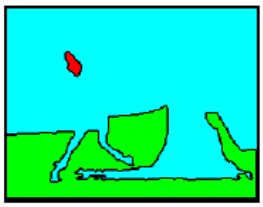
X. Ren and J. Malik. Learning a classification model for segmentation. ICCV 2003.

Source: Lana Lazebnik

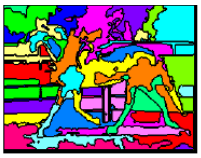
Types of segmentations

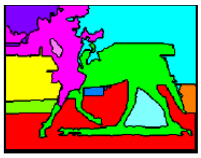


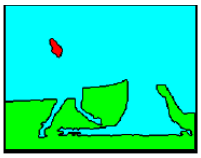
Oversegmentation



Undersegmentation

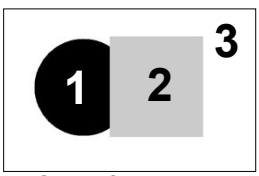




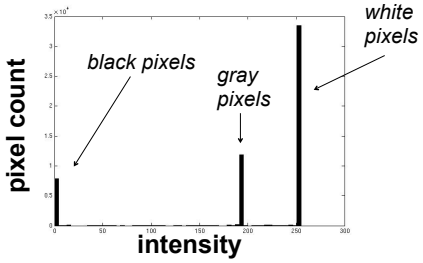


Multiple Segmentations

Image segmentation: toy example

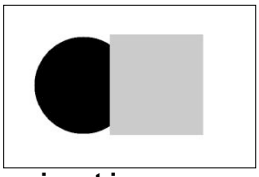


input image

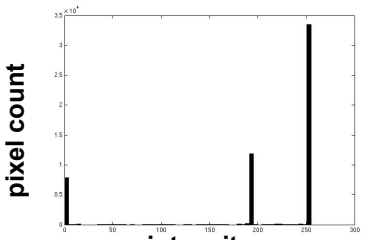


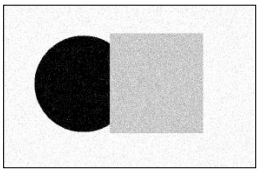
- These intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., *segment* the image based on the intensity feature.
- What if the image isn’t quite so simple?

Kristen Grauman

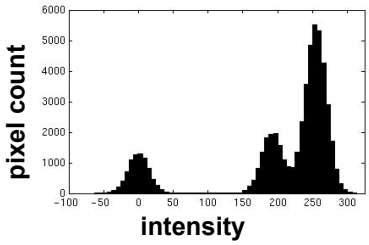


input image

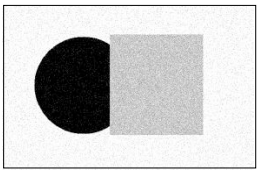




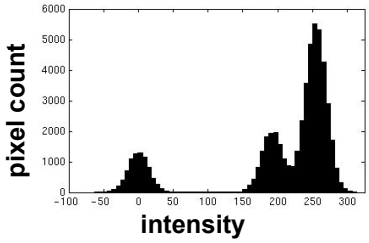
input image



Kristen Grauman

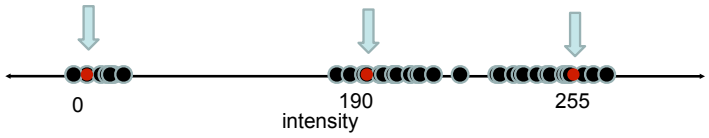


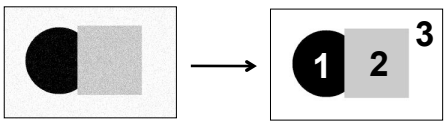
input image



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

Kristen Grauman





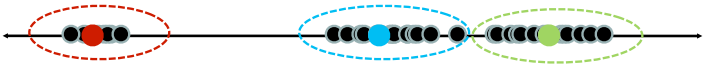
- Goal: choose three "centers" as the representative intensities, and label every pixel according to which of these centers it is nearest to.
- Best cluster centers are those that minimize SSD between all points and their nearest cluster center c_i:

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

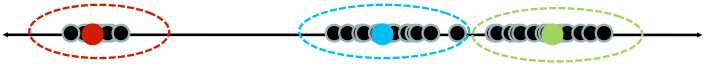
Kristen Grauman

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.



Kristen Grauman

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

Key Challenges:

1) What makes two points/images/patches similar?

2) How do we compute an overall grouping from pairwise similarities?

Slide: Derek Hoiem

How do we cluster?

- K-means
  - Iteratively re-assign points to the nearest cluster center
- Agglomerative clustering
  - Start with each point as its own cluster and iteratively merge the closest clusters
- Mean-shift clustering
  - Estimate modes of pdf
- Spectral clustering
  - Split the nodes in a graph based on assigned links with similarity weights

Clustering for Summarization

Goal: cluster to minimize variance in data given clusters

– Preserve information

$$\mathbf{c}^*, \ddot{\mathbf{a}}^* = \underset{\mathbf{c},\ddot{\mathbf{a}}}{\operatorname{argmin}} \frac{1}{N} \sum_{j}^{N} \sum_{i}^{K} \delta_{ij} \left(\mathbf{c}_i - \mathbf{x}_j\right)^2$$

Cluster center

Data

Whether x_j is assigned to c_i

Slide: Derek Hoiem

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, $c_1, \ldots, c_K$
  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

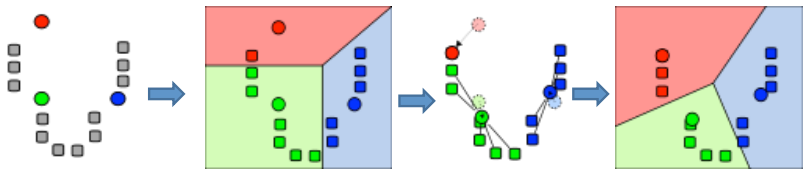
Properties

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

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

Source: Steve Seitz

K-means



0. Initialize Cluster Centers

1. Assign Points to Clusters

2. Re-compute Means

Repeat (1) and (2)

Illustration Source: wikipedia

K-means

1. Initialize cluster centers: $\mathbf{c}^0$; t=0
2. Assign each point to the closest center

$$\ddot{a}^t = \arg\min_{\ddot{a}} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij} \left(\mathbf{c}_i^{t-1} - \mathbf{x}_j \right)^2$$

3. Update cluster centers as the mean of the points

$$\mathbf{c}^t = \arg\min_{\mathbf{c}} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij}^t \left(\mathbf{c}_i - \mathbf{x}_j \right)^2$$

Slide: Derek Hoiem

K-means: design choices

- Initialization
  - Randomly select K points as initial cluster center
  - Or greedily choose K points to minimize residual
- Distance measures
  - Traditionally Euclidean, could be others
- Optimization
  - Will converge to a *local minimum*
  - May want to perform multiple restarts

How to choose the number of clusters?

- Minimum Description Length (MDL) principle for model comparison
- Minimize Schwarz Criterion
  - also called Bayes Information Criteria (BIC)

$$\text{Distortion} + \lambda \, (\#\text{parameters}) \log R$$

$$= \text{Distortion} + \lambda mk \log R$$

m=#dimensions

k=#Centers

R=#Records

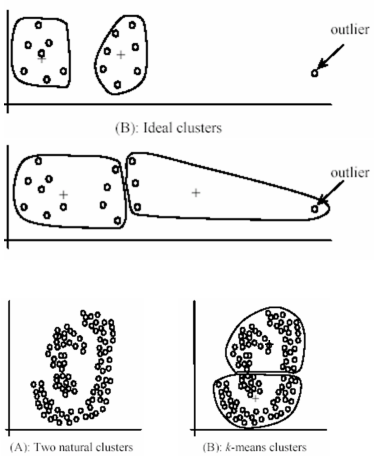
How to choose the number of clusters?

- Validation set
  - Try different numbers of clusters and look at performance
    - When building dictionaries (discussed later), more clusters typically work better

Slide: Derek Hoiem

K-Means pros and cons

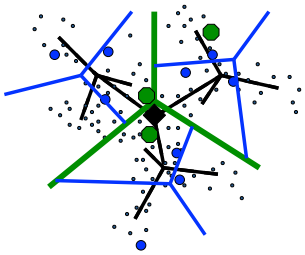
- Pros
  - Finds cluster centers that minimize conditional variance (good representation of data)
  - Simple and fast*
  - Easy to implement
- Cons
  - Need to choose K
  - Sensitive to outliers
  - Prone to local minima
  - All clusters have the same parameters (e.g., distance measure is non-adaptive)
  - *Can be slow: each iteration is O(KNd) for N d-dimensional points
- Usage



Building Visual Dictionaries

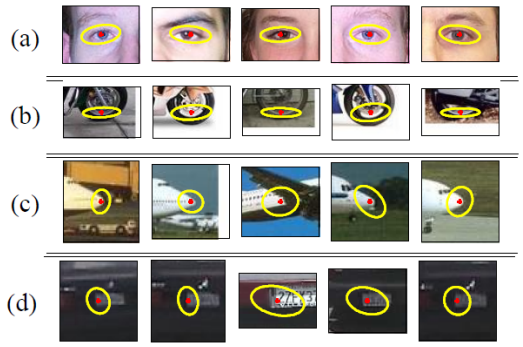
1. Sample patches from a database
   - E.g., 128 dimensional SIFT vectors

2. Cluster the patches
   - Cluster centers are the dictionary



3. Assign a codeword (number) to each new patch, according to the nearest cluster

Examples of learned codewords



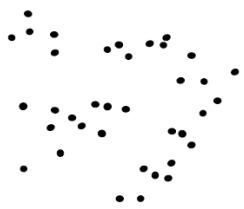
Most likely codewords for 4 learned "topics"
EM with multinomial (problem 3) to get topics

http://www.robots.ox.ac.uk/~vgg/publications/papers/sivic05b.pdf Sivic et al. ICCV 2005

How do we cluster?

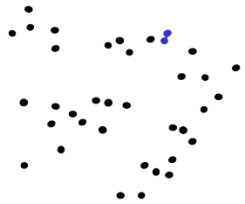
- K-means
  - Iteratively re-assign points to the nearest cluster center
- Agglomerative clustering
  - Start with each point as its own cluster and iteratively merge the closest clusters
- Mean-shift clustering
  - Estimate modes of pdf
- Spectral clustering
  - Split the nodes in a graph based on assigned links with similarity weights

Agglomerative clustering



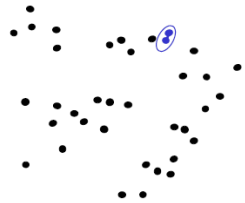
1. Say "Every point is its own cluster"

Agglomerative clustering



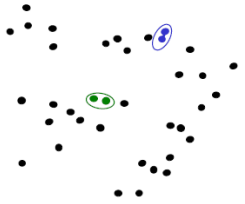
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters

Agglomerative clustering



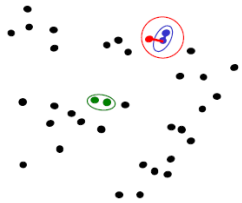
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster

Agglomerative clustering



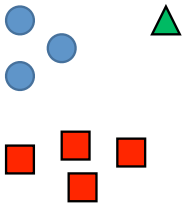
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

Agglomerative clustering



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

Agglomerative clustering

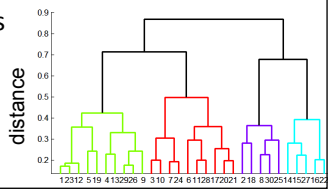


How to define cluster similarity?

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

How many clusters?

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



Conclusions: Agglomerative Clustering

Good

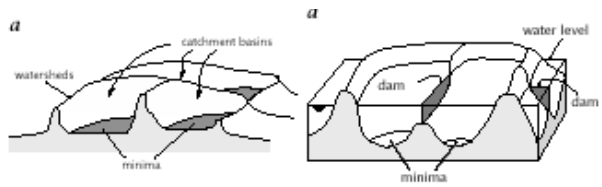
- Simple to implement, widespread application
- Clusters have adaptive shapes
- Provides a hierarchy of clusters

Bad

- May have imbalanced clusters
- Still have to choose number of clusters or threshold
- Need to use an "ultrametric" to get a meaningful hierarchy

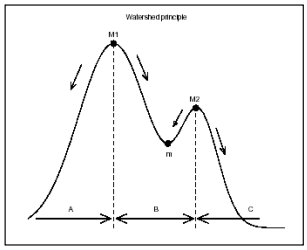
Watershed based region segmentation

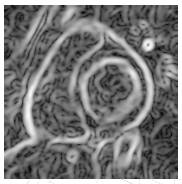
- A *watershed region* or *catchment basin* is defined as the region over which all points flow "downhill" to a common point.
- Watersheds of *gradient magnitude* make useful region- based segmentation primitives.

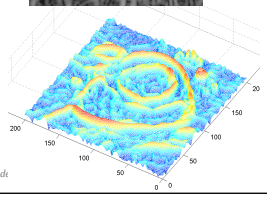


Watershed based region segmentation

Not directly applicable to intensity images, but makes sense when applied to gradient magnitude images

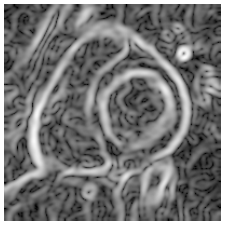


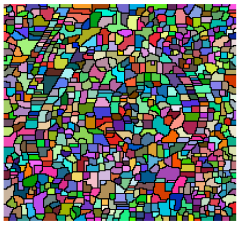




Calculation of WS regions

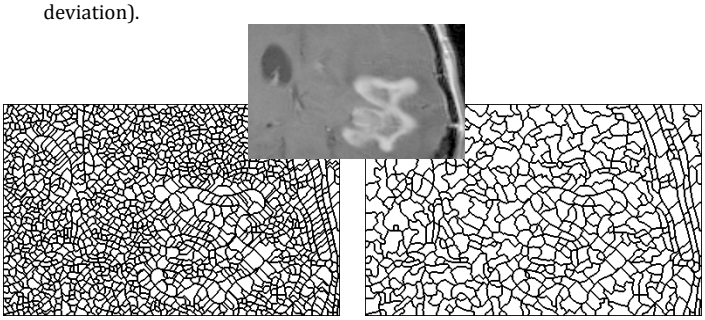
- Work on the *gradient magnitude image* .
  - » Link each pixel to the smallest of its neighbors.
  - » If no smaller neighbors, these are region seeds.
  - » All pixels that "flow" downhill (smallest neighbor) to the same point form a single region.



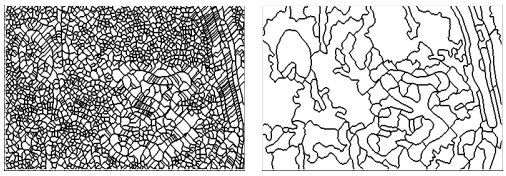


Region Merging

- Reducing the number of regions, combining fragmented regions, determining which regions are really part of the same area.
- Similarity based on e.g. Differences in average (compared to standard deviation).



Region merging based on intensity similarity



number of regions > < region homogeneity

global optimization problem

formulated mathematically using MDL

MDL criterion

" Most compact description is the best "

Description length (in bits):

$$B(\mathcal{R}) = \sum_i B_I(R_i) + B_B(\mathcal{R})$$

data in each region (homogeneity)

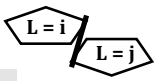
region boundaries (complexity)

? second order (m,s)
? polygonal
! entropy

$$B(\mathcal{R}) = \sum_i n_i.H(R_i) + B_B(\mathcal{R})$$

Region merging using mutual information

Merging of regions i en j ?



$$\delta B(R_i, R_j) = \delta B_I(R_i, R_j) + \delta B_B(R_i, R_j)$$

$$\begin{aligned} \delta B_I(R_i, R_j) &= n_i.H(R_i) + n_j.H(R_j) - (n_i + n_j).H(R_{ij}) \\ &= -(n_i + n_j).I(R_{ij}, L_{ij}) \end{aligned}$$

"Preferably merge regions with small mutual information between region intensity and region label"

Mutual information

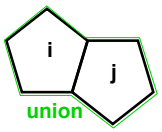
Fundamental concept from information theory

$$I(A,B) = \sum_{a,b} p_{AB}(a,b) \log_2 \frac{p_{AB}(a,b)}{p_A(a).p_B(b)}$$

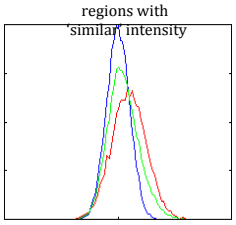
measure of statistical dependence of random variables

$$I(A,B) = H(A) - H(A|B)$$

information of one statistical quantity about another

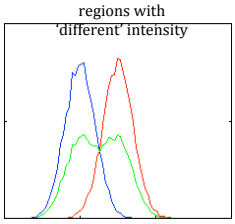


regions with 'similar' intensity



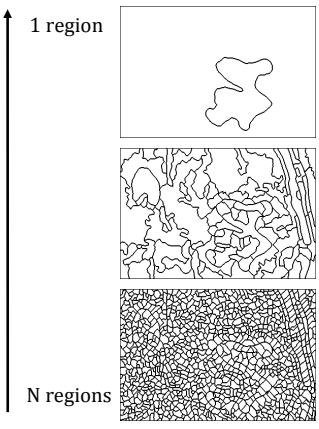
'unimodal' I(R,L) small

regions with 'different' intensity

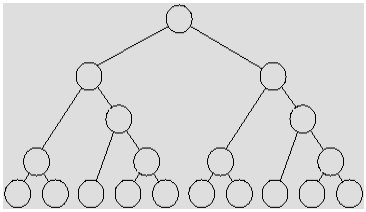


'bimodal' I(R,L) large

Hierarchical representation

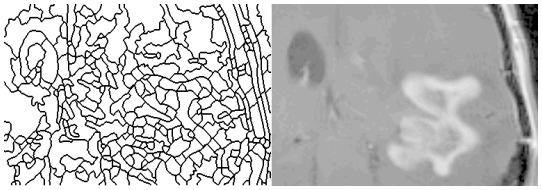


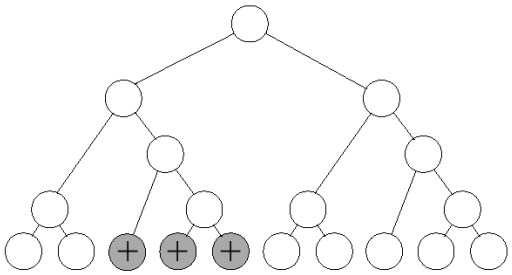
tree of hierarchically connected primitives



supports mechanisms for efficient interaction

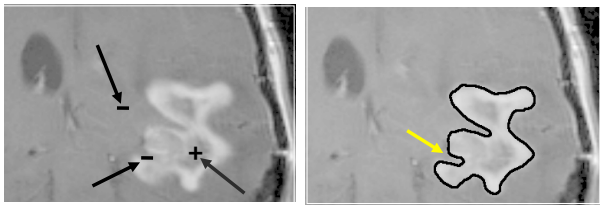
Interaction
‘Paintbrush’

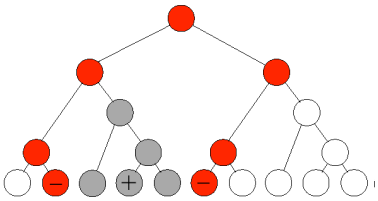




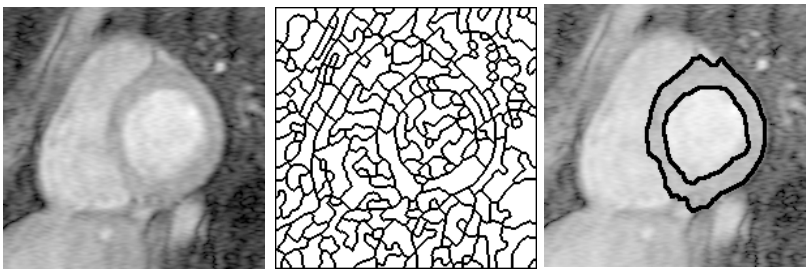
Interaction
‘Point and click’

user indicates points that do or do not belong to the object of interest

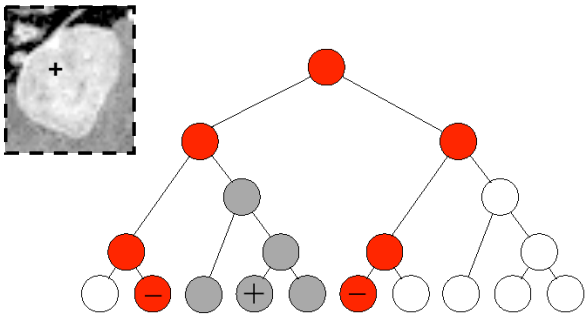




example



ROI + point and click



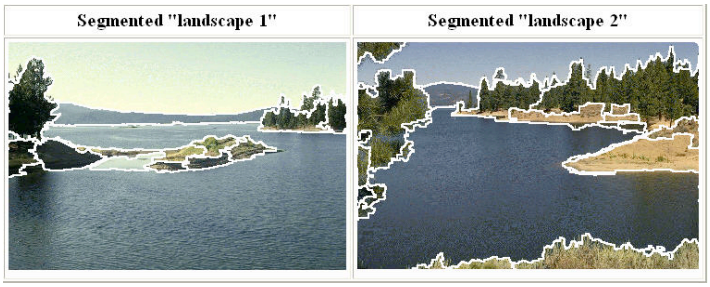
How do we cluster?

- K-means
  - Iteratively re-assign points to the nearest cluster center
- Agglomerative clustering
  - Start with each point as its own cluster and iteratively merge the closest clusters
- Mean-shift clustering
  - Estimate modes of pdf
- Spectral clustering
  - Split the nodes in a graph based on assigned links with similarity weights

Mean shift segmentation

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

- Versatile technique for clustering-based segmentation

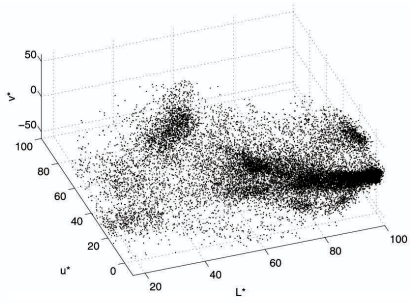


Mean shift algorithm

- The mean shift algorithm seeks *modes* or local maxima of density in the feature space

image

Feature space
(L*u*v* color values)



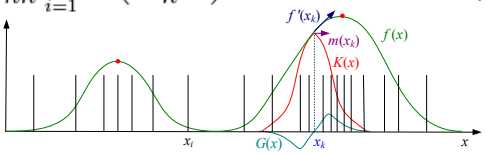
Kernel density estimation

Kernel density estimation function

$$\hat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

Gaussian kernel

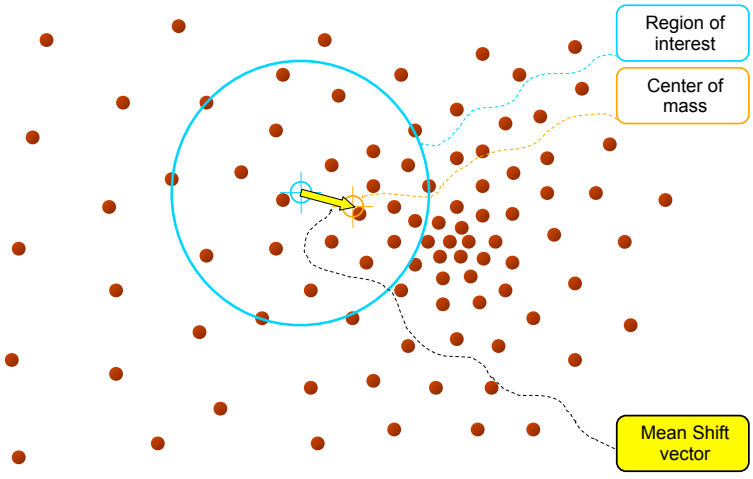
$$K\left(\frac{x - x_i}{h}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-x_i)^2}{2h^2}}.$$



First constructing the kernel density estimate f_h and then searching for its maxima is computationally prohibitive for higher dimensions

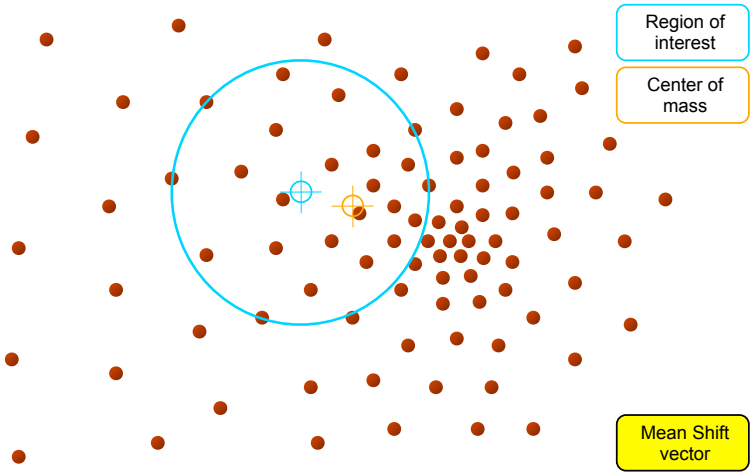
Instead: use Mean Shift trick to directly calculate the gradient ∇f_h without explicitly calculating f_h

Mean shift



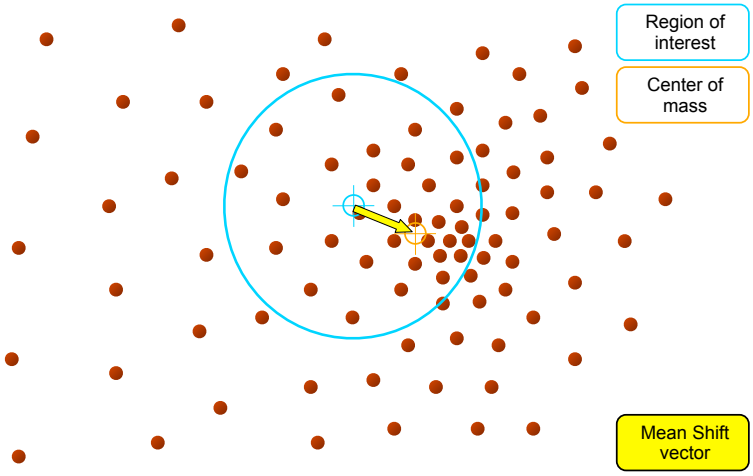
Slide by Y. Ukrainitz & B. Sarel

Mean shift



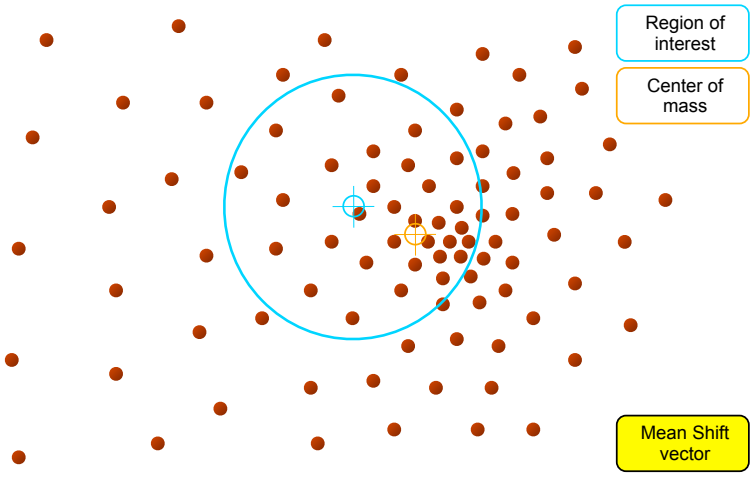
Slide by Y. Ukrainitz & B. Sarel

Mean shift



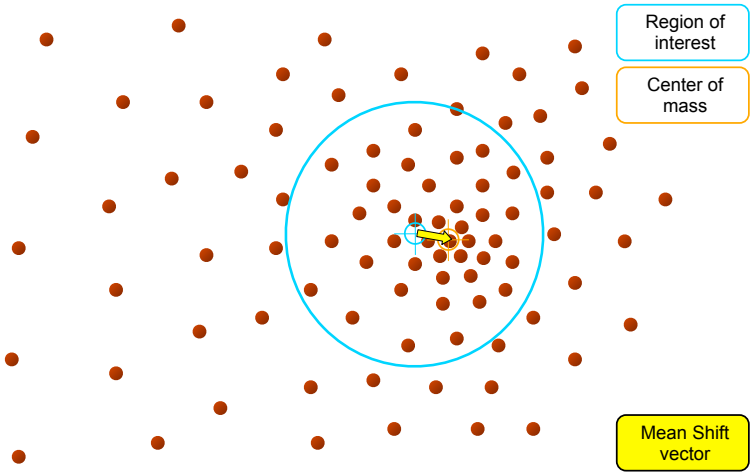
Slide by Y. Ukrainitz & B. Sarel

Mean shift



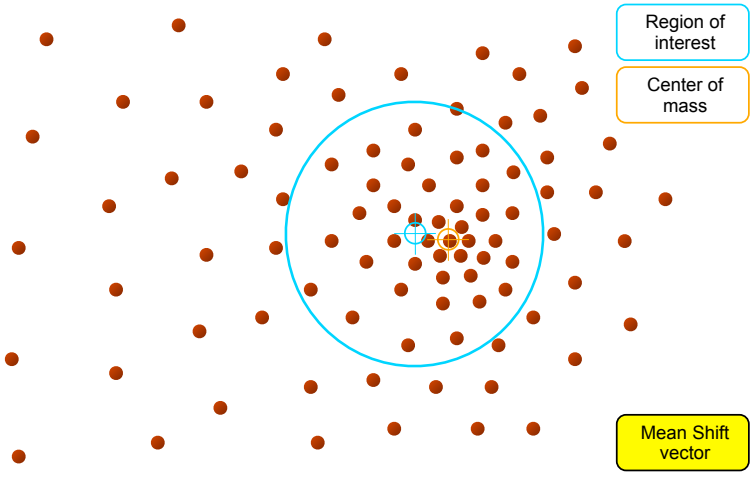
Slide by Y. Ukrainitz & B. Sarel

Mean shift



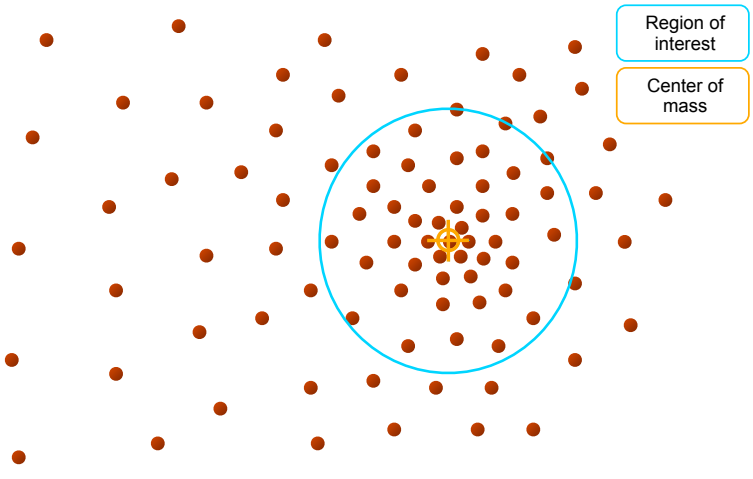
Slide by Y. Ukrainitz & B. Sarel

Mean shift



Slide by Y. Ukrainitz & B. Sarel

Mean shift



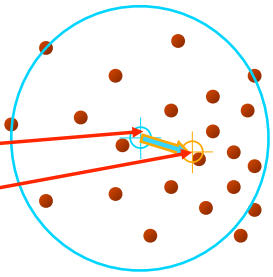
Slide by Y. Ukrainitz & B. Sarel

Computing the Mean Shift

Simple Mean Shift procedure:
- Compute mean shift vector

- Translate the Kernel window by **m(x)**

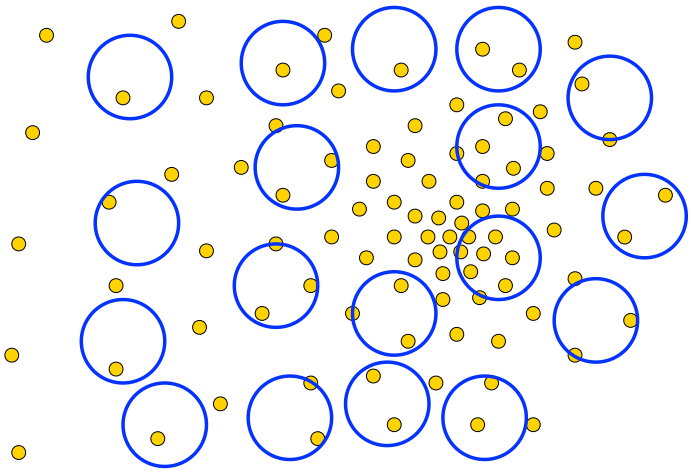
$$\mathbf{m}(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h} \right)}{\sum_{i=1}^{n} g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h} \right)} - \mathbf{x} \right]$$



$g(\mathbf{x}) = -k'(\mathbf{x})$

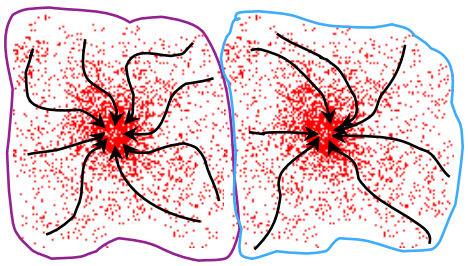
Slide by Y. Ukrainitz & B. Sarel

Real Modality Analysis



Attraction basin

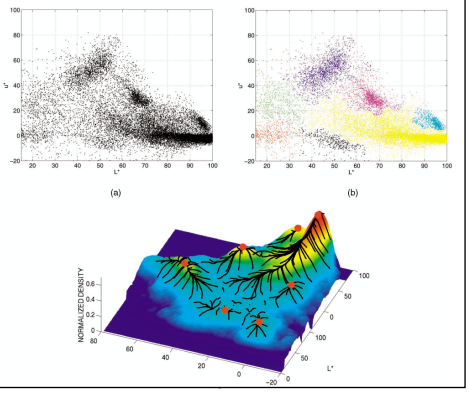
- Attraction basin: the region for which all trajectories lead to the same mode
- Cluster: all data points in the attraction basin of a mode



Slide by Y. Ukrainitz & B. Sarel

Mean shift clustering/segmentation

- Find features (color, gradients, texture, etc)
- Initialize windows at individual feature points
- Perform mean shift for each window until convergence
- Merge windows that end up near the same “peak” or mode



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

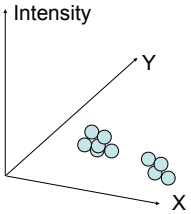
Clusters based on intensity similarity don’t have to be spatially coherent.

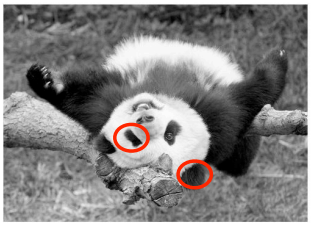
Kristen Grauman

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

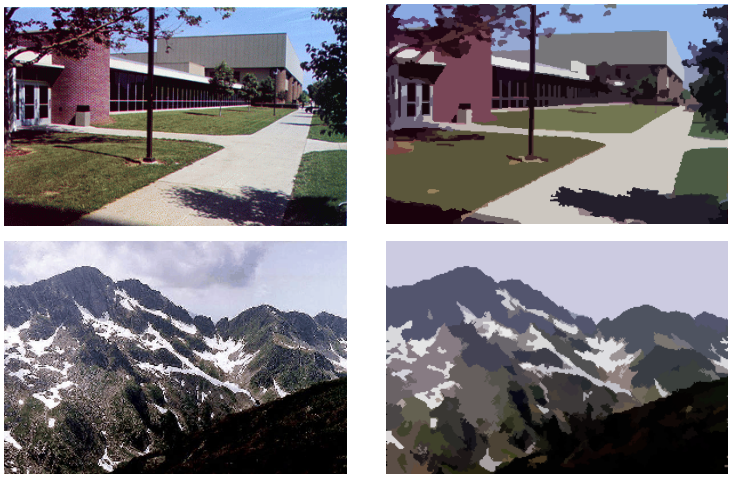




Both regions are black, but if we also include **position (x,y),** then we could group the two into distinct segments; way to encode both similarity & proximity.

Kristen Grauman

Mean shift segmentation results



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

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

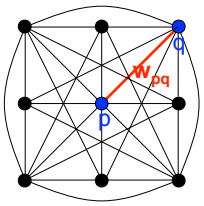
Mean shift pros and cons

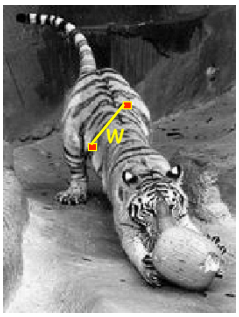
- Pros
  - Good general-practice segmentation
  - Flexible in number and shape of regions
  - Robust to outliers
- Cons
  - Have to choose kernel size in advance
  - Not suitable for high-dimensional features

How do we cluster?

- K-means
  - Iteratively re-assign points to the nearest cluster center
- Agglomerative clustering
  - Start with each point as its own cluster and iteratively merge the closest clusters
- Mean-shift clustering
  - Estimate modes of pdf
- Spectral clustering
  - Split the nodes in a graph based on assigned links with similarity weights

Images as graphs





- node (vertex) for every pixel
- link between *every* pair of pixels, **p**,**q** (or every pair of "sufficiently close" pixels)
- affinity weight w_{pq} for each link (edge)
  - w_{pq} measures *similarity*
    - » similarity is *inversely proportional* to difference (in color and position…)

Source: Steve Seitz

Measuring affinity

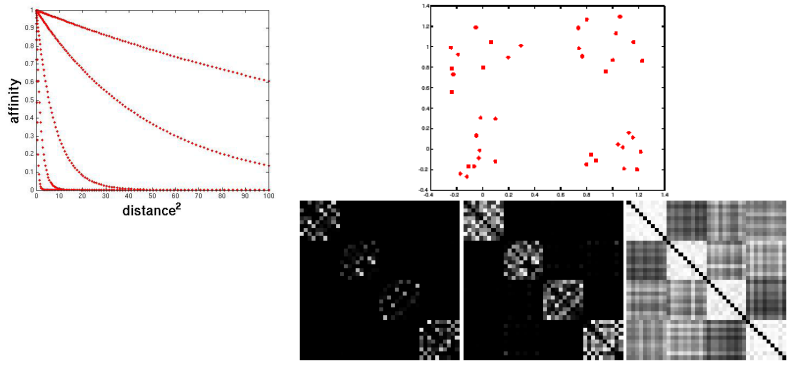
- Suppose we represent each pixel by a feature vector **x**, and define a distance function appropriate for this feature representation
- Then we can convert the distance between two feature vectors into an affinity with the help of a generalized Gaussian kernel:

$$\exp\left(-\frac{1}{2\sigma^2}\text{dist}(\mathbf{x}_i,\mathbf{x}_j)^2\right)$$

Slide credit: S. Lazebnik

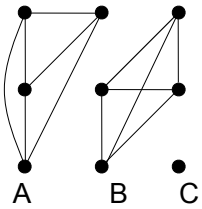
Scale affects affinity

- Small σ: group only nearby points
- Large σ: group far-away points



Slide credit: S. Lazebnik

Segmentation by Graph Cuts

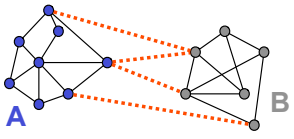


Break Graph into Segments

- Want to delete links that cross **between** segments
- Easiest to break links that have low similarity (low weight)
  - similar pixels should be in the same segments
  - dissimilar pixels should be in different segments

Source: Steve Seitz

Graph cut



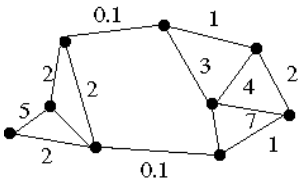
- Set of edges whose removal makes a graph disconnected
- Cost of a cut: sum of weights of cut edges
- A graph cut gives us a segmentation
  - What is a "good" graph cut and how do we find one?

Source: S. Seitz

Minimum cut

- We can do segmentation by finding the *minimum cut* in a graph
  - Efficient algorithms exist for doing this

Minimum cut example

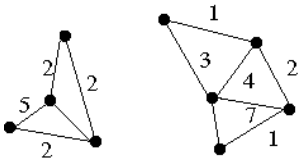


Slide credit: S. Lazebnik

Minimum cut

- We can do segmentation by finding the *minimum cut* in a graph
  - Efficient algorithms exist for doing this

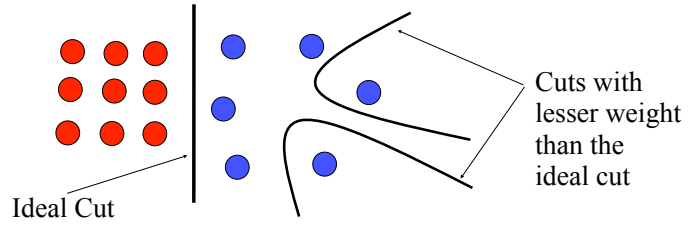
Minimum cut example



Slide credit: S. Lazebnik

Normalized cut

- Drawback: minimum cut tends to cut off very small, isolated components



* Slide from Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Normalized cut

- Drawback: minimum cut tends to cut off very small, isolated components
- This can be fixed by normalizing the cut by the weight of all the edges incident to the segment
- The *normalized cut* cost is:

$$\frac{w(A,B)}{assoc(A,V)} + \frac{w(A,B)}{assoc(B,V)}$$

$w(A, B)$ = sum of weights of all edges between A and B
$assoc(A,V)$ = sum of all weights in cluster A + w(A,B)

J. Shi and J. Malik. Normalized cuts and image segmentation. PAMI 2000

Normalized cut

- Finding the exact minimum of the normalized cut cost is NP-complete, but we *relax* to let nodes take on arbitrary values:
- Let W be the adjacency matrix of the graph
- Let D be the diagonal matrix with diagonal entries $D(i, i) = \Sigma_j W(i, j)$
- Then the normalized cut cost can be written as

$$\frac{y^T(D-W)y}{y^T Dy}$$

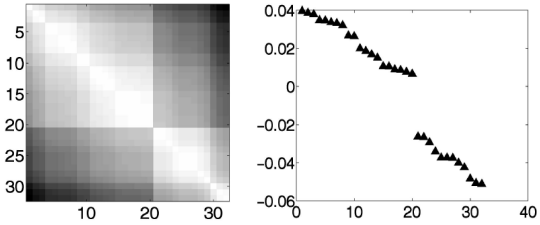
where y is an indicator vector whose value should be 1 in the ith position if the ith feature point belongs to A and a negative constant otherwise

J. Shi and J. Malik. Normalized cuts and image segmentation. PAMI 2000

Normalized cut

- We can minimize the relaxed cost by solving the *generalized eigenvalue problem* $(D - W)y = \lambda Dy$
- The solution y is given by the generalized eigenvector corresponding to the second smallest eigenvalue
- Intuitively, the ith entry of y can be viewed as a "soft" indication of the component membership of the ith feature
  - Can use 0 or median value of the entries as the splitting point (threshold), or find threshold that minimizes the Ncut cost

Example eigenvector



(a) (b)

Figure 5.20 Sample weight table and its second smallest eigenvector (Shi and Malik 2000) © 2000 IEEE: (a) sample 32×32 weight matrix $\boldsymbol{W}$; (b) eigenvector corresponding to the second smallest eigenvalue of the generalized eigenvalue problem $(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{y} = \lambda \boldsymbol{D}\boldsymbol{y}$.

Normalized cut algorithm

1. Represent the image as a weighted graph $G = (V,E)$, compute the weight of each edge, and summarize the information in D and W
2. Solve $(D - W)y = \lambda Dy$ for the eigenvector with the second smallest eigenvalue
3. Use the entries of the eigenvector to bipartition the graph

To find more than two clusters:

- Recursively bipartition the graph
- Run k-means clustering on values of several eigenvectors

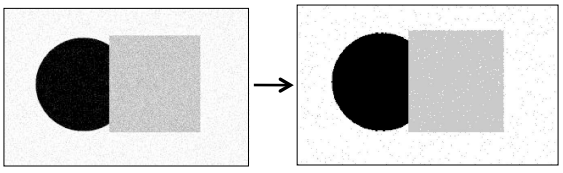
Example results

How do we cluster?

- K-means
  - Iteratively re-assign points to the nearest cluster center
- Agglomerative clustering
  - Start with each point as its own cluster and iteratively merge the closest clusters
- Mean-shift clustering
  - Estimate modes of pdf
- Spectral clustering
  - Split the nodes in a graph based on assigned links with similarity weights
- Graph-cuts and energy-based methods

Smoothing out cluster assignments

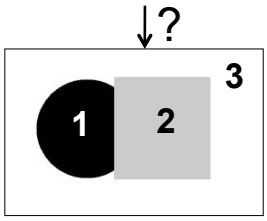
- Assigning a cluster label per pixel may yield outliers:



original

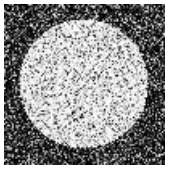
labeled by cluster center's intensity

- How to ensure they are spatially smooth?



Kristen Grauman

Solution



P(foreground | image)

Encode dependencies between pixels

Normalizing constant

$$P(\mathbf{y};\theta,data) = \frac{1}{Z}\prod_{i=1..N} f_1(y_i;\theta,data)\prod_{i,j\in edges} f_2(y_i,y_j;\theta,data)$$

Labels to be predicted — Individual predictions — Pairwise predictions

Slide: Derek Hoiem

Writing Likelihood as an "Energy"

$$P(\mathbf{y};\theta,data) = \frac{1}{Z}\prod_{i=1..N} p_1(y_i;\theta,data)\prod_{i,j\in edges} p_2(y_i,y_j;\theta,data)$$

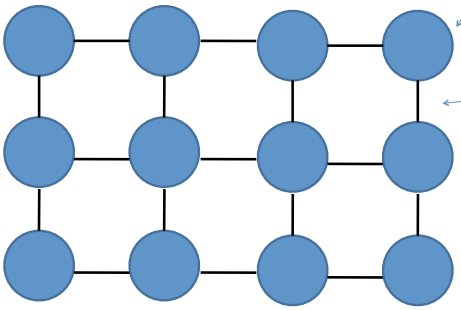
$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) + \sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

"Cost" of assignment y_i

"Cost" of pairwise assignment y_i, y_j

Slide: Derek Hoiem

Markov Random Fields



Node y_i: pixel label

Edge: constrained pairs

Cost to assign a label to each pixel

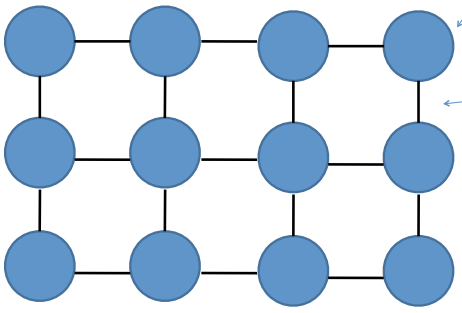
Cost to assign a pair of labels to connected pixels

$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) + \sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

Slide: Derek Hoiem

Markov Random Fields

- Example: "label smoothing" grid



Unary potential

0: -logP(y_i = 0 ; data)

1: -logP(y_i = 1 ; data)

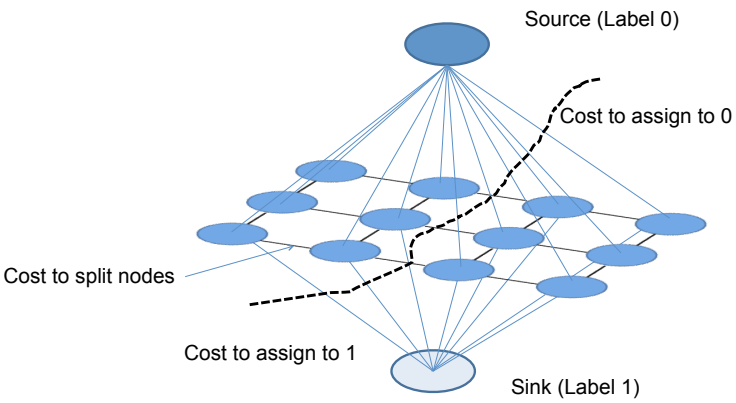
Pairwise Potential

	0	1
0	0	K
1	K	0

$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) + \sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

Slide: Derek Hoiem

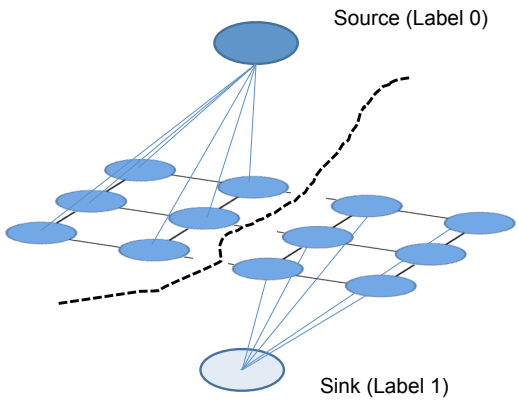
Solving MRFs with graph cuts



$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) + \sum_{i,j\in edges} \psi_2(y_i,y_j;\theta,data)$$

Slide: Derek Hoiem

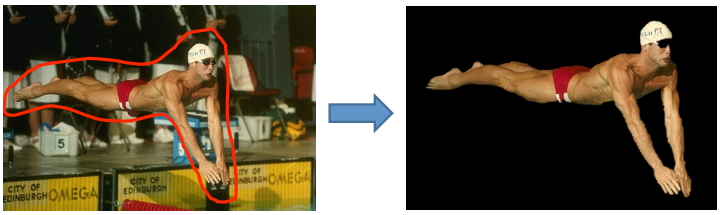
Solving MRFs with graph cuts



$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) + \sum_{i,j\in edges} \psi_2(y_i,y_j;\theta,data)$$

Slide: Derek Hoiem

GrabCut segmentation

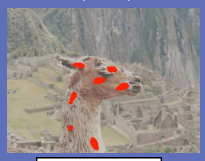
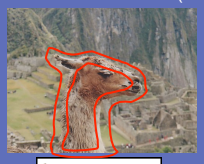
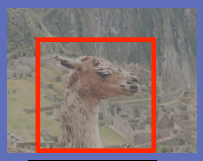


User provides rough indication of foreground region.

Goal: Automatically provide a pixel-level segmentation.

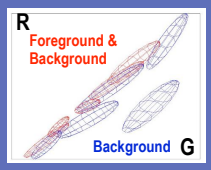
Slide: Derek Hoiem

Grab cuts and graph cuts

	Magic Wand (198?)	Intelligent Scissors Mortensen and Barrett (1995)	GrabCut
User Input			
Result			
	Regions	Boundary	Regions & Boundary

Source: Rother

Colour Model



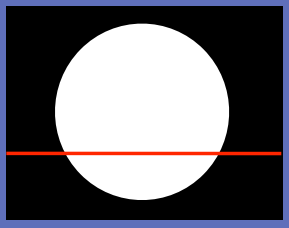
Gaussian Mixture Model (typically 5-8 components)

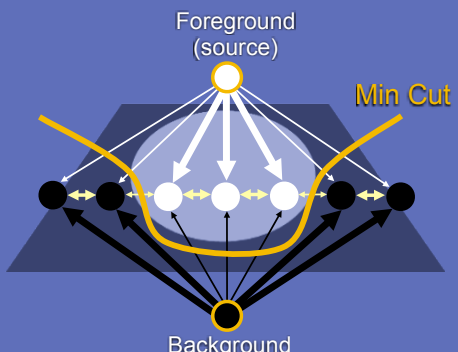
Source: Rother

Graph cuts

Boykov and Jolly (2001)

Image





Cut: separating source and sink; Energy: collection of edges

Min Cut: Global minimal enegry in polynomial time

Source: Rother

GrabCut segmentation

1. Define graph
   - usually 4-connected or 8-connected
2. Define unary potentials
   - Color histogram or mixture of Gaussians for background and foreground

$$unary\_potential(x) = -\log\left(\frac{P(c(x);\theta_{foreground})}{P(c(x);\theta_{background})}\right)$$

3. Define pairwise potentials

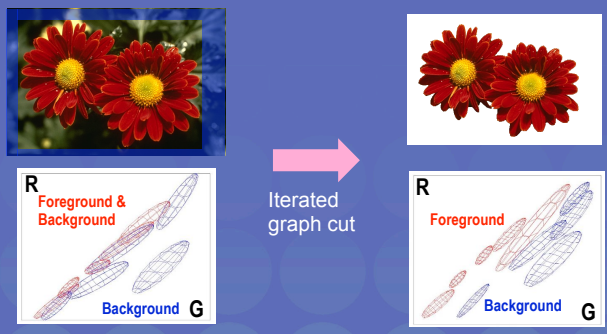
$$edge\_potential(x,y) = k_1 + k_2 \exp\left\{\frac{-\|c(x)-c(y)\|^2}{2\sigma^2}\right\}$$

4. Apply graph cuts
5. Return to 2, using current labels to compute foreground, background models

Slide: Derek Hoiem

Colour Model

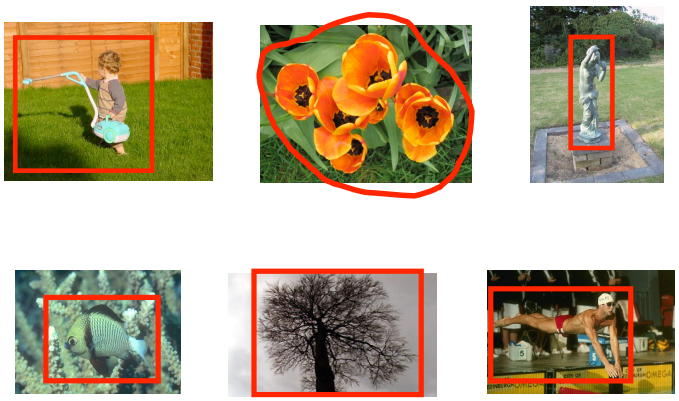
SIGGRAPH2004



Gaussian Mixture Model (typically 5-8 components)

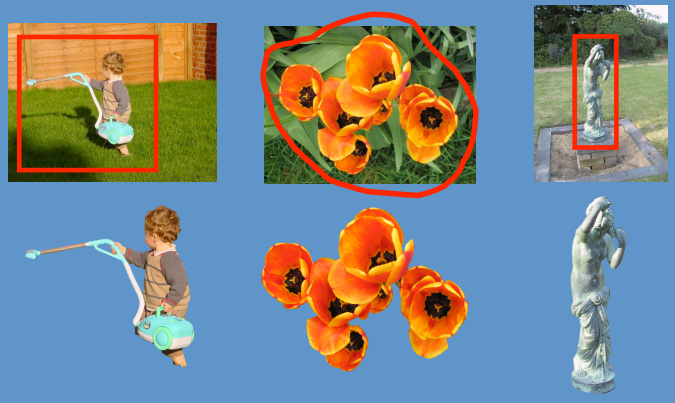
Source: Rother

What is easy or hard about these cases for graphcut-based segmentation?

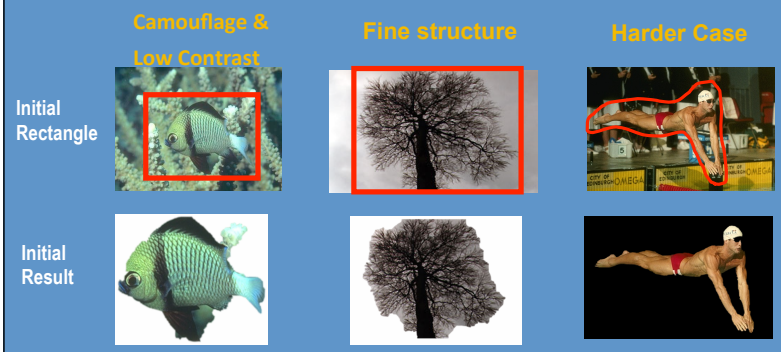


Slide: Derek Hoiem

Easier examples



More difficult Examples



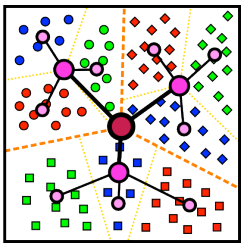
Graph cuts: Pros and Cons

- Pros
  - Very fast inference
  - Can incorporate data likelihoods and priors
  - Applies to a wide range of problems (stereo, image labeling, recognition)
- Cons
  - Not always applicable (associative only)
  - Need unary terms (not used for generic segmentation)
- Use whenever applicable

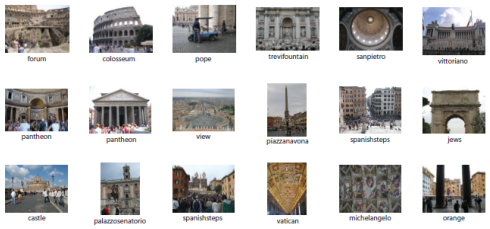
Slide: Derek Hoiem

Which algorithm to use?

- Quantization/Summarization: K-means
  - Aims to preserve variance of original data
  - Can easily assign new point to a cluster



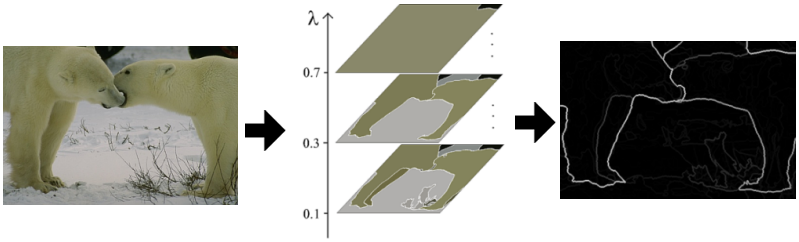
Quantization for computing histograms



Summary of 20,000 photos of Rome using "greedy k-means"
http://grail.cs.washington.edu/projects/canonview/

Which algorithm to use?

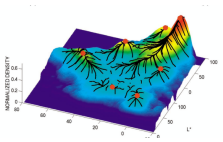
- Image segmentation: agglomerative clustering
  - More flexible with distance measures (e.g., can be based on boundary prediction)
  - Adapts better to specific data
  - Hierarchy can be useful

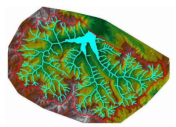


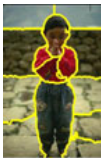
http://www.cs.berkeley.edu/~arbelaez/UCM.html

Segmentation

- Mean-shift segmentation
  - Flexible clustering method, good segmentation
- Watershed segmentation
  - Hierarchical segmentation from soft boundaries
- Normalized cuts
  - Produces regular regions
  - Slow but good for oversegmentation
- MRFs with Graph Cut
  - Incorporates foreground/background/object model and prefers to cut at image boundaries
  - Good for interactive segmentation or recognition







Slide: Derek Hoiem